Network Manager IP Edition
4.2

Administration Guide



2021-4213-01

Note

Before using this information and the product it supports, read the information in "Notices" on page 649.

This edition applies to version 4.2 of IBM Tivoli Network Manager IP Edition (product number 5724-S45) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2006, 2021.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this publication ..... **xi**
Publications xi
Accessibility xii
Tivoli technical training xiii
Support and community information xiv

Part 1. Administering **1**

Chapter 1. Starting and stopping Network Manager 3
Setting environment variables 3
Starting Network Manager 4
Starting all Network Manager components (UNIX only) 4
Starting Network Manager processes using the command console 5
Stopping Network Manager 5
Stopping all Network Manager components (UNIX only) 5
Stopping Network Manager processes using the command console 6
Restarting the Dashboard Application Services Hub server 6

Chapter 2. Administering processes 9
About process control 9
Network Manager processes 9
About Network Manager domains 13
Domain-specific configuration files 13
Checking process status 14
Running the itnm_status command 14
Monitoring process status messages 14
Checking process status by querying ncp_ctrl databases 15
Managing process dependencies 19
Listing process dependencies 19
Identifying dependencies for a particular process 19
Configuring process dependencies 20
List of process dependencies 20
Process control configuration files 21
Starting and stopping processes 21
Configuring managed processes 22
Starting unmanaged processes 23
Stopping managed processes 23
Running processes remotely 24

Chapter 3. Administering logs 25
Setting up logging for GUI 25
GUI component log file overview 25
Locating GUI log files 27
Changing the logging level for GUIs 28
Setting the log file size 31
Setting up logging for processes 31
Process log file overview 31
Locating log files for a process 32
Changing the logging level for processes 32
Using log file rotation to avoid large file errors 33

Chapter 4. Administering ports 39
About inter-process communication 39
About Really Small Message Broker 39
About multicast 40
Changing host and port settings for Really Small Message Broker 40
Updating the Really Small Message Broker configuration file 40
Stopping Really Small Message Broker 41
Running a separate message broker for each domain 41
Checking port usage 41
Defining a fixed TCP port 42
Defining a fixed multicast address 42
List of ports used by the product 43
ServiceData configuration file 44

Chapter 5. Administering users 45
Default users 45
Network Manager user roles 46
User groups 52

Chapter 6. Administering system passwords 53
Encrypting or decrypting a password manually 53
Changing the encryption key 54
Deactivating password encryption 54
List of passwords in Network Manager 55

Chapter 7. Administering management databases 57
Querying management databases using the Management Database Access page 57
Logging into the Management Database Access page 57
Issuing a query using the Management Database Access page 57
Listing the databases and tables of the current service 58
Querying management databases from the command line 59
Starting the OQL Service Provider 59
Listing the databases and tables of the current service 60
Using OQL queries in scripts 61
Exiting the OQL Service Provider 62
OQL Service Provider tips 62
Show history of commands 62
Execute a previous command 62
Turn on tabular display mode 63
Turn off tabular display mode 63

Chapter 8. Administering the NCIM topology database 65
Changing the NCIM access details 65
Updating NCIM access settings for the Web applications 65
Updating the NCIM access details used by Reporting Services 66
Updating NCIM access settings in the Network Manager core components 66
Re-creating network views 67
Creating the topology database schemas 67
Creating Db2 topology database schemas on UNIX 68
Creating Oracle topology database schemas on UNIX 69
Removing domains from NCIM 71
Removing all entities from domains 72
Removing the topology database 72
Removing a Db2 topology database on UNIX 72
Removing an Oracle topology database on UNIX 72
Removing an Oracle topology database on Windows 73

Chapter 9. Administering reports....75
Creating and editing reports....75
Creating a URL to run reports....75
Changing the data source isolation level....76

Chapter 10. Troubleshooting and support....77
Troubleshooting Network Manager....77
Troubleshooting the installation....77
Troubleshooting the Dashboard Application Services Hub....78
Troubleshooting Web Applications....78
Troubleshooting reporting....80
Troubleshooting database access....81
Troubleshooting the ITM Agent....82
Trace logging....83
Problems and workarounds....90
IBM Tivoli Monitoring Agent Reference....99

Chapter 11. Command reference....131
Commands to start Network Manager....131
itnm_status....131
itnm_start....132
itnm_stop....133
Restarting the Dashboard Application Services Hub server....134
Process control....134
ncp_ctrl....134
Commands for processes under process control....135
Commands for processes not started by default....151
ncp_crypt....151
ncp_mib....152
ncp_oql....154
ncp_trapmux....156

Part 2. Discovering the network....159

Chapter 12. About discovery....161
About types of discovery....161
Scopes....161
Types of scoping....162
Defining discovery zones to restrict discovery....162
Seeds....171
Device access....172
Agents....172
Device filters....172
SNMP interface filters....173
Domain Name System....174
Network address translation....175
Advanced settings....175
Context-sensitive discovery....175
Helpers....175
Collectors....176
Specialized discoveries....176

Chapter 13. Configuring network discovery....177
Planning for discovery....177
Configuring standard discoveries....178
Using the wizard....178

Using the GUI....182
Using the command-line interface....205
Configuring backups of the ncp_store cache....244
Configuring specialized discoveries....244
Configuring a context-sensitive discovery....245
Configuring EMS discoveries....245
Configuring MPLS discoveries....325
Configuring NAT discoveries....336
Configuring cross-domain discoveries....348
Configuring geographical discoveries....357
Configuring IP SLA discoveries....365

Chapter 14. Monitoring network discoveries....367
From the GUI....367
Monitoring full and partial discoveries....367
Monitoring discovery agent progress....369
From the command line....372
Monitoring full and partial discoveries....372

Chapter 15. Classifying network devices....383
Changing the device class hierarchy....383
Listing the existing device classes....383
Creating and editing AOC files....384
Applying AOC changes to the topology and to the reports....386
AOC file samples....388
EndNode class....388
NetworkDevice class....388
AOC specific to device class....389
Entity types....390

Chapter 16. Keeping discovered topology up-to-date....401
Scheduling discoveries....401
Viewing discovery status for a device....401
Manually discovering a device or subnet....402
Manually discovering a device or subnet using the GUI....403
Manually discovering a device or subnet from the command line....405
Removing a device from the network....406
Setting the linger time for a device....406

Chapter 17. Troubleshooting discovery....407
Troubleshooting discovery with reports....407
Monitoring discovery status....408
Process flow for creating discovery events....408
Monitoring discovery status messages....409
NCIM entity ID retrieval failure....409
Troubleshooting discovery agents....410
Troubleshooting an unusually long discovery....410
Identifying failed agents....412
Troubleshooting missing devices....412
Troubleshooting an idle discovery....413
Repairing a corrupted discovery....413
Removing discovery cache files....414
Troubleshooting illegal characters....415

Chapter 18. Discovering and using custom data....417
Reasons for adding custom data....417
Discovering custom data....418
Using the File finder....418

Developing agents or collectors to retrieve custom data....420
Using custom tag tables....420
Storing custom data as name-value pairs....426
Importing name-value pairs into the DNCIM discovery database....427
Preserving custom name-value pairs between discoveries....428
Storing custom data in new database tables....428
Creating new NCIM database tables....429
Updating dNCIM to store custom data....430
Mapping retrieved data to DNCIM custom data tables....431
Using custom data to enrich events....433
Visualizing custom data in the **Structure Browser**....434
Using custom data in polling....435
Visualizing custom data in the topology views....435

Part 3. Polling the network....437

Chapter 19. About polling the network....439
Poll policies....439
Parameters....439
Scope....440
Poll definitions....441
Parameters....441
Polling mechanisms....442
Poll definition types....445
Data labels....446
Ping polling properties and metrics....447
Multibyte data in poll definitions....447

Chapter 20. Enabling and disabling polls....449

Chapter 21. Creating polls....451
Creating fully featured poll policies....451
Creating simple poll policies....456
Default poll policies....457
Default ping policies....457
Default remote ping policies....458
Default SNMP threshold policies....458
Default SNMP link state policies....461

Chapter 22. Creating new poll definitions....463
Creating basic threshold poll definitions....463
Creating generic threshold poll definitions....466
Creating chassis and interface ping poll definitions....467
Creating remote ping and link state poll definitions....469
Default poll definitions....470
Example trigger and clear thresholds....476

Chapter 23. Changing polls....479
Changing poll policies....479
Example poll policy....482
Changing poll definitions....483
Changing basic threshold poll definitions....483
Changing generic threshold poll definitions....486
Changing chassis and interface ping poll definitions....488
Changing remote ping and link state poll definitions....489
Example customized poll definition....491
Example basic threshold expression....492

Example generic threshold expression....492

Chapter 24. Deleting poll policies....495

Chapter 25. Deleting poll definitions....497

Chapter 26. Managing adaptive polling....499
Adaptive polling scenarios....499
Rapid confirmation that device is really down....499
Rapid confirmation of a threshold violation....501
Creating adaptive polls....503

Chapter 27. Administering network polling....507
Administering polls....507
Configuring checking of SNMP credentials....507
Retrieving poll status....507
Enabling and disabling polls....508
Refreshing polls....508
Copying polls across domains....509
Polling suspension options....510
Adjusting polling bandwidth....510
Configuring checking of SNMP credentials....513
Configuring Link State polling....513
Administering multiple pollers....514
Multiple poller overview....514
Setting up an additional poller....514
Removing a poller....516
Administering historical poll data....517
About poll data aggregation....517
Capacity guidelines for historical data polling....519
Starting and stopping Apache Storm....521
Configuring poll data aggregation....521
Monitoring poller capacity....522

Chapter 28. Troubleshooting network polling....527
Troubleshooting ping polling of the network....527
Troubleshooting SNMP polling....528
Troubleshooting the processing of historical poll data....529

Chapter 29. About event enrichment and correlation....531
Event enrichment....531
Quick reference for event enrichment....531
Event filtering....533
Event states....539
Event handling....542
Example: Default enrichment of a Tivoli Netcool/OMNIbus trap event....563
RCA Plugin....565
Quick reference for RCA....566
Precedence value....566
Poller entity....568
RCA and unmanaged status....569
RCA stitchers....570
Examples of root cause analysis....574
Checking topology paths used by RCA....584
RCA plug-in subscriptions....588
Other plugins....588
Adaptive polling plug-in....590
Compatibility Check plug-in....593

Disco plug-in....594
Failover plug-in....596
PostNCIMProcessing plug-in....597
SAE Aggregated Link plug-in....598
SAE IP Path plug-in....598
SAE ITNM Service plug-in....599
SAE MPLS VPN plug-in....599
zNetView plug-in....600

Chapter 30. Configuring event enrichment....603
Configuring extra event enrichment....603
Modifications to the ObjectServer alerts.status table....603
Example: Enriching an event with main node device location....603
Example: Enriching an event with interface name....605
Configuring the ObjectServer update interval field....606

Chapter 31. Using the OQL service provider to log into the Event Gateway databases....609
Querying the ObjectServer....609
Querying the NCIM database....609

Chapter 32. Resynchronizing events with the ObjectServer....611

Chapter 33. Configuring common Event Gateway properties....613

Chapter 34. Network Manager event categories....615
Network Manager network events....615
Network Manager status events....616

Chapter 35. Configuring Event Gateway plug-ins....621
Enabling and disabling plugins....621
Listing plug-in information....622
Modifying event map subscriptions....623
Setting plug-in configuration parameters....625
Configuring the SAE plug-in....627
Configuring summary field information in service-affected events....627
Adding SAE types to the SAE plug-in....627
Configuring the Disco plug-in....628

Chapter 36. Configuring root-cause analysis....631
Configuring the poller entity....631
Configuring the maximum age difference for events....632

Chapter 37. Configuration of the Probe for Tivoli Netcool/OMNIbus....635
About the nco_p_ncpmonitor.props file....635
nco_p_ncpmonitor.rules configuration reference....636
Example of rules file processing....637
Network Manager event data fields....639
alerts.status fields used by Network Manager....642

Notices....649
Trademarks....650

About this publication

The *IBM Tivoli Network Manager IP Edition Administration Guide* describes administration tasks such as how to start and stop the product, discover the network, poll the network, manage events, administer processes, and query databases. This publication is for administrators who are responsible for the maintenance and availability of Network Manager.

Publications

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

Your Network Manager library

The following documents are available in the Network Manager library:

- The *IBM Tivoli Network Manager IP Edition Release Notes* give important and late-breaking information about Network Manager. This publication is for deployers and administrators, and should be read first.
- The *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*describes how to install Network Manager. It also describes necessary and optional post-installation configuration tasks. This publication is for administrators who need to install and set up Network Manager.
- The *IBM Tivoli Network Manager IP Edition Administration Guide* describes administration tasks such as how to start and stop the product, discover the network, poll the network, manage events, administer processes, and query databases. This publication is for administrators who are responsible for the maintenance and availability of Network Manager.
- The *IBM Tivoli Network Manager Reference* contains reference information including the system languages, databases, and Perl API used by Network Manager. This publication is for advanced users who need to customize the operation of Network Manager.

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*

  Includes installation and upgrade procedures 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*

  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*

  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*

  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*

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

© Copyright IBM Corp. 2006, 2021

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 products, as they become available and whenever they are updated, to the IBM Knowledge Center Web site at:

http://www.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 IBM publications online at the following Web site:

http://www.ibm.com/e-business/linkweb/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 IBM publications. To locate the telephone number of your local representative, perform the following steps:

1. Go to the following Web site:

   http://www.ibm.com/e-business/linkweb/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

Network Manager includes the following major accessibility features:

- Operations that use a screen reader.

Network Manager uses IBM Installation Manager to install the product. You can read about the accessibility features for IBM Installation Manager at https://www.ibm.com/support/knowledgecenter/SSDV2W/im_family_welcome.html.

Network Manager uses the latest W3C Standard, http://www.w3.org/TR/wai-aria/, to ensure compliance to http://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards), and http://www.w3.org/TR/WCAG20/. To take advantage of accessibility features, use the latest release of your screen reader in combination with the latest web browser that is supported by this product.

The Network Manager online product documentation in IBM Knowledge Center is enabled for accessibility. The accessibility features of IBM Knowledge Center are described at https://www.ibm.com/support/knowledgecenter/v1/content/about/releasenotes.html#accessibility.

Keyboard navigation

This product uses standard navigation keys.

Interface information

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:

- The Network Manager user interfaces do not have content that flashes more than two times in any one second period.

The Network Manager web user interface includes WAI-ARIA navigational landmarks that you can use to quickly navigate to functional areas in the application.

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 -**.

Related accessibility information

In addition to standard IBM help desk and support websites, IBM has established a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

TTY service
800-IBM-3383 (800-426-3383)
(within North America)

IBM and accessibility

For more information about the commitment that IBM has to accessibility, see https://www.ibm.com/able.

Tivoli technical training

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

https://www.ibm.com/training/search?query=tivoli

Support and community information

Use IBM Support, Service Management Connect, and Tivoli user groups to connect with IBM and get the help and information you need.

IBM Support

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 https://www.ibm.com/support/home/ 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 https://www.ibm.com/support/knowledgecenter/SSLLVC/welcome/isa_welcome.html

Part 1. Administering Network Manager

To administer the product, you start and stop it, and administer processes, logs, ports, users, passwords, reports, and databases.

About this task

Configuration tasks are usually done only once, in order to set up the product.

Configuration tasks are described in the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

In contrast, administration tasks are performed as needed on an ongoing basis. Administration tasks are described in this section.

Related tasks

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.

© Copyright IBM Corp. 2006, 2021

Chapter 1. Starting and stopping Network Manager

Your options for starting and stopping Network Manager are explained here.

About this task

Setting environment variables

Before starting any components or working with any configuration files, set the appropriate environment variables by running the environment scripts.

About this task

The environment scripts set the following required environment variables. The environment scripts are configured automatically with the correct locations where the components are installed. Other environment variables are set automatically when necessary by Network Manager components.

$NCHOME
The Netcool® home location that defaults to `netcool` directory under the installation directory:

- `/opt/IBM/netcool/core`

$NMGUI_HOME
The location where the Network Manager GUI components are installed. By default, this location is `/opt/IBM/netcool/gui/precision_gui`.

$ITNMHOME and $PRECISION_HOME
The Network Manager home location that defaults to $NCHOME/`precision`:

- `/opt/IBM/netcool/core/precision`

Note: The script also sets $PRECISION_HOME. By default, $PRECISION_HOME is set to the same location as $ITNMHOME, but is used by other parts of the product.

$DASH_HOME
The installation location of Dashboard Application Services Hub. By default, this location is `/opt/IBM/JazzSM/ui`.

$JazzSM_HOME
The installation location of Jazz for Service Management. By default, this location is `/opt/IBM/JazzSM`.

To set the Network Manager environment variables, run the appropriate script for the components that you installed. There are different scripts on the server where the core components are installed and the server where the GUI components are installed.

Important: If you have installed the core components and the GUI components on one server, run both scripts.

Procedure

- Run the appropriate environment script:

  On the server where the Network Manager core components are installed, the environment script is `installation_directory/netcool/core/env.sh`.

  For example, on Bash, Bourne, and Korn shells, source the env.sh script using a command similar to the following:

```
. /opt/IBM/netcool/core/env.sh
```

© Copyright IBM Corp. 2006, 2021

On the server where the Network Manager GUI components are installed, the script is `installation_directory/nmgui_profile.sh`, for example, `/opt/IBM/netcool/nmgui_profile.sh`.

For example, on Bash, Bourne, and Korn shells, source the nmgui_profile.sh script using a command similar to the following:

```
. /opt/IBM/netcool/nmgui_profile.sh
```

What to do next

After you have set the environment variables, start Network Manager and make sure it is running correctly.

Related tasks

Starting Network Manager

You can start Network Manager using the methods outlined here. From Network Manager V4.2 and above, you must start Tivoli Netcool/OMNIbus separately, using the Tivoli Netcool/OMNIbus commands.

Starting Network Manager

You can start Network Manager using the methods outlined here. From Network Manager V4.2 and above, you must start Tivoli Netcool/OMNIbus separately, using the Tivoli Netcool/OMNIbus commands.

About this task

Important: Tivoli Netcool/OMNIbus and the topology database must both be started before Network Manager.

Related tasks

Setting environment variables

Before starting any components or working with any configuration files, set the appropriate environment variables by running the environment scripts.

Starting all Network Manager components (UNIX only)

You can start Network Manager and all of its component s using the **`itnm_start`** command. This command also can also be used to start the Apache Storm real-time computation system, which us used to aggregate raw poll data into historical poll data.

Before you begin

Before working with processes, set the appropriate environment variables by running the environment scripts.

About this task

In the case of Network Manager, the **`itnm_start`** command starts the master process controller, **`ncp_ctrl`**, which starts all the Network Manager processes.

To run the **`itnm_start`** command:

Procedure

1. If you have not set up the UNIX environment, change to the `$NCHOME/precision/bin` directory.
2. Type the following command: `itnm_start -domain` *`NCOMS`*.

   This command starts all of the Network Manager components that are installed on the server in the example domain NCOMS.

Related tasks

Setting environment variables

Before starting any components or working with any configuration files, set the appropriate environment variables by running the environment scripts.

Starting Network Manager processes using the command console

You can start Network Manager processes by starting the master process controller, **ncp_ctrl**. using the command console.

Before you begin

Before beginning this task, check the following:

- If you want different process dependencies to the defaults, ensure they are configured first.
- Ensure that the UNIX environment is set up.

About this task

If you start Network Manager processes using the master process controller, only processes that belong to the Network Manager core components are started.

To start Network Manager using the command console

Procedure

Type the following command:

```
ncp_ctrl -domain DOMAIN &
```

where *DOMAIN* is the domain in which you want to start the core components.

Stopping Network Manager

You can stop Network Manager using the methods outlined here. From Network Manager V4.2 and above, you must stop Tivoli Netcool/OMNIbus separately, using the Tivoli Netcool/OMNIbus commands.

Stopping all Network Manager components (UNIX only)

You can stop Network Manager and all of its components, using the **itnm_stop** command. This command also can also be used to stop the Apache Storm real-time computation system, which us used to aggregate raw poll data into historical poll data.

Before you begin

Before working with processes, set the appropriate environment variables by running the environment scripts.

About this task

On all supported operating systems, you can use the **itnm_stop** script to stop the Network Manager domain process controller, the **ncp_ctrl** process (which then stops all required processes).

To run the **itnm_stop** command, complete the following steps.

Procedure

1. Change to the `$NCHOME/precision/bin` directory.
2. Type the following command: `itnm_stop -domain NCOMS`

   This command stops all of the Network Manager components that are installed on the server in the example domain NCOMS.

Related tasks

Setting environment variables
Before starting any components or working with any configuration files, set the appropriate environment variables by running the environment scripts.

Stopping Network Manager processes using the command console

You can stop all the processes of Network Manager by stopping the master process controller, the `ncp_ctrl` process.

About this task

If you stop Network Manager processes using the master process controller, only processes that belong to the Network Manager core components are stopped.

To stop the `ncp_ctrl` process:

Procedure

1. Select the console window where the `ncp_ctrl` process is running.
2. Press Ctrl+C.

Results

The `ncp_ctrl` process stops, and also stops all its managed processes.

Restarting the Dashboard Application Services Hub server

After customization and configuration activities, you might need to restart the Dashboard Application Services Hub.

About this task

Fix Pack 7 You do not normally need to restart the Dashboard Application Services Hub server if you change a `.properties` file that belongs to Network Manager. Such changes are read automatically.

Restarting Dashboard Application Services Hub restarts all of the GUI applications that are running on your Dashboard Application Services Hub server.

The applications that are restarted include Network Manager GUI applications and Web GUI applications.

To restart the server:

Procedure

1. On the relevant Dashboard Application Services Hub, open a command window.
2. Change to the `$JazzSM_HOME/profile/bin` directory.
3. Stop the server by using the following command:

   ```
   stopServer.sh server1
   ```

   Note: You are prompted to supply the user name and password of the administrative user.
4. Wait a moment for the server to completely shut down and, confirm that all Java™ processes stopped running.

   The following messages confirm that the server is shut down:

   ```
   ADMU3201I: Server stop request issued. Waiting for stop status.
   ADMU4000I: Server server1 stop completed.
   ```
5. Start the server by using the following command:

```
startServer.sh server1
```

Chapter 2. Administering processes

You can start, stop, and investigate individual Network Manager processes.

About this task

About process control

You can check the status of Network Manager processes using the master process controller, the **ncp_ctrl** process.

By default, the **ncp_ctrl** process launches all the Network Manager processes in the appropriate order, in line with configured process dependencies. You can also use the **ncp_ctrl** process to start individual Network Manager processes.

The **ncp_ctrl** process is the only Network Manager component that can start another process. It is also used by other Network Manager processes that need to launch and manage their subprocesses.

The **ncp_ctrl** process is the master process and must be run before all other processes. The **ncp_ctrl** process launches and manages the appropriate processes when their dependencies have been satisfied.

Network Manager processes

Processes might be referred to in documentation by their executable name (which begins ncp_) or by a descriptive name.

The following table describes the Network Manager processes.

Table 1. Network Manager processes

Executable name	Descriptive name	Description
ncp_brokerd	Really Small Message Broker daemon	Message broker daemon that launches the Really Small Message Broker. Communication between Network Manager core components is managed by Really Small Message Broker. ncp_brokerd starts automatically when any Network Manager process starts.
ncp_class	Active Object Class manager, CLASS	Dynamic library management system responsible for managing the Active Object Classes (AOCs). It is the only component that has direct contact with the AOC definitions, and it distributes these definitions to any component that needs them. You can edit AOCs using a text editor. Restart the **ncp_class** process after changing AOC files. After ncp_class is restarted and running, restart the **ncp_model** process. **Note:** Ensure that you make a backup of any original AOCs before you edit them. If you overwrite the original copy, the backup copy can be restored.

© Copyright IBM Corp. 2006, 2021

Table 1. Network Manager processes (continued)

Executable name	Descriptive name	Description
`ncp_config`	Network Manager GUI configuration file server, CONFIG	Configuration file server that provides a means for Network Manager GUIs to read from and write to schema files.
`ncp_ctrl`	Master process controller, CTRL	Master process controller that launches all the Network Manager processes in the appropriate order, in line with configured process dependencies. You can also use the `ncp_ctrl` process to start individual Network Manager processes.
`ncp_crypt`	Password encryption utility	Utility for manual encryption of passwords.
`ncp_disco`	Discovery engine	Manages the process of discovering device existence and interconnectivity. Subprocesses of ncp_disco include the following finder processes, which are responsible for determining device existence: • ncp_df_ping (Ping finder): Makes a simple ICMP echo request for broadcast or multicast addresses, individual IP addresses, or all devices on a subnet. • ncp_df_file (File finder): Parses a file, such as `/etc/hosts`, to retrieve a list of devices to find devices on the network. • ncp_df_dbentry (Database finder): Reads a database in order to retrieve a list of devices to find on the network. • ncp_df_collector (Collector finder): Retrieves a list of devices managed by Element Management Systems (EMSs) on the network
`ncp_dla`	Discovery Library Adapter	Collects data on network resources and relationships from Network Manager for import into the Tivoli® Change and Configuration Management Database (CCMDB).

Table 1. Network Manager processes (continued)

Executable name	Descriptive name	Description
`ncp_d_helpserv`	Helper Server	Helpers retrieve information from the network during a discovery. The Helper Server manages the helpers and stores information retrieved from the network. Discovery agents retrieve their information through the Helper Server to reduce the load on the network. The Helper Server can service the requests directly with cached data or pass on the request to the appropriate helper. The Helper Server manages the following helpers: • ncp_dh_arp (ARP helper): performs IP address to MAC address resolution • ncp_dh_dns (DNS helper): performs IP address to device name resolution • ncp_dh_ping (Ping helper): either pings each device in a subnet, an individual IP address or a broadcast or multicast address • ncp_dh_snmp (SNMP helper): returns results of an SNMP request such as Get, GetNext and GetBulk • ncp_dh_telnet (Telnet helper): returns results of a Telnet operation into a specified device • ncp_dh_xmlrpc (Collector helper): provides communications facilities with EMS collectors using the XML-RPC interface
`ncp_g_event`	Event Gateway	Provides a bidirectional interface between Network Manager and Tivoli Netcool/OMNIbus. The Event Gateway also forwards events to plug-ins that subscribe to specific types of event and perform further action or further event enrichment based on event data.
`ncp_mib`	MIB update administration utility	Use the MIB update administration utility to update your MIB data for use within the SNMP MIB Browser.
`ncp_model`	Topology manager	Stores the topology data following a discovery and sends the topology data to the topology database (NCIM), where it can be queried using SQL. The topology visualization GUIs retrieve topology data from NCIM to display to network operators.
`nco_p_ncpmonitor`	Probe for Tivoli Netcool/ OMNIbus	Enables events generated by the Network Manager polling to be sent to the Tivoli Netcool/OMNIbus ObjectServer. The `nco_p_ncpmonitor` process converts these events into ObjectServer format.
`ncp_poller`	Polling engine	Controls network device polling.

Table 1. Network Manager processes (continued)		
Executable name	**Descriptive name**	**Description**
`ncp_oql`	OQL Service Provider	Command-line interface that enables administrators to query and update data in Network Manager databases.
`ncp_trapmux`	SNMP trap multiplexer	In most networks, traps arrive on a single default port. The SNMP trap multiplexer resolves this problem by listening to a single port and forwarding all the traps it receives to a set of host/socket pairs.
`ncp_virtualdomain`	Virtual Domain	Virtual Domain is used when running Network Manager with failover. Any connection to this virtual domain is routed to the Network Manager server that is running as the primary server in the failover architecture.
`ncp_webtool`	Webtools	Provides for the hosting of WebTools on the backend server so that they can be accessed in distributed environments where Topoviz is running on a different server to the Network Manager backend processes and there is a firewall between the two.

Managed and unmanaged processes

The ncp_ctrl process starts two types of process: managed and unmanaged processes.

Managed processes
Processes for which the ncp_ctrl process is fully responsible. The ncp_ctrl process not only starts and stops these processes, but also keeps track of their activities and restarts them if they die. The number of times that ncp_ctrl will restart a managed process is configurable using the retryCount field in the services.inTray database.

Unmanaged processes
Processes that the ncp_ctrl process is only responsible for starting or stopping. The ncp_ctrl process is not responsible for tracking independent unmanaged processes and makes no attempt to restart these processes if they die. Unmanaged processes can be divided into two types, as follows:

Independent unmanaged processes
An independent unmanaged process continues to run if other processes die.

Dependent unmanaged processes
Dependent unmanaged processes are stopped by ncp_ctrl if the parent process dies. An example of dependent unmanaged processes are the discovery agents started by the parent Discovery engine, ncp_disco, process.

The core Network Manager processes (that is, those responsible for discovery, monitoring, root cause analysis, and topology data) are handled as managed processes. Processes such as scripts should be launched as unmanaged processes.

About Network Manager domains

A domain is a part of the network that is discovered and managed separately. Many Network Manager processes run one instance per domain. Each domain has a unique name.

Running multiple domains enables you to discover, visualize and monitor multiple network topologies. Multiple Network Manager processes can run independently of each other on the same server if they belong to different domains.

Dividing your network into domains allows you to discover your network in sections. You might want to do this for reasons of scalability: your network might be too big to be discovered in one piece. Alternatively, you might want to break the network into geographical regions, and make each region correspond to a domain.

By default, Network Manager runs on a single domain.

The domain in which a component runs is determined by the command-line argument -domain, which is compulsory for all components, with the exception of the **ncp_mib** process, which manages the importing of MIBs across all domains using the same Netcool Common Inventory Model (NCIM) database.

Configuration files that are specific to a particular domain have the domain name appended to the file name. For example, the configuration file for the ncp_ctrl process running in domain NCOMS would be `CtrlServices.NCOMS.cfg`

Restriction: Use only alphanumeric characters and underscores (_) for domain names. Any other characters, for example hyphens (-), are not permitted.

.

Domain-specific configuration files

You can use different configuration settings for different domains by saved domain-specific versions of the configuration files that you edit.

If a domain-specific configuration file is found, those configuration settings are used by the relevant processes in that domain. If a domain-specific configuration file is not found, the settings from the non-specific configuration file are used.

To save a domain-specific version of a configuration file, add the domain name to the end of the file name immediately before the file extension. For example, the configuration file for the `ncp_ctrl` process in the domain NCOMS is called `CtrlServices.NCOMS.cfg`.

Although in practice there are some files that you are unlikely to need to alter, in principle all of the following types of files can be made domain-specific:

- Configuration files, that is, all files ending in `.cfg`
- Discovery agent files, that is, all files ending in `.agnt`
- Active Object Class files, that is, all files ending in `.aoc`
- Text-based stitcher files, that is, all files in a `stitchers` directory ending in `.stch`

In the Network Manager documentation, these files are referred to using their default names unless noted otherwise.

Checking process status

You can check the status of IBM Tivoli Netcool/OMNIbus, and individual Network Manager processes.

Checking process status by running the itnm_status command

On UNIX operating systems, you can check the status of Network Manager by using the **itnm_status** command.

Before you begin

Before working with processes, set the appropriate environment variables by running the environment scripts.

About this task

To check the status of all Network Manager components on the current server, complete the following steps:

Procedure

1. Change to the `$NCHOME/precision/bin` directory.
2. Type the following command: `itnm_status`

   This command displays the status of all of the Network Manager components that are installed on the server.

Related tasks

Setting environment variables
Before starting any components or working with any configuration files, set the appropriate environment variables by running the environment scripts.

Monitoring process status messages

You can view status messages from Network Manager to understand the health and status of the product.

About this task

The Network Manager processes send messages to IBM Tivoli Netcool/OMNIbus when they start and stop. You can view these messages to see which processes have started and stopped, and to see failover status.

For more information about Network Manager status events, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

To view process status messages, complete the following tasks.

Procedure

1. Add an **Event Viewer** widget to a dashboard.
2. Apply a filter to the **Event Viewer** so that only events with an `Alert Group` of `ITNM Status` are displayed.

Checking process status by querying ncp_ctrl databases

On all operating systems, you can check the status of individual Network Manager processes by querying the databases of the **ncp_ctrl** process.

About this task

The **ncp_ctrl** process must also be running for the domain that you want to interrogate.

Identifying which Network Manager processes are running

To identify the processes that were started by the **ncp_ctrl** process, have finished starting up, and are currently running, issue a query to the services.inTray database table.

About this task

To identify which processes are running:

Procedure

1. Log in to the Ctrl service using either the OQL Service Provider or the Management Database Access page:
   - Start the OQL Service Provider by typing a command similar to the following:

     ```
     ncp_oql -domain NCOMS -service Ctrl
     ```

     where *NCOMS* is the domain name. If authentication has been configured for the OQL Service Provider, enter your username and password.
   - Log in to the Management Database Access page and select the Ctrl service.
2. Issue the following command:

   ```
   select serviceName, binaryName, domainName, processId
   from services.inTray
   where serviceState = 4 ;
   go
   ```

   Note:

   It can take a few minutes for some processes to fully start up after ncp_ctrl has started them. While a process is starting up, it is not returned by this query.

Example

The following example output shows that 12 processes were started by the **ncp_ctrl** process and are currently running:

```
...........
{
        binaryName='ncp_store';
        domainName='SCO099';
        processId=12129;
        serviceName='ncp_store';
}
{
        binaryName='ncp_class';
        domainName='SCO099';
        processId=12130;
        serviceName='ncp_class';
}
{
        binaryName='ncp_model';
        domainName='SCO099';
        processId=12384;
        serviceName='ncp_model';
}
{
```

```
        binaryName='ncp_disco';
        domainName='SCO099';
        processId=12488;
        serviceName='ncp_disco';
}
{
        binaryName='ncp_d_helpserv';
        domainName='SCO099';
        processId=12131;
        serviceName='ncp_d_helpserv';
}
{
        binaryName='ncp_config';
        domainName='SCO099';
        processId=12132;
        serviceName='ncp_config';
}
{
        binaryName='ncp_poller';
        domainName='SCO099';
        processId=12906;
        serviceName='ncp_poller_default';
}
{
        binaryName='ncp_poller';
        domainName='SCO099';
        processId=12907;
        serviceName='ncp_poller_admin';
}
{
        binaryName='nco_p_ncpmonitor';
        domainName='SCO099';
        processId=12133;
        serviceName='nco_p_ncpmonitor';
}
{
        binaryName='ncp_g_event';
        domainName='SCO099';
        processId=12552;
        serviceName='ncp_g_event';
}
{
        binaryName='ncp_webtool';
        domainName='SCO099';
        processId=12134;
        serviceName='ncp_webtool';
}
{
        binaryName='ncp_virtualdomain';
        domainName='SCO099';
        processId=13182;
        serviceName='ncp_virtualdomain';
}
( 12 record(s) : Transaction complete )
```

Identifying which processes are started automatically

To identify which processes are started automatically by the **ncp_ctrl** process, issue a query to the services.inTray database table.

About this task

To identify which processes are started automatically:

Procedure

1. Log in to the Ctrl service using either the OQL Service Provider or the Management Database Access page:
   - Start the OQL Service Provider by typing a command similar to the following:

     ```
     ncp_oql -domain NCOMS -service Ctrl -username admin
     ```

     where *NCOMS* and *admin* are your domain name and username.

- Log in to the Management Database Access page and select the Ctrl service.

2. Issue the following command:

```
select * from services.inTray;
go
```

Example

The following example output shows processes configured to be started by the **ncp_ctrl** process:

```
{
        serviceName='ncp_disco';
        binaryName='ncp_disco';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin';
        domainName='NCOMS';
        argList=['-domain','$PRECISION_DOMAIN','-discoOnStartup',
                 '0','-latency','100000','-debug','0','-messagelevel',
                 'warn'];
        dependsOn=['ncp_d_helpserv','ncp_model'];
        retryCount=5;
        serviceId=4;
        traceLevel=0;
        logLevel='warn';
        serviceKey='ncp_disco_NCOMS';
        serviceState=4;
        interval=10;
        processId=2622;
}
.....
.....
{
        serviceName='ncp_model';
        binaryName='ncp_model';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin';
        domainName='NCOMS';
        argList=['-domain','$PRECISION_DOMAIN','-latency','100000',
                 '-debug','0','-messagelevel','warn'];
        dependsOn=['ncp_config','ncp_store','ncp_class'];
        retryCount=5;
        serviceId=3;
        traceLevel=0;
        logLevel='warn';
        serviceKey='ncp_model_NCOMS';
        serviceState=4;
        interval=10;
        processId=2542;
}
```

Identifying unmanaged processes

To identify which processes are started but not managed by the **ncp_ctrl** process, issue a query to the services.unManaged database.

About this task

Insertions into the `services.unManaged` table are made by other Network Manager components in order to start and stop their subprocesses; for example, the **ncp_disco** process uses the **ncp_ctrl** process to start the finders.

To identify unmanaged processes:

Procedure

1. Log in to the Ctrl service using either the OQL Service Provider or the OQL Workbench:
   - Start the OQL Service Provider by typing a command similar to the following:

     ```
     ncp_oql -domain DOMAIN_NAME -service Ctrl
     ```

   - Log in to the OQL Workbench and select the Ctrl service.

2. Issue one of the following command to list unmanaged processes:

   a) Issue the following command to list all unmanaged processes:

```
select * from services.unManaged;
go
```

   b) Issue the following command to list all only dependent unmanaged processes:

```
select * from services.unManaged
WHERE dependency is not NULL;
go
```

Example

The following example output shows unmanaged processes have been started by the **ncp_ctrl** process:

```
{
        serviceName='ncp_df_ping';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        dependency=19803;
        argList=['-domain','LNX39024','-server','ncp_disco.2622'];
        binaryName='ncp_df_ping';
        serviceId=14;
        logLevel='warn';
        traceLevel=0;
        domainName='NCOMS';
        processId=19869;
}
{
          serviceName='ncp_df_collector';
          servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
          dependency=19803;
          argList=['-domain','COLT45','-server','ncp_disco.19803'];
          binaryName='ncp_df_collector';
          serviceId=13;
          logLevel='warn';
          traceLevel=0;
          domainName='NCOMS';
          processId=19870;
}
{
          serviceName='ncp_df_file';
          servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
          dependency=19803;
          argList=['-domain','COLT45','-server','ncp_disco.19803'];
          binaryName='ncp_df_file';
          serviceId=14;
          logLevel='warn';
          traceLevel=0;
          domainName='NCOMS';
          processId=19871;
}
{
         serviceName='ncp_agent';
         servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
         dependency=19803;
          argList=['-domain','COLT45','-agent','Details','-server',
'ncp_disco.19803','-threads','100','-messagelevel','warn'];
           endSignal=2;
          binaryName='ncp_agent';
          serviceId=17;
          logLevel='warn';
          traceLevel=0;
          domainName='NCOMS';
           processId=28352;
}{
        serviceName='ncp_dh_snmp';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        dependency=19646;
        argList=['-domain','LNX39024'];
        binaryName='ncp_dh_snmp';
        serviceId=19;
        logLevel='warn';
        traceLevel=0;
        domainName='NCOMS';
        processId=28460;
```

```
}
( 5 record(s) : Transaction complete )
```

Managing process dependencies

The processes that are managed by ncp_ctrl in the default configuration cannot start until the processes on which they depend have fully started. Incorrectly-configured process dependencies can result in problems starting processes.

About this task

Listing process dependencies

You can issue a query to the services.inTray database to identify which processes have dependencies on other processes.

About this task

To identify process dependencies:

Procedure

1. Log into the process control databases.
2. Issue the following command:

```
select serviceName, dependsOn
from services.inTray;
go
```

3. The following example output shows that **ncp_class** and **ncp_store** have no dependencies, and that **ncp_model** is dependent on **ncp_class** and **ncp_store**:

```
.....
{
                serviceName='ncp_class';
                dependsOn=[];
}
{
                serviceName='ncp_store';
                dependsOn=[];
}
.....
.....
{
                serviceName='ncp_model';
                dependsOn=['ncp_class', 'ncp_store'];
}
( 4record(s) : Transaction complete )
```

Identifying dependencies for a particular process

To identify the dependencies for a particular process, issue a query to the services.inTray database.

About this task

To identify process dependencies for a particular process:

Procedure

1. Log into the process control databases.
2. Issue the following command:

```
select serviceName, dependsOn
from services.inTray
where serviceName='SERVICE';
go
```

Where *PROCESS* is the name of the process for which you want to query the dependencies; for example, `ncp_disco`.

Example

The following example output shows that **ncp_model** is dependent on **ncp_class** and **ncp_store:**

```
{
                serviceName='ncp_model';
                dependsOn=['ncp_class', 'ncp_store'];
}
( 1record(s) : Transaction complete )
```

Configuring process dependencies

To configure process dependencies, edit the `$NCHOME/etc/precision/CtrlServices.cfg` configuration file.

About this task

The process dependencies defined in the `CtrlServices.cfg` configuration file specify the order in which the **ncp_ctrl** process starts the processes.

Tip:

Process dependencies are configured by default and do not normally need to be changed.

To configure process dependencies:

Procedure

1. Back up and edit he `CtrlServices.`*DOMAIN*`.cfg` configuration file for your domain, where *DOMAIN* is the name of your domain.
2. Locate the entry for the process whose dependencies you want to configure by looking for the following line in the file:

   ```
   serviceName='process_name';
   ```

   where *process_name* is the name of the process.
3. Change the dependencies of the process by adding or removing process names to the following line, directly underneath the previous line:

   ```
   dependsOn=['process_name', 'process_name2'];
   ```
4. Save the **CtrlServices.cfg** configuration file.
5. Restart the master process controller, the **ncp_ctrl** process, for your changes to take effect.

List of process dependencies

The Network Manager processes that are managed by ncp_ctrl by default must be started in the correct order.

The process dependencies are shown in the following table:

Table 2. Dependencies of the Network Manager processes	
Process	**Dependencies**
ncp_class	No dependency
ncp_config	No dependency
ncp_ctrl	No dependency
ncp_disco	**ncp_d_helpserv**, **ncp_model**
ncp_d_helpserv	No dependency
ncp_g_event	**ncp_model**, Tivoli Netcool/OMNIbus ObjectServer
ncp_model	**ncp_config**, **ncp_class**, **ncp_store**
ncp_poller	**ncp_g_event**, **nco_p_ncpmonitor**
nco_p_ncpmonitor	Netcool/OMNIbus ObjectServer
ncp_store	No dependency
ncp_virtualdomain	**ncp_g_event**, **ncp_poller**
ncp_webtool	No dependency

Process control configuration files

Use the following configuration files to configure the ncp_ctrl process.

- `$NCHOME/etc/precision/CtrlSchema.cfg` contains the definitions for the databases of the ncp_ctrl process. You should not need to edit this file.
- `$NCHOME/etc/precision/CtrlServices.cfg` contains any necessary inserts into the databases of the ncp_ctrl process to tell the `ncp_ctrl` process which processes to start and in what order.

In order to configure the `ncp_ctrl` process to launch and manage the appropriate processes, you must append OQL inserts to the configuration file for the `ncp_ctrl` process, `$NCHOME/etc/precision/CtrlServices.cfg`.

Starting and stopping processes

You can start and stop individual processes manually or automatically.

About this task

Configuring managed processes

You can configure managed processes, which are started automatically by the **ncp_ctrl** process, by editing the $NCHOME/etc/precision/CtrlServices.cfg file and starting or restarting ncp_ctrl. You can choose which processes are started this way, and you can change the command line parameters.

About this task

Your changes take effect when the **ncp_ctrl** process is stopped and restarted. Your changes also persist if ncp_ctrl is later stopped and restarted. For this reason, starting managed processes using the $NCHOME/etc/precision/CtrlServices.cfg as described here is more robust than making inserts directly into the services.inTray database table.

You might only want to use a subset of the Network Manager functionality. For example, you might want to use Network Manager to discover your network and visualize the topology only. In this case you can configure the **ncp_ctrl** process so that it does not start the processes that monitor the network and perform root-cause analysis on network events.

To configure which processes are started automatically, perform the following steps:

Procedure

1. Back up your $NCHOME/etc/precision/CtrlServices.cfg file.
2. Save a copy of the CtrlServices.cfg file with your domain name included in the filename, for example, CtrlServices.NCOMS.cfg.
3. Edit the CtrlServices.MASTER.cfg file. For example, if you want to discover your network and visualize the topology only, you must delete or comment out the entries for **ncp_g_event**, **ncp_poller**, and **ncp_virtualdomain** processes from the CtrlServices.NCOMS.cfg file.
4. Make any changes to the command line parameters that you want to use to start the processes.
5. Start the **ncp_ctrl** process in the domain NCOMS.

Results

The **ncp_ctrl** process now starts the limited set of processes in the domain NCOMS in the order you specified.

Sample: Starting Network Manager with discovery and visualization functionality only

This sample shows you how to configure the master process controller to start only the processes that perform and support network discovery and visualization.

To ensure that monitoring and event management processes are not started on the current server, you must remove insert statements related to the ncp_g_event, ncp_poller and ncp_virtualdomain processes from the CtrlServices.DOMAIN.cfg file.

Note: The probe for Tivoli Netcool/OMNIbus, nco_p_ncpmonitor, should be left in the CtrlServices.DOMAIN.cfg file, as it will still be used to pass Network Manager status events to the ObjectServer.

For the ncp_g_event process, the lines to be removed look similar to the following lines:

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

```
(
    "ncp_g_event",
    "ncp_g_event",
    "$NCHOME/preecision/platform/$PLATFORM/bin",
    "DOMAIN",
    [ "-domain" , "DOMAIN", "-latency", "60000" , "-debug", "0", "-messagelevel",
 "warn" ],
    [ "ncp_model" ],
    5
);
```

Starting unmanaged processes

You can start a process as an unmanaged process by making an OQL insert into the `services.inTray` table.

About this task

Your changes will be lost if the **ncp_ctrl** process is stopped and restarted.

To start an unmanaged process:

Procedure

1. Ensure that the **ncp_ctrl** process is running.
2. Log into the process control databases.
3. Issue a command similar to the following:

```
insert into services.unmanaged
(
        serviceName, servicePath, argList
)
values
(
        "user_script",
        "/opt/netcool/precision/solaris2/scripts/",
        [ ]
);
```

Results

The above insert starts a script called user_script, located in the `$NCHOME/precision/scripts` directory.

Stopping managed processes

You can stop a managed process that is running by deleting the record in the services.inTray table.

About this task

If you stop a managed process in any other way than deleting its entry in the services.inTray table, ncp_ctrl restarts it. If you delete a record from the services.inTray table, the process is restarted only when the **ncp_ctrl** process is restarted.

To stop a managed process:

Procedure

1. Ensure that the **ncp_ctrl** process is running.
2. Log into the process control databases.
3. Issue a command similar to the following:

```
delete from services.inTray
where serviceName = 'ncp_model' ;
go
```

Note: Stopping the ncp_disco process in this way does not stop the discovery immediately if it is busy. Stopping the ncp_disco process forcibly can corrupt the discovery.

Running processes remotely

If you want Network Manager processes on one server to be managed by **ncp_ctrl** on another server, you must configure both instances of **ncp_ctrl**.

Procedure

1. Install Network Manager on both servers.
2. Configure Really Small Message Broker to allow communication between the master server and slave server.
3. On the master server, configure the `CtrlServices.`*`DOMAIN`*`.cfg` file.
   a) Back up and edit the `CtrlServices.`*`DOMAIN`*`.cfg` file.
   b) For each process that you want to run on the remote server, set the **hostName** parameter to the host name of the remote server. Make sure the host name is the name as defined on the remote server.

   The following example configures the **ncp_store** process to run on the remote server called example.com in domain TARA.

```
insert into services.inTray
(
    serviceName,
    binaryName,
    servicePath,
    domainName,
    hostName,
    argList,
    retryCount
)
values
(
    "ncp_store",
    "ncp_store",
    "/opt/IBM/netcool/core/precision/platform/linux2x86/bin",
    "TARA",
    "example.com",
    [ "-domain" , "<DOMAIN>" , "-latency" , "100000", "-debug", "0" ],
    5
);
```

4. On the remote server, ensure that the `CtrlServices.`*`DOMAIN`*`.cfg` file is empty of content. Then, start the **ncp_ctrl** process on the remote server in slave mode.
   The following example starts the **ncp_ctrl** process in slave mode in the TARA domain .

```
ncp_ctrl -domain TARA -slave
```

5. Start the **ncp_ctrl** process on the local server in master mode using the normal command line options. The following example starts the ncp_ctrl process in master mode domain TARA.

```
ncp_ctrl -domain TARA
```

Results

The processes that you configured to be run on the slave server are started, and controlled by the **ncp_ctrl** process on the master server. The **ncp_ctrl** process on the master server also starts and controls any processes that it is configured to manage on the master server.

Chapter 3. Administering logs

Network Manager provides logging capabilities for its GUI components and back-end processes. You can set up logging for Network Manager to produce log or trace files that can be used for troubleshooting purposes.

Related tasks

Troubleshooting Network Manager

Consult these troubleshooting notes to help determine the cause of the problem and what to do about it.

Setting up logging for GUI

You can set up Network Manager to create log or trace files that can be used for troubleshooting GUI issues. You can also adjust the logging level for each component, and the maximum size and number of the log files the systems saves.

GUI component log file overview

Log messages produced by Network Manager GUI components are written to log and trace files.

- Log files provide log information in a standard format that is compatible with IBM®'s Common Base Event (CBE) format. Messages in CBE format can be used in the IBM Support Assistant Log Analyzer for offline analysis.

  Note: The IBM Support Assistant Log Analyzer is not shipped as part of IBM Tivoli Network Manager IP Edition. You must download and install it separately.
- Trace files capture all messages a log file contains and also additional technical details of operation. Trace files are intended to aid enhanced problem resolution, and are useful to provide to your IBM support contact when requested.

Log message format

GUI log messages are recorded in text format as follows:

```
[<date>T<time>]:<severity>:<message_code_id>:[<thread_id>]:<message>
```

For example:

```
[2010-09-02T04:50:57]:INFO:HNMOB0001I:[Deferrable Alarm : 0]:Initialising
Discovery GUI Server
```

Date and time

The date and time is in ISO 8601 format.

Severity

The following severity levels are available:

- `CONFIG`:

  Logs all events up to and including configuration changes.
- `INFO`:

  Logs only system state changes. This is the default setting.
- `WARNING`:

  Logs recoverable system errors.
- `SEVERE`:

  Logs unrecoverable system errors.

© Copyright IBM Corp. 2006, 2021

Message code ID

The message code provides more information on what component of the system the message originates from.

Table 3. Message code IDs

Message code ID	GUI component
HNM T *letter*	Topology visualization components:
HNM T A	Topology client
HNM T B	Topology server
HNM T C	Topology common
HNM N *letter*	MIB GUI components:
HNM N A	MIB browser
HNM N B	MIB grapher
HNM O *letter*	Discovery GUI components:
HNM O A	Discovery configuration GUI
HNM O B	Management Database (formerly called OQL Workbench)
HNM P *letter*	Network polling GUI components:
HNM P A	Network polling configuration (poll policies and definitions)
HNM S *letter*	Structure view components:
HNM S A	Structure browser
HNM X *letter*	Common GUI components:
HNM X A	OQL interface
HNM X B	Others, including: Tools, Filter Builder, Widgets, Entity Search, Expressions, Tree table.
HNM Z *letter*	External product interfaces:
HNM Z A	Tivoli Netcool/OMNIbus Web GUI

Thread ID

The thread ID indicates the task associated with the function the message originates from.

Message

The log message itself that provides a description of the event being logged.

Trace message format

Trace messages provide more granular details about operation in the following format:

```
<date> <component_id>\n
<severity>: <message>
```

For example:

```
Aug 24, 2010 3:34:30 AM com.micromuse.precision.disco.server.DiscoConfigLogger
FINE: Received unknown request from the network
```

Trace logs do not provide standardized message format as they are for more enhanced troubleshooting purposes. The severity levels available for trace messages are as follows:

- `FINE`:

  Minimum level of tracing. The majority of stack traces appear at this level already and are written to the trace file. The trace file also includes all log messages.
- `FINER`:

  Medium level of tracing that provides more detailed debug messages.
- `FINEST`:

  Maximum level of tracing that produces very detailed technical information.

Related tasks

Changing the logging level for GUIs
You can set the level of detail log files contain for GUI components as whole, or specify logging levels on a more granular basis for specific GUI application segments.

Locating GUI log files

All log files generated for GUI components are saved to the `$NMGUI_HOME/profile/logs/tnm/` directory.

About this task

The default name of the log or trace file is `ncp_`*`component_name`*`.`*`number`*`.log` or `ncp_`*`component_name`*`.`*`number`*`.trace`, respectively.

To locate a log file for a component:

Procedure

1. Go to `$NMGUI_HOME/profile/logs/tnm/`.
2. Locate the log and trace files that correspond to the GUI component you want to check the log messages for and open the file.

Table 4. GUI component log file mapping

GUI component	Logging properties set in file	.log and .trace file name
Discovery Status GUI	`discoconfig.properties`	`ncp_disco.0.log` `ncp_disco.0.trace`
Discovery Configuration GUI	`discoconfig.properties`	`ncp_guiconfig.0.log` `ncp_guiconfig.0.trace`
Management Database Access (formerly called OQL Workbench)	`nmdb.properties`	`ncp_nmdb.0.log` `ncp_nmdb.0.trace`
Network Polling GUI (poll policies and definitions)	`monitorconfig.properties`	`ncp_monitor.0.log` `ncp_monitor.0.trace`

Table 4. GUI component log file mapping (continued)		
GUI component	**Logging properties set in file**	**.log and .trace file name**
SNMP MIB Grapher **SNMP MIB Browser**	`itnmgraph.properties`	`ncp_mib.0.log` `ncp_mib.0.trace`
Structure Browser	`structurebrowser.properties`	`ncp_structureview.0.log` `ncp_structureview.0.trace`
Topology Visualization GUIs: • **Network Views** • **Network Hop View** • **Path Views GUI** • **Path View Administration GUI**	`topoviz.properties`	`ncp_topoviz.0.log` `ncp_topoviz.0.trace`
General settings including database properties for GUI components	`tnm.properties` **Note:** This is not to be confused with the log file of the same name located at `$NMGUI_HOME/precision/platform/java/lib/ncp_topoviz/etc/tnm/tnm.properties`. The latter file is used by the Polling engine, `ncp_poller`, to trigger updates to network views, so that poll policy scope is kept up-to-date.	`ncp_guiconfig.0.log` `ncp_guiconfig.0.trace`

Note: Once the log file reaches the specified maximum size limit, it is renamed and a new file is created. The first log file is named `ncp_`*`component_name`*`.0.log`, and the most recent log messages are always in this file. Previous log files are saved with the number increased (for example `ncp_nmdb.1.log`, `ncp_nmdb.2.log`, and so on).

Changing the logging level for GUIs

You can set the level of detail log files contain for GUI components as whole, or specify logging levels on a more granular basis for specific GUI application segments.

Setting logging level for GUI components

You can set the amount of information log files capture for each GUI component. The changes can be made before system startup or during operation. The changes are persistent, and are not affected by system restarts.

About this task

To set the logging behavior, you need to modify the corresponding configuration file.

Procedure

1. Go to `$NMGUI_HOME/profile/etc/tnm/`.
2. Open the `.properties` file of the GUI component you want to set the logging level for:

Option	Description
discoconfig.properties	Discovery configuration GUI
itnmgraph.properties	MIB graphing MIB Browser
monitorconfig.properties	Network polling configuration (poll policies and definitions)
nmdb.properties	Management database (formerly called OQL Workbench)
nm_rest.properties	ReST API and SQL queries
structurebrowser.properties	Structure browser
tnm.properties	General settings including database properties for GUI components
topoviz.properties	Topology visualization GUI

3. Edit the line *name*`.log.level` to set the message level:

Option	Description
CONFIG	Logs all events up to and including configuration changes.
INFO	Logs only system state changes. This is the default setting.
WARNING	Logs recoverable system errors.
SEVERE	Logs unrecoverable system errors.
FINE	Minimum level of tracing. The majority of stack traces appear at this level already and are written to the trace file. The trace file also includes all log messages. **Note:** When setting the logging level to FINE, FINER, FINEST, or ALL, both log files and trace files will contain information, and the trace files will include all messages from the log files in addition to the more technical details of operation. If any other logging level is set, the trace files remain empty.
FINER	Medium level of tracing that provides more detailed debug messages.
FINEST	Maximum level of tracing that produces very detailed technical information.
ALL	Enables logging and tracing on all levels for the application.
OFF	Disables all logging and tracing for the application.

4. Save and close the `.properties` file.

   Note: The changes take effect immediately if they are made before starting Network Manager. If the changes are made when the system is already running, Network Manager reads the configuration files every 60 seconds and applies any changes immediately.

Example

The following example shows the section of the `structurebrowser.properties` file that determines logging level:

```
structurebrowser.log.filename=ncp_structureview.%g.log
structurebrowser.log.level=INFO
structurebrowser.log.maxsize=10
structurebrowser.log.count=1

structurebrowser.trace.filename=ncp_structureview.%g.trace
structurebrowser.trace.maxsize=10
structurebrowser.trace.count=1
```

The settings here show the default **INFO** setting for the log files. This means log files are populated with information about system state changes, and trace files remain empty.

To change the logging level to have all log messages and enable trace messages, change **INFO** to at least **FINE** (or FINER, or FINEST, depending on the level of detail you require in the trace files). This will mean both log files and trace files will contain information. The following example reflects this change:

```
structurebrowser.log.filename=ncp_structureview.%g.log
structurebrowser.log.level=FINE
structurebrowser.log.maxsize=10
structurebrowser.log.count=1

structurebrowser.trace.filename=ncp_structureview.%g.trace
structurebrowser.trace.maxsize=10
structurebrowser.trace.count=1
```

Setting logging level for application segments

When a specific area requires enhanced troubleshooting, you can enable logging for GUI application segments.

Before you begin

Contact IBM support to identify which application segments require logging to be set for problem determination.

Note: These changes are not persistent. If the system is restarted, all log settings for specific GUI application segments are removed. Logging levels set for the whole GUI component are not affected.

Procedure

1. In the navigation pane, click **Settings** > **Websphere Administrative Console**.
2. Click **Launch WebSphere administrative console** to start the WebSphere® Application Server console.
3. In the administrative console, click **Troubleshooting** > **Logs and Trace**.
4. In the list, click the name of the server Network Manager is running on.
5. Click **Change Log Detail Levels** and then the **Runtime** tab.
6. Locate the specific application segment name by scrolling down the list and expanding any item as necessary.
7. Click the segment name and select the required logging level from the drop-down menu. Logging and trace level options are the same as for the GUI components.

   Note: By default, whatever is set in the GUI component `.properties` file is the default log and trace level for all relevant segments of that GUI component.
8. View the corresponding GUI component log file to check the messages logged for the segment. For example view the `ncp_disco.0.log` or `ncp_disco.0.trace` files for discovery GUI segments.

Related tasks

Locating GUI log files

All log files generated for GUI components are saved to the `$NMGUI_HOME/profile/logs/tnm/` directory.

Setting the log file size

You can set how large a log file can grow in MB, and determine the number of log files the systems keeps.

About this task

Follow these steps to set the maximum size of your log files in MB. After the file reaches the maximum size, it is renamed and a new file is created. You can also set the number of files to be stored after the size limit is reached.

Procedure

1. Go to `$NMGUI_HOME/profile/etc/tnm/` and open the `.properties` file of the GUI component you want to set the log size for.
2. In the properties file, perform the following steps:
   a) Locate the *component_name*`.log.maxsize` line and set the maximum size a log file can reach in MB. For example, **`nmdb.log.maxsize=20`** means the maximum allowed size of the Management Database log file is 20 MB. The default setting is 10 MB.
   b) Locate the *component_name*`.log.count` line and set the maximum number of files to be stored. For example, **`nmdb.log.count=2`** means the 2 latest log files will be kept apart from the one being written to at the moment. The default setting is 1, meaning the current and 1 previous file are saved only.

   Note: Once the log file reaches the specified maximum size limit, it is renamed and a new file is created. The first log file is named `ncp_`*`component_name`*`.0.log`, and the most recent log messages are always in this file. Previous log files are saved with the number increased (for example `ncp_nmdb.1.log`, `ncp_nmdb.2.log`, and so on).
3. Perform the same steps for the trace files by locating and editing the *component_name*`.trace.maxsize` and *component_name*`.trace.count` lines.
4. Save the `.properties` file.

Setting up logging for processes

You can troubleshoot processes by looking in the log files for information. You can set up Network Manager to record log or trace files for processes. You can also set the debug level for processes.

About this task

Process log file overview

Network Manager can create log and trace files for its processes.

Log files provide information about important process events, such as changes in state, warnings, or errors, in a standard format that is compatible with IBM's Common Base Event (CBE) format. Log files help administrators to monitor their systems and are useful to provide to your IBM Support contact when requested.

Trace files capture low-level system output and technical details. They are intended to aid enhanced problem resolution, and are useful to provide to your IBM Support contact when requested.

Log files can be identified by their .log suffix, and have the following characteristics:

- Log messages have timestamps.
- Log messages are graded by level, such as error, warn, info, and debug.
- Log messages are formatted to be compatible with IBM's Common Base Event format.

- Log files can be deleted and recreated to enable log file rotation.

Trace files can be identified by their .trace suffix. They can capture different levels of detail, referred to as debug levels. Debug level 4 is the most verbose. Trace files set to the higher debug levels can quickly consume disk space, and therefore should be used only when very detailed information is required to solve a problem.

Locating log files for a process

Locate log files for a process to obtain information that might be helpful in troubleshooting the process.

About this task

The default name of the log file is the process name followed by the domain name and then the `.log` or `.trace` file extension.

To locate a log file for a process:

Procedure

1. Navigate to the default location for process log and trace files, `$NCHOME/log/precision`.
2. Locate the log and trace files that correspond to the process name.
   For example, an instance of the **`ncp_disco`** process running on the NCOMS domain generates the following files:

   ncp_disco.*DOMAIN*.log
   ncp_disco.*DOMAIN*.trace
3. For log files for processes other than those that begin with `ncp_`, for example, for databases or third-party components, check the `$NCHOME/PD/core/`*`component_name`* directories.

Changing the logging level for processes

Change the logging level of a process before you start the process or while the process is running.

Changing the logging level before starting a process

Change the value of the relevant command-line argument in the configuration file to change the logging level that a process will use when it is started or restarted.

About this task

The –debug and –logdir command-line arguments are used for trace information and the -messagelevel and -messagelog command-line arguments are used for log information.

The default message level is `warn`, which means by default the log files do not contain `info` or debug messages.

To change the logging level:

Procedure

1. Navigate to the `CtrlServices.cfg` file. The file is located in the following directory:

   ```
   NCHOME/etc/precision/CtrlServices.domain_name.cfg
   ```

   The *domain_name* is the name of the domain for which the logging level is to be changed.
2. From the `CtrlServices.cfg` file, change the argument specified in the file to –debug for trace or –messagelevel for logging.

The following example shows how the **ncp_webtool** process might be configured in this file.

```
insert into services.inTray
(
    serviceName,
    binaryName,
    servicePath,
    domainName,
    argList,
    retryCount
)
values
(
    "ncp_webtool",
    "ncp_webtool",
    "$PRECISION_HOME/platform/$PLATFORM/bin",
    "$PRECISION_DOMAIN",
    [ "-domain" , "$PRECISION_DOMAIN" , "-latency" , "100000", "-debug", "0",
 "-messagelevel", "warn"],
    5
);
```

3. Start, or restart, the **ncp_ctrl** process.

   The **ncp_ctrl** process is used to stop and start all other processes. You can also restart Network Manager using the `itnm_start ncp` command.

Changing the logging level for a running process

Change the logging level of a running process to provide more detailed log or trace files to aid debugging.

About this task

You can change the logging level or trace level (also called the debug level) of a running process by sending a USR1 or USR2 signal to the process. Sending a USR1 signal changes the logging level and sending a USR2 signal changes the trace level. The extra information provided by increasing the logging or trace levels can help you when debugging a problem with a process.

Trace files have five debug levels (0 to 4) that you can cycle through to provide increasing levels of detail about a process. For example, if a process is already at level 3, a USR2 signal will increase the level to 4; if the process is at level 4, a USR2 signal will move it to level 0.

Log files have four message levels that you can cycle through to increase the level of detail that is captured: error, warn, info, and debug.

The following procedure describes how to increase the trace level of a process. To increase the logging level of a process, perform the same procedure using the USR1 signal instead of USR2.

To increase the trace level of a process, perform the following procedure:

Procedure

1. Find the Process ID (*PID*) of the process you are investigating:
   a) On Unix and Linux® operating systems, enter `ps -ef | grep ncp` at the command line.
2. To increase the debug level by one level:
   a) On Unix and Linux operating systems, enter `kill -USR2` *PID* at the command line.

Avoiding process errors with large trace or log files by using log file rotation

If trace or log files become too large, processes might quit unexpectedly. If too many files are created, you might run out of disk space. Configure the number and size of trace and log files by configuring log file rotation.

You can control the size of log and trace files using one of the following criteria:

- Maximum size

- Time of day

You can configure the number of log files by configuring a pool of files, that is, a number of files that are written to in succession. The process of using new files when triggered is called log file rotation. All log files can be rotated, but not all components have log files. Some components have only trace files, which can also be rotated.

Restriction: Only processes that are managed by the domain process controller, **ncp_ctrl**, can have their log files rotated.

Setting the maximum size of log and trace files

Large log or trace files can cause process errors. The default maximum size for a log file is 1 GB on UNIX systems. As a guidance estimate for log files, assuming that each log file is 1 GB in size and six processes are set to full debug level, you would require 24 GB of disk space. (6 processes x 4 log or trace files each = 24 log or trace files x 1 GB = 24 GB).

Set the NDE_LOGFILE_MAXSIZE environment variable to determine the maximum size that a log or trace file can reach for a process. This setting applies to all Network Manager processes. When the log file reaches the maximum size, the next log file is used.

- If log file rotation is turned off for a process, only two files are used.
- If log file rotation is turned on for a process, the number of files configured in the pool are used.

You can either rotate log files according to file size or time of day. If log file rotation according to time of day is configured, the settings for file size are ignored.

The following example shows how to set the maximum log file size to 1 GB (the setting is in bytes).

```
setenv NDE_LOGFILE_MAXSIZE 1073741824
```

Note: Do not set a maximum size of less than 1000000 (1MB). Setting a maximum size of less than 1000000 (1MB) can cause the ncp_g_event and nco_p_monitor processes to fail.

You must restart Network Manager processes after you change the log file rotation environment variables so that the processes use the updated variables.

Important: On UNIX systems, make sure that you set the logging environment variables in the appropriate shell profile files for the account that Network Manager is running. Do not set them in the `NCHOME/env.sh` file because this file is not used when Network Manager starts.

Setting a time of day for log file rotation

To set log file rotation to occur at a certain time of day, set the NDE_LOGFILE_ROTATION_TIME environment variable to the time required, and set the NDE_LOGFILE_ROTATION_FORMAT environment variable to the required naming format. If you set a time of day, this setting takes precedence over the maximum file size setting.

The NDE_LOGFILE_ROTATION_TIME environment variable specifies the time after which a log file rotation occurs each day. Log files are rotated when the first write is made to the file after the time that is set by the NDE_LOGFILE_ROTATION_TIME environment variable. This environment variable is an integer, with a default of 0000.

The following example specifies log file rotation at the first write after midnight each day:

```
setenv NDE_LOGFILE_ROTATION_FORMAT "%Y%m%d-%H%M"
setenv NDE_LOGFILE_ROTATION_TIME 0000
```

The following example shows how to rotate the log files at midnight each day, and append the old log file name with the literal character string `"old"`

```
setenv NDE_LOGFILE_ROTATION_FORMAT \'old\'
setenv NDE_LOGFILE_ROTATION_TIME 0000
```

Restriction: Literal characters must be escaped by quotation marks (' ') as described in http://userguide.icu-project.org/formatparse/datetime.

The NDE_LOGFILE_ROTATION_FORMAT environment variable configures how the log files are created.

- If NDE_LOGFILE_ROTATION_FORMAT is set to "old", and a pool of log files is not configured for the process in question, two files are used.
- If a pool of log files is configured for a process, the number of files in the pool is used.
- If NDE_LOGFILE_ROTATION_FORMAT is set to a supported time stamp format, and a pool of log files is not configured for the process in question, a new file is created each day. Using a time stamp format without configuring a pool of log files can result in an unlimited number of log files.

  Restriction: The only time stamp format that is supported for use by Network Manager processes is the POSIX format. The following options are supported:

  %a
  Abbreviated weekday name, for example, Thu

  %A
  Full weekday name, for example, Thursday

  %b
  Abbreviated month name, for example, Aug

  %B
  Full month name, for example, August

  %d
  Day of the month, zero-padded, (01-31), for example, 23

  %e
  Day of the month, space-padded, (1-31), for example, 23

  %h
  Equivalent to %b.

  %H
  Hour in 24h format (00-23), for example, 14

  %I
  Hour in 12h format (01-12), for example, 02

  %j
  Day of the year (001-366), for example, 235

  %m
  Month as a decimal number (01-12), for example, 08

  %M
  Minute (00-59), for example, 55

  %p
  AM or PM designation, for example, PM

  %R
  24-hour HH:MM time, equivalent to %H:%M, for example, 14:55

  %S
  Second (00-61), for example, 02

  %T
  ISO 8601 time format (HH:MM:SS), equivalent to %H:%M:%S, for example, 14:55:02

  %u
  ISO 8601 weekday as number with Monday as 1 (1-7), for example, 4

  %V
  ISO 8601 week number (00-53), for example, 34

  %y
  Year, last two digits (00-99), for example, 01

%Y
Year, for example, 2001

%Z
Timezone name.

For example, %Y%m%d-%H%M generates rotate log files with the year, month, day, hour, and minute added. An example file is ncp_model.NCOMS.log_20100430-0000.

You must restart Network Manager processes after you change the log file rotation environment variables so that the processes use the updated variables.

Important: On UNIX systems, make sure that you set the logging environment variables in the appropriate shell profile files for the account that Network Manager is running. Do not set them in the NCHOME/env.sh file because this file is not used when Network Manager starts.

Configuring the number of log files created per process

The number of log files created by a process is determined on a per-process basis by the command-line variables -logfileusepool and -logfilepoolsize. If a Network Manager process is started with the -logfileusepool command line option, then the process uses a pool of log files. The number of files in the pool is defined by the value of the -logfilepoolsize command line option. If the -logfilepoolsize command line option is not specified, the default value of 10 is used. The minimum value of -logfilepoolsize is 2 and the maximum is 99.

The log files in the pool are deleted each time the process is restarted by the user or by the system. If you want to keep historical logs you need to use the NDE_LOGFILE_ROTATION_FORMAT without a log pool.

Important: If you have many processes that are configured to create a pool of their own log and trace files, the total number of log files created can be large. Ensure you have enough disk space. Ensure that your operating system allows enough files to be opened concurrently. On UNIX and Linux operating systems, ensure that there are enough file handles allowed.

Restriction:

Do not start the ncp_g_event or nco_p_ncpmonitor processes with the -logfileusepool command line option if you have set the NDE_LOGFILE_ROTATION_FORMAT environment variable. Using log file pooling and a timed log file rotation together for these processes is not supported.

Example of starting a process with log file pool specified

The following example excerpt from the CtrlServices.cfg configuration file configures the **ncp_disco** process to use a pool of 5 log files:

```
insert into services.inTray
    (
        serviceName,
        binaryName,
        servicePath,
        domainName,
        argList,
        dependsOn,
        retryCount
    )
    values
    (
        "ncp_model",
        "ncp_model",
        "$PRECISION_HOME/platform/$PLATFORM/bin",
        "$PRECISION_DOMAIN",
        [
            "-domain" , "$PRECISION_DOMAIN", "-latency" ,
            "100000" , "-debug", "1", "-messagelevel", "debug", "-logfileusepool",
            "-logfilepoolsize", "5"
        ],
        [ "ncp_disco" ],
        1
    );
```

What happens when log files are rotated

If a pool of log files is configured for a process, log file rotation occurs as follows:

1. A number of log files are created for that process when it starts. The number of files that are opened is configured by the value of the `-logfilepoolsize` command line option. The process keeps the log files open for as long the process runs. By contrast, `.trace` files are only created when they are needed, and are not kept open. Files in the pool are named using the format *process*`[.`*domain*`].log_ID` or *process*`[.`*domain*`].trace_ID`, where `ID` is a two-digit number that starts from 01, and ranges up to the maximum number specified for the `-logfilepoolsize` option.
2. If they already exist, the first N .log files are truncated (set to zero size), and the first N `.trace` files are deleted, where N is the number of log files specified for the `-logfilepoolsize` option for the process. If there are more than N .log or .trace files, for example from a previous time when the log pool size was greater, the remaining files are not affected. If you want to be sure to keep all previous .log or `.trace` files, copy them to a different location before starting or restarting the process.

   Note: An exception is the ncp_disco process. When the ncp_disco process is restarted (including on every full discovery) the old `ncp_disco.`*DOMAIN*`.log_`n files are renamed with an extension of `.old` appended to the file name
3. The file with the lowest available file number is used for logging, regardless of which file was last used.
4. When the criteria for log rotation are met, the logging system writes to the next file.
5. When all the files in the pool are at maximum size, the logging system truncates the first file in the pool and starts writing to it again.

If a pool of log files is not configured, log file rotation occurs as follows:

1. The ncp_ctrl process renames the log file from *logfilename.log* to the format set by the NDE_LOGFILE_ROTATION_FORMAT environment variable, or to *logfilename.log_old* if the NDE_LOGFILE_ROTATION_FORMAT environment variable is not set.
2. The ncp_ctrl process generates a new log file named *logfilename.log*.
3. When the new *logfilename.log* file reaches the maximum size, the ncp_ctrl process overwrites *logfilename.log_old*.

Chapter 4. Administering ports

If there are conflicts with ports already in use on your system, change some of the default ports.

About this task

If you are deploying Network Manager in a secure environment, you might need to know which ports are used by different processes in order to configure a firewall or other security application.

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*.

About inter-process communication

Network Manager processes communicate using TCP connections, multicast, and Really Small Message Broker.

About Really Small Message Broker

Communication between Network Manager core components is managed by Really Small Message Broker. To ensure the correct operation of Network Manager, Really Small Message Broker must be running at all times.

Various Network Manager core components pass information to other components on the same server, and to any Network Manager core components on different servers.

This communication is managed by Really Small Message Broker.

Really Small Message Broker is installed and started automatically by the Network Manager installation process.

If you stop Really Small Message Broker while core Network Manager processes such as the Discovery engine, ncp_disco, are running, then the core processes will restart Really Small Message Broker automatically.

Note: Running multiple domains in parallel can overload the message broker on some systems. If you want to run multiple domains under load in parallel then it is good practice to run a separate message broker for each domain.

Related tasks

Running a separate message broker for each domain

© Copyright IBM Corp. 2006, 2021

If you want to run multiple domains under load in parallel then it is good practice to run a separate message broker for each domain.

About multicast

Processes that use direct TCP communication first use multicast to locate each other, then set up TCP sockets.

Changing host and port settings for Really Small Message Broker

You can change the host and port settings for Really Small Message Broker by modifying the Really Small Message Broker configuration file and then stopping the broker.

Updating the Really Small Message Broker configuration file

You can configure the host and port for Really Small Message Broker.

Before you begin

Before updating the Really Small Message Broker configuration file (`precision.broker.cfg`), you must stop all ncp processes.

About this task

To configure the host and port for Really Small Message Broker, complete the following steps on each server where Network Manager core components are installed.

Procedure

1. Ensure that all ncp processes have been stopped.
2. Delete the following file:

   ```
   $NCHOME/etc/precision/broker_1883.cfg
   ```

   Important: `Broker_1883.cfg` is automatically generated from `precision.broker.cfg` when the Really Small Message Broker starts. If this file is not deleted before the Really Small Message Broker file is edited, a mismatch can occur between two versions of the file. This can prevent Network Manager from starting.
3. Edit the following file:

   ```
   $NCHOME/etc/precision/Precision.broker.cfg
   ```

4. Locate the following section in the file:

   ```
   broker session =
     {
       'service' = '1883',
       'network' = '127.0.0.1'
     };
   ```

   Note: The `broker session` settings use the IP address of the loopback interface. This ensures that you can only access the broker from the local server. If you want to allow external connections then you must bind to the IP address of the server. Note that allowing external connections to the broker might constitute a security risk.
5. Change the value of `'service'` to the port you want to use. Ensure that it does not conflict with any other ports on your system.
6. Change the value of `'network'` to the address of the current server.
7. Save and close the file.

Stopping Really Small Message Broker

Once you have changed the Really Small Message Broker configuration file, you must stop Really Small Message Broker for your changes to take effect.

About this task

To stop Really Small Message Broker, run the following script.

```
$NCHOME/precision/scripts/perl/scripts/stop_broker.pl
```

Any currently running Network Manager core processes such as the Discovery engine, ncp_disco will restart Really Small Message Broker automatically. The new instance of Really Small Message Broker will pick up the configuration changes.

Running a separate message broker for each domain

If you want to run multiple domains under load in parallel then it is good practice to run a separate message broker for each domain.

About this task

To run a separate message broker for each domain, proceed as follows.

Procedure

1. Ensure that all ncp processes have been stopped.
2. Create a domain specific `Precision.broker.cfg` configuration file.

   Do this by copying the following file: `$NCHOME/etc/precision/Precision.broker.cfg` to a domain-specific copy: `$NCHOME/etc/precision/Precision.broker.`*`DOMAIN_NAME`*`.cfg`

   Where *DOMAIN_NAME* is the name of one of your domains.
3. Locate the following section in the file:

   ```
   broker session =
     {
       'service' = '1883',
       'network' = '127.0.0.1'
     };
   ```
4. Change the value of `'service'` to the port you want to use. Ensure that it does not conflict with any other ports on your system.
5. Save and close the file.
6. Repeat steps “2” on page 41 to “5” on page 41 to create a separate message broker for each of your domains.
7. Restart all ncp processes.

Checking port usage

You can check which ports are in use on the current server, to help investigate or prevent port conflicts.

About this task

To check which ports are in use on the current server, enter the following command:

```
netstat -a
```

The command returns a list of listening daemons and established connections.

Defining a fixed TCP port

For processes that use TCP socket-based connections, you can define a fixed port instead of using the default randomly assigned port.

About this task

To avoid firewall issues or port conflicts, you might have to define a specific TCP port for a process. For example, you might need to do this if the helpers and the Helper Server, `ncp_d_helpserv`, are running on a different host to the Discovery engine, `ncp_disco`, and these hosts are behind a firewall. You might also need to define a fixed TCP port as part of failover configuration.For more information on how to define a fixed TCP port specifically for failover, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

To define a fixed TCP port for a process, complete the following steps:

Procedure

1. On the first server, start the process.
2. Make a backup copy of the `ServiceData.cfg` file.
3. Edit the `ServiceData.cfg` file and copy the line relevant to the process for which you want to define a port.

   The existing line might resemble the following example:

   ```
   SERVICE: Helper DOMAIN: DEMO ADDRESS: 192.168.31.8 PORT: 51153
   SERVERNAME: britanicus DYNAMIC: YES
   ```

   In this example, `DYNAMIC: YES` shows that the port for the Helper Server has been assigned dynamically.
4. Change the `PORT` setting to the required value.
5. Change the string `DYNAMIC:YES` to `DYNAMIC:NO`. This forces the process to use the same address and port next time it starts.
6. Save the `ServiceData.cfg` file.
7. On the second server, make a backup copy of the `ServiceData.cfg` file.
8. Copy the relevant line from the `ServiceData.cfg` file on the first server to the `ServiceData.cfg` file on the second server.
9. Save the `ServiceData.cfg` file.

Defining a fixed multicast address

You can define which address and port is used by processes for multicast communication by editing the `ServiceData.cfg` configuration file.

About this task

If a Network Manager process needs to know what port another process is running on, it looks up the TCP/IP port defined for that process in the `ServiceData.cfg` file. If there is no port defined for a specific service, then the process broadcasts an address request using multicast. You can define the address that is used for this multicast address request.

The multicast address must be the same on all servers that have Network Manager processes that communicate with each other.

To define the address for multicast communication, complete the following steps.

Procedure

1. Back up and edit the `ServiceData.cfg` file.

2. Edit the line that contains `SERVICE: MulticastService`. Set the ADDRESS and PORT variables.
3. Set the `DOMAIN` to `ANY_PRECISION_DOMAIN`. This means that the service uses the same multicast address for all domains in which it is executed.
   The line should resemble this example:

```
SERVICE: MulticastService DOMAIN: ANY_PRECISION_DOMAIN
ADDRESS: 224.0.0.108 PORT: 33000
```

4. Save and close the `ServiceData.cfg` file.

List of ports used by the product

Network Manager uses different ports for communication: some fixed, some defined by configuration files, and some assigned by the operating system.

The following table describes the default ports used by Network Manager.

Table 5. Default ports used by Network Manager

Port	Protocol	Description
22	SSH over TCP/IP	If SSH support is enabled, the Telnet Helper uses this port to communicate with network devices.
23	Telnet over TCP/IP	If SSH support is not enabled, the Telnet Helper uses this port to communicate with network devices.
161	SNMP	Port 161 is the default port on network devices to which SNMP queries are sent during the discovery and monitoring processes. Defined in the column m_SnmpPort in the database table snmpStack.verSecurityTable.
162	UDP	Default trap port. Used by the Trap polling agent. If more than one application/process needs access to this port, ncp_trapmux, the SNMP trap multiplexer, can be used to forward traps. The SNMP trap multiplexer, the Trap discovery agent, and the Trap polling agent can all be configured to use a different port.
1883	Message Queuing Telemetry Transport (MQTT)	Default port used by Really Small Message Broker for inter-process communication.
4100	TCP/IP	Default ObjectServer port. This must be entered at install time. Defined in `interfaces.Arch` on the ObjectServer workstation. This port is used by the `ncp_g_event` process to communicate with the ObjectServer.
7968	TCP/IP	Default port for access to the Network Manager server from Dashboard Application Services Hub. This is used by the Discovery Configuration GUI and it is defined in the `ServiceData.cfg` configuration file. If you want to change this port, edit the `ServiceData.cfg` configuration file and restart the ncp_model process and the ncp_config process using CTRL.
16310	HTTP	Default port for the Dashboard Application Services Hub. The Dashboard Application Services Hub allocates the next thirteen ports up from the port specified for the Dashboard Application Services Hub during the installation for its own use. By default, this port redirects to 16316.
16311	HTTPS	Default secure port for the Dashboard Application Services Hub.

Table 5. Default ports used by Network Manager (continued)		
Port	**Protocol**	**Description**
33000	TCP/IP	By default, the multicast IP address 225.13.13.13 and port 33000 are used to enable the discovery helpers and discovery agents to locate the Helper server. This multicast address is specified in the file `$NCHOME/etc/precision/ServiceData.cfg`. Once a process has located the Helper server, a TCP connection is established on a port assigned by the operating system.
50000	TCP/IP	Default Db2® database port.
OS-assigned	TCP/IP	TCP ports are assigned by the operating system for TCP communication between processes, for example, the communication between discovery agents and the helper server. If this is an issue, you must ensure that your firewall is external to the Network Manager server, and that all discovery processes are run on the same host.
1521	TCP/IP	Default Oracle database port.

ServiceData configuration file

The `ServiceData` configuration file is a dynamic file that lists TCP and multicast connection information for Network Manager processes.

On startup, every Network Manager service (that is, component or process) that uses a TCP socket adds a line to the ServiceData configuration file. This line contains information about the service. The following information is appended to the configuration file:

- The service name
- The service domain
- The service IP address
- The service port number
- The server on which the process is being run

In the following example configuration file, the first service called MulticastService shows the multicast address and port number. The second service shows that the Helper service is running on the DEMO domain, and includes information about the IP address, port number and the name of the server where the Helper service is running. DYNAMIC: YES means that the port is assigned by the operating system each time the process starts. DYNAMIC: NO defines a fixed port.

```
--
-- Server data file - contains info on servers and the general multicast
-- address to use.
--
SERVICE: MulticastService DOMAIN: ANY_PRECISION_DOMAIN ADDRESS: 225.13.13.13
PORT: 33000

SERVICE: Helper DOMAIN: DEMO ADDRESS: 192.168.31.8 PORT: 51153
SERVERNAME: britanicus DYNAMIC: YES
```

Chapter 5. Administering users

Use the functions of the Web console to provide access to the Web-based interfaces for users, based on the default user roles and user groups. Users and profiles for the OQL Service Provider are managed separately.

About this task

The user interfaces can be categorized as follows:

Web applications
Network Manager includes the following Web applications:

- Network Discovery GUI
- Network Polling GUI
- Topoviz Hop View
- Topoviz Network Views
- SNMP MIB Browser
- Structure Browser

To add and manage users, refer to the information about *Managing Jazz for Service Management users and groups* in the Jazz for Service Management IBM Knowledge Center at https://www.ibm.com/support/knowledgecenter/SSEKCU.

OQL Service Provider
User authentication for the OQL Service Provider is managed separately from Web application users. See *Configuring OQL Service Provider authentication* in the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

Default users

Several users are supplied with Network Manager.

Users and their groups

The following table describes users that are present after installation, along with their groups.

Table 6. Users present after installation

User name	Group	Password	Description
`smadmin`	None	Defined during installation. The default value for a basic installation is netcool. The administrator should change this password.	The administrator for Dashboard Application Services Hub. In a new installation, this user has permissions to administer users, groups, roles, and pages. Defined in the file-based user repository.

© Copyright IBM Corp. 2006, 2021

Table 6. Users present after installation (continued)			
User name	**Group**	**Password**	**Description**
`itnmadmin`	Network_Manager_IP_Admin	Defined during installation.	The administrator for Network Manager. In a new installation, this user has permissions to administer all of the Network Manager Web applications. Defined in the user repository chosen during installation. This user also has the following Dashboard Application Services Hub roles by default: • administrator • chartAdministrator • chartCreator
`itnmuser`	Network_Manager_User	Defined during installation.	An example operator user for Network Manager. Defined in the user repository chosen during installation.

Network Manager user roles

Network Manager defines a number of default roles, which provide users with the ability to perform a predefined set of activities within Web applications.

Access to the Web applications and to functions within the Web applications depends on the roles that are assigned to users. Network Manager roles are usually assigned to users by using groups. Users can also have roles assigned to them from other products. After the administrator adds or removes roles, the revised function is not available to users until users log out and log back in to the Dashboard Application Services Hub.

Note: For information about the user roles that are defined by Cognos Analytics see the Cognos Analytics Knowledge Center at the following web address: https://www.ibm.com/support/knowledgecenter/SSEP7J.

The following table lists all the default roles that are defined by Network Manager.

Table 7. Network Manager roles		
Role	**Assigned to group**	**Description**
`ncp_bookmark_admin`	`Network_Manager_IP_Admin`	User can modify network view bookmark permissions.
`ncp_config`	`Network_Manager_IP_Admin`	User can save any configuration changes that they have made.
`ncp_config_editor`	`Network_Manager_IP_Admin`	User can edit the following widgets. Network Discovery Configuration Configure NCIM Database Access
`ncp_disco_config`	`Network_Manager_IP_Admin`	User can view and edit the discovery configuration settings.

Table 7. Network Manager roles (continued)

Role	Assigned to group	Description
ncp_disco_config_alter_domain	Network_Manager_IP_Admin	User can change the domain for which they are configuring a discovery.
ncp_disco_editor	Network_Manager_IP_Admin	User can edit the Network Discovery Status widget.
ncp_disco_status	Network_Manager_IP_Admin	User can view the status of a discovery as it is running.
ncp_disco_status_control	Network_Manager_IP_Admin	User can start or stop the discovery, or run a discovery with same configuration settings. This role is ineffective without the role Network Manager IP Discovery Status.
ncp_disco_status_alter_domain	Network_Manager_IP_Admin	User can change the domain from which they are getting discovery status. **Note:** Do not remove this role from discovery administrators.
ncp_event_analytics	Not assigned to a group by default.	Enables Event Analytics right-click tools on devices in the topology graph.
ncp_gis	Not assigned to a group by default.	User can open geographical views.
ncp_gis_admin	Not assigned to a group by default.	User can edit portlet layout preferences in geographical views.
ncp_hopview	Network_Manager_User	User can access the Hop View.
ncp_hopview_editor	Network_Manager_IP_Admin	User can edit the Network Hop View widget.
ncp_manage_unmanage	Network_Manager_IP_Admin	User can set devices to managed and unmanaged status.
ncp_mibbrowser	Network_Manager_User	User can access the MIB Browser.
ncp_mibbrowser_config	Network_Manager_User	User can access the MIB Browser for configuration purposes.
ncp_mibbrowser_editor	Network_Manager_IP_Admin	User can edit the SNMP MIB Browser widget.
ncp_mibgraph_default_properties_config	Network_Manager_IP_Admin	User can change the MIB graph default properties. This role is ineffective without the following Network_Manager_User group roles: ncp_mibgraph_user, ncp_mibgraph_config, ncp_mibbrowser.
ncp_mibgraph_config	Network_Manager_IP_Admin, Network_Manager_User	Enables access to the SNMP MIB Graph widget and right-click tools.

Table 7. Network Manager roles (continued)

Role	Assigned to group	Description
ncp_mibgraph_editor	Network_Manager_IP_Admin	User can edit the SNMP MIB Graph widget.
ncp_mibgraph_user	Network_Manager_User	User can access SNMP MIB Graph.
ncp_monitor_policy	Network_Manager_IP_Admin	Enables access to the Configure Poll Policies widget, as well as access to the Create Poll Policy right-click tool.
ncp_monitor_editor	Network_Manager_IP_Admin	User can edit the following widgets. Configure Poll Definitions Configure Poll Policies
ncp_monitor_policy_alter_domain	Network_Manager_IP_Admin	User can select a domain other than the default for poll policies.
ncp_monitor_template	Network_Manager_IP_Admin	User can create a new poll policy definition.
ncp_networkhealth_dashboard	Network_Manager_User	User can access the Network Health Dashboard.
ncp_networkhealth_dashboard_admin	Network_Manager_IP_Admin	User can edit Network Health Dashboard widgets.

Table 7. Network Manager roles (continued)

Role	Assigned to group	Description
ncp_networkview	Network_Manager_User	User can access the Network Views and to display any of the following views: • User Views: Network views that are created by the user. • Group Views: Views that are assigned to the group or groups that this user belongs to. • Global Views: Views accessible to all users regardless of the group to which they belong. Users with this role can not change the view layout, unless the administrator gives them access to the Hierarchichal, Symmetric, Circular, and Tabular layout buttons. To enable users to change (but not save) the layout, set the topoviz.networkview.readonly.enablelayout=true option in the $NMGUI_HOME/profile/etc/tnm/topoviz.properties file. To grant more permissions to users, assign a different role, such as ncp_networkview_admin_user.
ncp_networkview_admin_global	Network_Manager_IP_Admin	User can create, edit, partition, and delete Global Views. These are views accessible to all users regardless of the group to which they belong. User can also perform Move operations on network views within the global views.
ncp_networkview_admin_group	Network_Manager_IP_Admin	User can create, edit, partition, and delete Group Views. These are views assigned to the group or groups that this user belongs to. This role also allows the user to perform Move operations on network views within a group view collection.

Table 7. Network Manager roles (continued)

Role	Assigned to group	Description
ncp_networkview_admin_user	Network_Manager_User	User can create, edit, partition, and delete their own set of network views. This role also allows the user to perform Move operations on network views within a user view.
ncp_networkview_admin_all_users	Network_Manager_IP_Admin	User can create, edit, partition, and delete Private Views. These are private views created by users who have the Network Manager IP Network View - Administer views for user role. This role also allows the user to perform Move operations on network views within a group view collection.
ncp_networkview_editor	Network_Manager_IP_Admin	User can edit the Network Views widget.
ncp_oql	Network_Manager_IP_Admin	User can perform and display the results of select type operations using the Management Database Access page.
ncp_oql_editor	Network_Manager_IP_Admin	User can edit the Management Database Access widget.
ncp_oql_update	Network_Manager_IP_Admin	User can perform and display the results of update type operations using the Management Database Access page.
ncp_pathview	Network_Manager_IP_Admin, Network_Manager_User	User can create, edit, and delete path views.
ncp_pathview_editor	Network_Manager_IP_Admin	User can edit the Path Views widget.
ncp_reporting_user	Network_Manager_IP_Admin, Network_Manager_User	Adds the Cognos Reporting menu item.
ncp_reporting_admin	Not assigned to a group by default.	This role is not currently used.
ncp_rest_api	Network_Manager_IP_User	Required for access to GUI elements that use RESTful APIs. Leave this role assigned to all users.
ncp_structurebrowser	Network_Manager_User	User can use the Structure Browser.
ncp_structurebrowser_editor	Network_Manager_IP_Admin	User can edit the Structure Browser widget.
ncp_structureview_entitysearch	Network_Manager_User	User can search entities in the Structure Browser.

Table 7. Network Manager roles (continued)		
Role	**Assigned to group**	**Description**
ncp_structureview_ interport_navigation	Network_Manager_User	User can navigate from a port on one device to a port on another device in the Structure Browser.
ncp_topo_mgmt	Network_Manager_IP_Admin	User can add and remove devices and connections to the topology using the topology management functionality available within the **Network Hop View**.
ncp_webtools	Network_Manager_User	User can use the WebTools.
ncp_webtools_editor	Network_Manager_IP_Admin	User can edit Web Tools, which is a set of GUIs available form the right-click menu on a device in the topology map.
netcool_rw	Not assigned to a group by default.	User can use the Management Database Access and Network Polling widgets.
noi_npi	Network_Manager_User	User can view the **Device Dashboard** and, in particular, the **Performance Insights** widget used in this dashboard.
Fix Pack 1 Fix Pack 1 noi_npi_admin	Network_Manager_IP_Admin	User can edits the **Device Dashboard** and, in particular, the **Performance Insights** widget used in this dashboard.

User roles for charting

Users must have the user IDs assigned to a chart role before they can see and work with the charting functions.

The main administrator of Jazz for Service Management already has the chartAdministrator role, and can assign users to any of the three chart roles that are available. Logged in users will have no access privileges to the charting features if their user ID has not been assigned to a chart role. These are the capabilities of the chart roles:

chartAdministrator
Users with this role can create and delete charting connections to data sources, upload charts, and can clear the charting cache (useful for troubleshooting).

chartCreator
Users with this role can upload charts, view, and edit them. They cannot create or delete chart connections nor can they clear the charting cache.

chartViewer
Users assigned to this role can select and view charts, but cannot modify them or their preferences. They cannot upload charts, create connections, or clear the charting cache.

Roles are assigned through **Users and Groups > Administrative User Roles**.

User groups

Use groups to organize users into units with common functional goals. Several Network Manager groups are created on installation.

Default user groups

The following groups are supplied with Network Manager. Roles are assigned to these groups during installation.

Network Manager IP Admin
Assign all Network Manager administrators to this group in order to give the users administrative permissions for the Network Manager Web applications.

Network Manager User
Assign all Network Manager end users and operators to this group in order to give the users permissions to use the Network Manager Web applications.

Chapter 6. Administering system passwords

In addition to user passwords, Network Manager uses a number of passwords internally and when interacting with the network.

About this task

All password encryption in Network Manager is performed using FIPS 140–2 compliant algorithms.

Restriction: All passwords on a given server must be encrypted with the same `conf.key` file. The most robust way to set up SNMP and command-line access passwords is by using the **Discovery Configuration GUI**. If you copy any files containing encrypted passwords from one server to another, you must also copy the `conf.key` file that was used to encrypt them, and you must ensure that all passwords on the server are encrypted using that key. You can check the passwords used in discovery by validating them using the **Discovery Configuration GUI**, before running a discovery.

Restriction: All passwords used in Network Manager must conform to the password policies of the server or system environment.

Encrypting or decrypting a password manually

If you set a password using a configuration file, you must encrypt or decrypt the password manually. By default the **ncp_crypt** command encrypts the password provided. However, if you specify the decrypt option, then the password is decrypted.

About this task

Complete these steps to encrypt or decrypt a password from the configuration file.

Note: All password encryption in Network Manager is performed using FIPS 140–2 compliant algorithms.

Procedure

1. Stop Network Manager.
2. Encrypt or decrypt the required password from the command line using the **ncp_crypt** utility in the `ITNMHOME/bin` directory.

   ```
   ncp_crypt -password password [ -decrypt ] [ -help ] [ -version ]
   ```

3. Configure an insert in the relevant configuration file.
   a) Use the output from the **ncp_crypt** encryption utility.
   b) Set the value of the `m_EncryptedPwd` field to 1.
4. Restart Network Manager.

Example

To encrypt the password, type the following command.

```
ncp_crypt -password mypassword
```

To decrypt a password you use the same utility that is used to encrypt the password, but with an additional command line argument.

```
ncp_crypt -decrypt -password @44:xXd7WUIC8teZDhLs8RQ1VjArw8HmUtNCwWs/VrVIxqI=@
```

Related tasks

Starting and stopping Network Manager

© Copyright IBM Corp. 2006, 2021

Your options for starting and stopping Network Manager are explained here.

Changing the encryption key

You can change the encryption key that Network Manager uses when performing password encryption.

Before you begin

Before changing the encryption key, you must first decrypt all the passwords currently used in configuration files using the **ncp_crypt** utility in the `ITNMHOME/bin` directory:

```
ncp_crypt -password password -decrypt
```

Where *password* is the password to decrypt.

About this task

During installation of Network Manager, a 128–bit encryption key is generated and is stored in the following location: `$NCHOME/etc/security/keys/conf.key`. You can change the encryption key using the Tivoli Netcool/OMNIbus utility **nco_keygen**.

Note: If you want to change the encryption key length, see *Configuring encryption length and type* in the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

To change the encryption key:

Procedure

1. Shut down all Network Manager processes.

   You can use the `itnm_stop` command.
2. If you have changed the length of the encryption key, edit the `$NCHOME/etc/precision/ConfigSchema.cfg` file and change the value that is inserted into `config.settings.m_KeyLength` to the length of the new key in bits. Permitted values are 128, 192 and 256.
3. Use the **nco_keygen** utility to generate a new encryption key. Ensure that you specify the output file as `$NCHOME/etc/security/keys/conf.key`.
4. Restart all Network Manager processes.

   You can use the `itnm_start` command.
5. Using the new encryption key, reencrypt all the passwords currently used in configuration files using the **ncp_crypt** utility by typing the following command.

   ```
   ncp_crypt -password password
   ```

   Where *password* is the password to encrypt.

Deactivating password encryption

You can configure Network Manager to deactivate password encryption. If you do this then passwords entered using the GUIs are written to disk in plain text.

About this task

To deactivate password encryption:

Procedure

1. Edit the **ncp_config** configuration file, `ConfigSchema.cfg`.
2. Configure the following insert to the config.settings table:

```
insert into config.settings
(
        m_EncryptPasswords,
        m_EncryptionKeyFile,
)
values
(
        0,
        ""
);
```

The above insert specifies no encryption (`m_EncryptPasswords = 0`), and to use the default encryption key

List of passwords in Network Manager

Any password changes should be made using the Network Manager GUIs where possible.

By default, Network Manager encrypts all passwords entered using the Network Manager GUIs. Some passwords cannot be changed using a GUI and can only be changed by configuring insert statements in the relevant configuration file. If you set a password using a configuration file, you must encrypt the password manually.

The following table lists all the passwords in Network Manager, and specifies how to change the password.

Table 8. Network Manager passwords

Access required to	Password type	Description	Change using
Telnet	Privileged mode password	Configured as part of discovery configuration. Network Manager needs this password in order to access a network device using Telnet.	Network Discovery GUI
Telnet	Password	Configured as part of discovery configuration. Network Manager needs this password in order to access a network device using Telnet.	Network Discovery GUI
SNMP	Community string	Configured as part of discovery configuration. Network Manager needs this password in order to access a network device using SNMP.	Network Discovery GUI
SNMP	Version 3 Authentication password	Configured as part of discovery configuration. Network Manager needs this password in order to access a network device using SNMP.	Network Discovery GUI
SNMP	Version 3 Private password	Configured as part of discovery configuration. Network Manager needs this password in order to access a network device using SNMP.	Network Discovery GUI

Table 8. Network Manager passwords (continued)

Access required to	Password type	Description	Change using
NCIM Database	Password for command-line access to topology database	Provides access to the NCIM topology database.	The `$NCHOME/etc/precision/DbLogins.`*`DOMAIN`*`.cfg` and `$NCHOME/etc/precision/MibDbLogin.cfg` configuration files.
NCIM Database	Access settings used by Network Manager Web applications	You need the NCIM topology database password in order to use GUIs that query the NCIM database.	Database Access Configuration GUI
Tivoli Netcool/OMNIbus ObjectServer	ObjectServer secure password	The Event Gateway needs this password in order to access the ObjectServer for event enrichment activities.	Configuration file insert
Web applications	`tnm.properties` password	Allows GUI access to the NCIM database.	Web applications

Chapter 7. Administering management databases

Use either the GUI-based Management Database Access page or the OQL Service Provider to access the databases of any process.

About this task

Querying management databases using the Management Database Access page

Use the Management Database Access page to perform queries on Network Manager component databases.

About this task

Logging into the Management Database Access page

To log in to the Management Database Access page:

Procedure

Click the **Administration** icon and select **Network** > **Management Database Access**.

Issuing a query using the Management Database Access page

Use the Management Database Access page to issue simple and complex queries against Network Manager databases.

About this task

To issue a database query using the Management Database Access page:

Procedure

1. Click the **Administration** icon and select **Network** > **Management Database Access**.
2. Specify a value for the following fields.

   Domain
   Select the domain in which to issue the OQL query.

   Service
   Select the service that you want to query.

3. To issue a single-line query, type the query in the **Query** field and click **Go**  .
4. To issue a multiple-line query:

   a) Click **Advanced OQL Query**  .

   b) Type the query in the **OQL Command** field and click **OK**.

   c) Click **Go**  .

   Tip: To skip clicking **Go**, append ; go to multiple-line queries.

Results

© Copyright IBM Corp. 2006, 2021

Listing the databases and tables of the current service

You can explore the databases of a service, the tables of those databases, and the columns of those tables.

About this task

Listing the databases of a service using the OQL Workbench

To display a list of the databases of the service you are logged into, use the **`show databases`** command.

About this task

To list the databases of a service:

Procedure

1. Click the **Administration** icon and select **Network** > **Management Database Access**.
2. Specify a value for the following fields.

   Domain
   Select the domain in which to issue the OQL query.

   Service
   Select the service that you want to query.

3. Click **Advanced OQL Query**  . In the **OQL Command** field, type the following query:

```
show databases;
```

Sample output

The following example output shows the databases of the ncp_model service:

```
[ 'dbModel', 'master', 'model', 'ncimCache' ]
```

Listing the tables of a database using the Management Database Access page

To display a list of the tables of a database, use the **`show tables from`** command.

About this task

To list the tables of a database:

Procedure

1. Click the **Administration** icon and select **Network** > **Management Database Access**.
2. Specify a value for the following fields.

   Domain
   Select the domain in which to issue the OQL query.

   Service
   Select the service that you want to query.

3. Click **Advanced OQL Query**  . In the **OQL Command** field, type the following query:

```
show tables from
database_name;
```

Sample output

The following example output shows the tables of the `master` database:

```
[ 'entityByName', 'entityByNeighbor', 'containers' ]
```

Listing the columns of a database table using the Management Database Access page

You can display a list of the columns of a database table using the **show table** command.

About this task

To list the columns of a database table:

Procedure

1. Click the **Administration** icon and select **Network** > **Management Database Access**.
2. Specify a value for the following fields.

   Domain
   Select the domain in which to issue the OQL query.

   Service
   Select the service that you want to query.

3. Click **Advanced OQL Query**  . In the **OQL Command** field, type the following query:

   ```
   show table
   database_name.table_name;
   ```

   database_name is the name of the database, and *table_name* is the name of the required table.

Querying management databases from the command line

You can use the OQL Service Provider to perform queries on Network Manager component databases.

About this task

Once you have logged into the OQL Service Provider, you can issue OQL statements to act on the databases of the service that you are logged into. You must terminate statements with a semi-colon (;) and the **go** keyword. You can also use the **send** keyword instead of **go**.

You can configure the OQL Service Provider to require authentication against NCIM or the ObjectServer. For more information, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

Starting the OQL Service Provider

Start the OQL Service Provider in order to log into the databases of a given Network Manager process.

About this task

Type the following command:

```
ncp_oql -domain DOMAIN_NAME -service SERVICE_NAME [-username USERNAME ] [-password
 PASSWORD ] [ -latency LATENCY ]
```

In this command:

- *DOMAIN_NAME* is name of the domain to query.
- *SERVICE_NAME* is the name of the Network Manager process to query.

- *USERNAME* is the username to authenticate with. This argument is only required if the OQL Service Provider has been configured to require authentication.
- *PASSWORD* is the password to authenticate with. This argument is only required if the OQL Service Provider has been configured to require authentication.
- *LATENCY* is the maximum time in milliseconds (ms) that the service provider waits to connect to another Network Manager process using the messaging bus. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic. The default value is 3000 (equivalent to 3 seconds). You might want to increase this value as the default value might not be long enough to get a response from a large or busy OQL database.

Listing the databases and tables of the current service using the OQL Service Provider

You can explore the databases of a service, the tables of those databases, and the columns of those tables.

Listing the databases of a service using the OQL Service Provider

To display a list of the databases of the service you are logged into, use the **show databases** command.

About this task

To list the databases of a service using the OQL Service Provider

Procedure

1. Start the OQL Service Provider.
2. Type the following query:

```
show databases;
go
```

Sample output

The following example output shows the databases of the ncp_model service:

```
{
        databases = [ 'master', 'translations' ]
}
```

Related tasks

Starting the OQL Service Provider

Start the OQL Service Provider in order to log into the databases of a given Network Manager process.

Listing the tables of a database using the OQL Service Provider

To display a list of the tables of a database, use the **show tables from** command.

About this task

To list the tables of a database using the OQL Service Provider:

Procedure

1. Start the OQL Service Provider.
2. Type the following query:

```
show tables from
database_name;
go
```

Sample output

The following example output shows the tables of the `master` database:

```
{
        tables = [ 'entityByName', 'entityByNeighbor', 'containers' ]
}
```

Related tasks

Starting the OQL Service Provider
Start the OQL Service Provider in order to log into the databases of a given Network Manager process.

Listing the columns of a database table using the OQL Service Provider

You can list the columns of a database table using the **show table** command.

About this task

To issue a database table query using the OQL Service Provider:

Procedure

1. Start the OQL Service Provider.
2. Type the following query:

```
show table
database_name.table_name;
go
```

database_name is the name of the database, and *table_name* is the name of the required table.

Related tasks

Starting the OQL Service Provider
Start the OQL Service Provider in order to log into the databases of a given Network Manager process.

Using OQL queries in scripts

You can launch the service provider in a special mode that executes a single specified query and disconnects from the service provider.

About this task

This allows OQL queries to be used in scripts.

The following example shows the `-query` option in use.

```
ncp_oql -domain NCOMS -service Disco -query "select * from disco.status;"
```

The above example performs a single query on the `disco.status` database table and disconnects from the OQL Service Provider. In order to perform this query, the **ncp_disco** process would have to be running in the NCOMS domain, and the specified user name and password combination would have to be valid.

Any acceptable OQL query can be specified with the `-query` option. The query must be terminated with a semi-colon but not the **go** keyword.

Exiting the OQL Service Provider

When you have finished issuing OQL queries, exit the OQL Service Provider.

About this task

To exit the service provider, type the following command:

```
quit
```

OQL Service Provider tips

OQL Service Provider provides a number of commands to facilitate interaction with the command line.

Restriction: These commands only work in the OQL Service Provider. They do not work in the Management Database Access page.

Tip: By default, null values are not displayed in the returned results. To display null values, use the `-displayNulls` command line option when starting **ncp_oql**.

Show history of commands

Use the `hist` command to show the most recent commands.

Using the `hist` command, you can display up to the thousand (1,000) most recent commands.

Sample

This sample shows how to use the `hist` command:

```
history

        1:      select * from services.unManaged;
        2:      select * from services.unManaged where serviceName like 'dh';
        3:      select count(*) from services.unManaged;
```

Execute a previous command

Use the `!` command together with a number from the command history list to repeat a recent command. Use the `!!` command to repeat the most recent command.

Sample

This sample shows how to use the `!` command:

```
history

        1:      select * from services.unManaged;
        2:      select * from services.unManaged where serviceName like 'dh';
        3:      select count(*) from services.unManaged;

!2
```

This executes the second command in the history list and produces the following output:

```
{
        serviceName='ncp_dh_dns';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=23;
        processId=10734;
}
{
        serviceName='ncp_dh_snmp';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=24;
```

```
        processId=10750;
}
{
        serviceName='ncp_dh_arp';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=25;
        processId=10872;
}
{
        serviceName='ncp_dh_telnet';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=52;
        processId=11424;
}
{
        serviceName='ncp_dh_ping';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=67;
        processId=13399;
}
( 5 record(s) : Transaction complete )
```

Turn on tabular display mode

Use the `tabon` command to turn on tabular display mode.

Sample

This sample shows how to use the `tabon` command:

```
tabon
select * from services.unManaged where serviceName like 'dh';
go
```

This produces the following output:

```
+---------------+----------------------------------------+--------------------
| serviceName   | servicePath                            | argList
+---------------+----------------------------------------+--------------------
| ncp_dh_dns    | $PRECISION_HOME/platform/$PLATFORM/bin/ | ['-domain','NCOMS']
| ncp_dh_snmp   | $PRECISION_HOME/platform/$PLATFORM/bin/ | ['-domain','NCOMS']
| ncp_dh_arp    | $PRECISION_HOME/platform/$PLATFORM/bin/ | ['-domain','NCOMS']
| ncp_dh_telnet | $PRECISION_HOME/platform/$PLATFORM/bin/ | ['-domain','NCOMS']
| ncp_dh_ping   | $PRECISION_HOME/platform/$PLATFORM/bin/ | ['-domain','NCOMS']
+---------------+----------------------------------------+--------------------

----+-----------+-----------+
    | serviceId | processId |
----+-----------+-----------+
    | 23        | 10734     |
    | 24        | 10750     |
    | 25        | 10872     |
    | 52        | 11424     |
    | 67        | 13399     |
----+-----------+-----------+

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

Turn off tabular display mode

Use the `taboff` command to turn off tabular display mode.

Sample

This sample shows how to use the `tabon` command:

```
taboff
select * from services.unManaged where serviceName like 'dh';
go
```

This produces the following output:

```
{
        serviceName='ncp_dh_dns';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=23;
        processId=10734;
}
{
        serviceName='ncp_dh_snmp';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=24;
        processId=10750;
}
{
        serviceName='ncp_dh_arp';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=25;
        processId=10872;
}
{
        serviceName='ncp_dh_telnet';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=52;
        processId=11424;
}
{
        serviceName='ncp_dh_ping';
        servicePath='$PRECISION_HOME/platform/$PLATFORM/bin/';
        argList=['-domain','NCOMS'];
        serviceId=67;
        processId=13399;
}
( 5 record(s) : Transaction complete )
```

Chapter 8. Administering the NCIM topology database

Network topology information is held in the NCIM database.

About this task

Changing the NCIM access details

If you change the host name, port, password, or database name of the NCIM database, you must complete some configuration tasks to enable Network Manager to connect to the database.

About this task

To change the NCIM access details, complete the following steps.

Procedure

Change the host name, port, password, or database name in the database.

To change the host name, port, password, or database name, refer to the configuration documentation for your version of the appropriate topology database:

- For information on installing and configuring Oracle, refer to the Oracle documentation at http://docs.oracle.com/en/database/.
- For information on installing and configuring Db2, refer to the Db2 documentation at http://www.ibm.com/support/knowledgecenter/SSEPGG/welcome
- For information about changing the password, refer to "Updating NCIM access settings for the Web applications" on page 65.

What to do next

After you change the access details in the database, update the settings in the Network Manager components that connect to the NCIM database.

Related tasks

Database access troubleshooting script

In the event of problems with access to the topology database, historical polling database, or polling database, run the **ncp_db_access.pl** script. This script checks database setup and determines whether access to the databases is being prevented by firewalls.

Updating NCIM access settings for the Web applications

If you changed the NCIM settings, you must configure access to NCIM for the Network Manager Web applications.

About this task

The NCIM access settings for the Web applications are set as part of the product installation. Update the settings only if you changed the NCIM access details.

Changing settings by using the GUI is less error-prone. If you are unable to use the GUI to change the settings, you can edit the access details in the `$NMGUI_HOME/profile/etc/tnm/tnm.properties` file.

To configure NCIM access settings by using the GUI, complete the following steps:

© Copyright IBM Corp. 2006, 2021

Procedure

1. Click the **Administration** icon and select **Network** > **Database Access Configuration**.
   The **Configure NCIM Database Access** widget is displayed.
2. Enter the host that the database is installed on in the **Database Host** field. By default, this host is the same host that Network Manager is installed on.
3. Enter the port that is used by the database in the **Database Port** field.
4. Enter the user name that was entered during installation of the database in the **Username** field.
5. Enter the password that was entered during installation of the database in the **Password** field.
6. Confirm the password.

Related reference

Topoviz screen is blank
If Topoviz fails to start, or starts with a blank screen, refresh the browser window. If the Network Manager splash screen does not appear, check the topology database access settings.

Updating the NCIM access details used by Reporting Services

If you change the NCIM access details, configure Cognos Analytics to use the new details.

Before you begin

You need the password for the Dashboard Application Services Hub administrator user (typically the smadmin user).

About this task

To configure Cognos Analytics to use different access details for the NCIM database, complete the following steps.

Procedure

1. Change the password used by Cognos Analytics by configuring the data source properties for reports and for all data sources. See the Cognos Analytics Knowledge Center at https://www.ibm.com/support/knowledgecenter/SSEP7J.
2. For each data source, use the Cognos® GUI to change the username or password.
   a) Click the **Reporting** icon and select **Cognos Reporting**. Within the widget select **Manage** > **Administration Console**.
   b) Click the **Configuration** tab.
   c) Click **NCIM** and then click **NCIM** again.
      The breadcrumb trail at the top of the GUI reads **Directory** > **Cognos** > **NCIM** > **NCIM**.
   d) Select the check box next to **ncim** and click **More** > **Set Properties**.
   e) On the **Signon** tab, click **Edit the Signon** and type the database user name and the new password.

Updating NCIM access settings in the Network Manager core components

If you change the NCIM access details, configure the Network Manager core components to use the new details.

About this task

To configure the Network Manager core components to use the new details, complete the following steps.

Procedure

1. If you want to change only the password, run the **ncp_password_update.pl** script. For information about running this Perl script, refer to *ncp_password_update.pl* in the *IBM Tivoli Network Manager Reference*.
2. Optional: If you are not sure whether the password is correct, turn encryption off by setting `tnm.database.password.encrypted=false` in the `tnm.properties` file, and enter the password in clear text. Do this only for troubleshooting purposes, and remember to turn encryption back on afterwards.
3. To change other access settings, such as the host name or port, run the **set_db_details.pl** script. For information about running this script, refer to *set_db_details.pl* in the *IBM Tivoli Network Manager Reference*.

Results

After changing the password, you can use the `NCHOME/precision/scripts/perl/scripts/ncp_db_access.pl` script to verify access. For more information on the `ncp_db_access.pl` script, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Re-creating network views

If you create a new NCIM database schema and want your Network Manager GUI to use the new schema, you must configure the GUI to access the new database. You also need to re-create your network views.

Before you begin

Ensure you configured GUI access to the new topology database.

About this task

Network views are auto-provisioned by the `default.xml` and `itnmuser.xml` files when installing Network Manager, and are then created in the database. However, if you do not create the topology database schemas during installation, or subsequently change your database, then you need to re-create your views.

To re-create network views:

Procedure

1. Go to `$NMGUI_HOME/profile/etc/tnm/autoprovision/`.
2. Rename the files called *`filename`*`.xml.processed` to *`filename`*`.xml`.
3. Save and close the files.

Creating the topology database schemas

You can set up the topology database during installation. If you need to set up a database after installation for an existing Network Manager, then you can do this manually using the scripts provided.

About this task

You must create the topology database before you can use Network Manager.

For information about creating the topology databases, see the ***Installing and configuring a topology database*** topic within the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

Creating Db2 topology database schemas on UNIX

You can use scripts to create the topology database schemas in a Db2 database on UNIX. Different users are required to run the different scripts.

About this task

To create the topology database:

Procedure

1. Make sure that you followed the prerequisites for installing Db2, and install and configure the Db2 database.
2. Ensure the database creation scripts are available on the database server.
   - If the Db2 database is on the same server as Network Manager, and the Network Manager has already been installed, the database creation scripts are available in `$ITNMHOME/scripts/sql`.
   - If the Db2 database is on a different server, either copy the contents of `$ITNMHOME/scripts/sql` to the Db2 server, or locate the `db2_creation_scripts.tar.gz` file at the top level which is available from IBM Installation Manager as a separate package.
3. As the administrative user, for example the `db2inst1` user, run the `create_db2_database.sh` script as the Db2 administrative user by typing the following command on the Db2 server:

   `./create_db2_database.sh` *`database_name user_name`* `-force`

   Where:

   `database_name`
   Is the name of the database.

   `user_name`

   Is the Db2 user to use to connect to the database.

   Important: This user must not be the administrative user. This user must be an existing operating system and Db2 user.

   `-force`
   Is an argument that forces any Db2 users off the instance before the database is created.

   For example, to create a Db2 database that is called `ITNM` for the Db2 user `ncim`, type:

   ```
   ./create_db2_database.sh ITNM ncim
   ```

   After you run `create_db2_database.sh`, restart the database as the Db2 administrative user as follows: run **db2stop** and then run **db2start**.
4. Create the database schema. To create the database schema, use one of the following approaches, you cannot use both approaches.
   a) To create the database schema from the Network Manager server, run the `$ITNMHOME/scripts/sql/create_all_schemas.sh` command on the Network Manager server as the administrative user as follows:

   `./create_all_schemas.sh` *`database_type database_name host user_name password port`*

   Where:

   database_type
   Identifies the type of database to create. In this case, it is db2.

   database_name
   Specifies the name of the database.

   host
   Specifies the server host name or IP address where the database is installed.

user_name
Identifies the Db2 user to use to connect to the database.

Important: This user must not be the administrative user. This user must be an existing operating system and Db2 user.

password
Provides the password for the user.

port
Specifies the port that is used by the database.

The following example creates the NCIM schemas on a Db2 database that has the service name DB_123 on a remote host that is called samplehost, on port 9088. The user/password combination for connecting to the database is ncim/password.

```
$NCHOME/precision/scripts/sql/create_all_schemas.sh db2 DB_123 samplehost
ncim password 9088
```

b) Optionally, to create the database schema on the database server, go to the location on the database server where you uncompressed the `db2_creation_scripts.tar.gz` file. As the administrative user, run the following script to populate the databases:

```
./populate_db2_database.sh database_name user_name password > db2.log 2>&1
```

5. Examine the `db2.log` file for any errors.
6. Log in as the Db2 administrator on the Db2 client that is running on the Dashboard Application Services Hub server.
7. Run the `$ITNMHOME/bin/ncp_mib` command to ensure that the `ncmib` database is fully populated with SNMP data from the MIBs before a discovery is run.
8. If you want your Network Manager GUI to use the database schema you created, configure access to the new topology database schema and re-create your network views.

Related tasks

Updating NCIM access settings for the Web applications
If you changed the NCIM settings, you must configure access to NCIM for the Network Manager Web applications.

Re-creating network views
If you create a new NCIM database schema and want your Network Manager GUI to use the new schema, you must configure the GUI to access the new database. You also need to re-create your network views.

Creating Oracle topology database schemas on UNIX

Use scripts to create the topology database schemas in an Oracle database on UNIX.

About this task

To create the topology database, complete the following steps on the server where the Oracle database is installed. Different users are required to run the different scripts.

Procedure

1. Make sure that you followed the prerequisites for installing Oracle, and install and configure the Oracle database.

   For information on installing and configuring Oracle, refer to the Oracle documentation at http://docs.oracle.com/en/database/.
2. Ensure the database creation scripts are available on the database server.
   - If the Oracle database is on the same server as Network Manager, and the Network Manager has already been installed, the database creation scripts are available in `$ITNMHOME/scripts/sql`.

- If the Oracle database is on a different server, either copy the contents of $ITNMHOME/scripts/sql to the Oracle server, or locate the oracle_creation_scripts.tar.gz file at the top level which is available from IM as a separate package.

3. Run the create_oracle_ncadmin_user.sh script on the server where the database is installed. Log in to the Oracle host as the Oracle database administrator, run the create_oracle_ncadmin_user.sh script by supplying the sys user and password. Run the script as in the following example.

```
$NCHOME/precision/scripts/sql/oracle/create_oracle_ncadmin_user.sh
sys
password [-pdb pluggable_database_name]
```

Where the following parameters apply:

password
Specifies the password of the sys user.

-pdb *pluggable_database_name*
Optional: if you are running Oracle 12c with RAC, you must use a pluggable database. In this case, use this parameter to specify the Oracle 12c pluggable database name.

4. To create the database, run the ./create_oracle_database.sh script. As the Oracle database administrator, run the ./create_oracle_database.sh script by supplying the system user and password. On the Network Manager server, the script is in the $ITNMHOME/scripts/sql/oracle directory. Run the script on the server where the database is installed. Run the script as in the following example.

```
./create_oracle_database.sh system password [-asm]
[-pdb pluggable_database_name]
```

Where the following parameters apply:

password
Specifies the password of the system user.

-asm
Specify -asm if your Oracle DB uses ASM.

-pdb *pluggable_database_name*
Optional: if you are running Oracle 12c with RAC, you must use a pluggable database. In this case, use this parameter to specify the Oracle 12c pluggable database name.

5. Create the database schema. To create the database schema, use one of the following approaches, you cannot use both approaches.

a) To create the database schema from the Network Manager server, run the $ITNMHOME/scripts/sql/create_all_schemas.sh command on the Network Manager server as the administrative user as follows:

./create_all_schemas.sh *database_type database_name host user_name password port*

Where:

database_type
Identifies the type of database to create. In this case, it is oracle.

database_name
Specifies the name of the database. For Oracle, this name must be the pluggable database.

host
Specifies the server host name or IP address where the database is installed.

user_name
Identifies the user to use to connect to the database.

password
Provides the password for the user.

port
Specifies the database port.

b) To create the database schema on the database server, go to the location on the database server where you uncompressed the `oracle_creation_scripts.tar.gz` file. As the administrative user, run the `populate_oracle_database.sh` script. Run the script as in the following example:

```
./populate_oracle_database.sh database_user_name password
[-pdb pluggable_database_name]
 > oracle.log 2>&1
```

Where the following parameters apply:

database_user_name
The value can be `system` or `ncim`. You created the `ncim` user in an earlier step.

password
Provides the password for the user.

-pdb *pluggable_database_name*
Optional: if you are running Oracle 12c with RAC, you must use a pluggable database. In this case, use this parameter to specify the Oracle 12c pluggable database name.

6. Examine the `oracle.log` file for any errors.
7. Run the `$ITNMHOME/bin/ncp_mib` command to ensure that the `ncmib` database is fully populated with SNMP data from the MIBs before a discovery is run.
8. If you want your Network Manager GUI to use the database schema you created, configure access to the new topology database schema and re-create your network views.

Related tasks

Updating NCIM access settings for the Web applications
If you changed the NCIM settings, you must configure access to NCIM for the Network Manager Web applications.

Re-creating network views
If you create a new NCIM database schema and want your Network Manager GUI to use the new schema, you must configure the GUI to access the new database. You also need to re-create your network views.

Removing domains from the topology database

When a domain is no longer required, use the `domain_drop.pl` script to remove it from the NCIM topology database. The entire topology for the domain is removed and any poll policies for that domain. The configuration information for the domain and the topology cache are not affected.

Procedure

1. Stop the domain by running the **`itnm_stop`** command.
   For example, to remove the domain OLDDOMAIN:

   ```
   itnm_stop ncp -domain OLDDOMAIN
   ```

   To verify that the processes for the domain were stopped, run the **`itnm_status`** command.
2. In `$NCHOME/precision/scripts/perl/scripts`, run the `domain_drop.pl` script.
   For example, to remove the domain that was stopped in the previous step:

   ```
   NCHOME/precision/bin/ncp_perl domain_drop.pl -domain OLDDOMAIN
   -password password
   ```

Removing all entities from domains

As an alternative to dropping domains from the NCIM database, you can use an SQL DELETE statement to remove all entities from domains. Unlike dropping domains, if you remove all the entities, the mapping between the entityName and entityID persists.

Procedure

To remove all entities from a domain, use a DELETE statement as shown in the following example.

```
delete from ncim.entityData where entityId in (select a.entityId from
ncim.entityData a inner join ncim.domainMembers b on
a.entityid=b.entityid and b.domainMgrId=domainMgrId;)
```

Where *domainMgrId* is the domainMgrId value for the domain.

Results

The entities are removed from the domain. Because the mapping between the entityName and entityID persists, if entities that have same name are rediscovered, they are assigned the same entityId as before deletion.

Removing the topology database

You can remove the topology database if it is no longer required.

About this task

Before removing the topology database, you must stop all processes that connect to the database.

Removing a Db2 topology database on UNIX

You can remove the Db2 database on UNIX using a script.

About this task

To remove a Db2 database, complete the following steps:

Procedure

1. Change to the scripts directory using the following command:

   ```
   cd $NCHOME/precision/scripts/sql/db2
   ```

2. Execute the script using the following command:

   ```
   drop_db2_database.sh NCIM [ force ]
   ```

Results

If you use the optional force option, the script forces any existing Db2 users off the instance before attempting to drop the database.

Removing an Oracle topology database on UNIX

You can remove the Oracle database on UNIX using a script.

About this task

To remove an Oracle database, complete the following steps:

Procedure

1. Ensure that you are logged in as the `system` user.
2. Change to the scripts directory using the following command:

```
cd $NCHOME/precision/scripts/sql/oracle
```

3. Execute the script using the following command:

```
drop_oracle_database.sh system password [ -pdb pluggable_db]
```

Removing an Oracle topology database on Windows

You can remove the Oracle database on Windows using a script.

About this task

To remove an Oracle database, complete the following steps:

Procedure

1. Ensure that you are logged in as the `system` user.
2. Change to the scripts directory using the following command:

```
cd %NCHOME%\precision\scripts\sql\oracle
```

3. Execute the script using the following command:

```
sqlplus system/password < drop_oracle_database.sql
```

Results

If you use the optional force option, the script forces any existing Oracle users off the instance before attempting to drop the database.

Chapter 9. Administering reports

You can create new reports, modify existing reports, and configure user access to reports.

About this task

Data sources for Cognos Analytics are set during installation. If the database details change, configure the data source again using the administrative functions in Cognos Analytics. See the Cognos Analytics Knowledge Center at https://www.ibm.com/support/knowledgecenter/SSEP7J.

Creating and editing reports

You can edit existing reports and create your own reports using the Cognos Analytics tools.

About this task

You can create new reports using the Cognos Query Studio, which is described in the *Query Studio User Guide*, available from the **Help** menu in the **Reporting** > **Common Reporting** widget.

You can edit existing reports using the Cognos Report Studio, which is described in the *Report Studio Professional Authoring User Guide*, available from the **Help** menu in the **Reporting** > **Common Reporting** widget.

Tip: If your network is large and complex, detailed reports can potentially contain very large amounts of data. Reports that are hundreds of thousands of lines long can be more difficult to use, and can cause the reporting components to run out of memory. Ensure that your reports are optimized to return the data that is useful to you.

If you create or edit custom Cognos-based reports, then the report creation or editing procedure will require selection of IBM Common Data Model (CDM) views and attributes. For more information on the CDM views, see the *IBM Tivoli Network Manager Reference*.

Creating a URL to run reports

You can construct a URL to open a report directly in a browser window. These URLs can be used by other applications to run reports.

About this task

To construct a URL to open a report, complete the following steps.

Procedure

1. Locate the report that you want to use and note which parameters are required.
2. Construct a URL similar to the following:

```
https://hostname:port/tarf/servlet/dispatch?
b_action=cognosViewer&ui.action=run&ui.object=/content/package[@name=
'Network Manager']/folder[@name='report_group_name']
/report[@name='report_name']&ui.name=report_name
&run.outputFormat=HTML&domainName=AUTO&report_parameter=
"value"
```

Where

- *hostname* is the name of the server where the Dashboard Application Services Hub is installed.
- *port* is the port number for the Dashboard Application Services Hub.
- *report_group_name* is the name of the group to which the report belongs.

© Copyright IBM Corp. 2006, 2021

- *report_name* is the name of the report that you want to open.
- *report_parameter* is a parameter to be passed to the report.
- *value* is the value of the parameter.

The following example URL opens the IP Routing Info report, which belongs to the Path Views Reports report group, showing path 3323 and device 13.

```
https://10.10.10.108:16311/tarf/servlet/dispatch?b_action=cognosViewer
&ui.action=run&ui.object=/content/package[@name='Network Manager']
/folder[@name='Path Views Reports']/report[@name='IP Routing Info']&ui.name
=IP Routing Info&run.outputFormat=HTML&domainName
=AUTO&pathEntityId=3323&entityId=13
```

Changing the data source isolation level

The default isolation level for the Cognos ODBC data source is set to read committed. If you have multiple domains or have many polls running, or both, you might need to change the isolation level to read uncommitted for performance reasons and to avoid table locks that trigger errors.

About this task

To set the isolation level to read uncommitted, change the NCPOLLDATA data source settings as follows:

Procedure

1. Click the **Reporting** icon and select **Cognos Reporting**. Within the widget select **Manage** > **Administration Console**.
2. Click the **Configuration** tab.
3. Click NCPOLLDATA.
4. Click the **Set properties** icon and then select the **Connection** tab.
5. Select **Read uncommitted** from the **Specify a value** drop down menu.

   Note: Some database vendors use different names for the isolation levels. See the Cognos Analytics help and search for "isolation levels" for information on the levels available and their corresponding names on different database types.
6. Click **OK**.

Chapter 10. Troubleshooting and support

Use this information to help you resolve problems with the product.

Troubleshooting Network Manager

Consult these troubleshooting notes to help determine the cause of the problem and what to do about it.
Related tasks

Administering logs
Network Manager provides logging capabilities for its GUI components and back-end processes. You can set up logging for Network Manager to produce log or trace files that can be used for troubleshooting purposes.

Troubleshooting discovery
You can troubleshoot discovery by monitoring discovery events and by running discovery reports. You can also configure your own discovery events.

Troubleshooting ping polling of the network
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.

Troubleshooting the installation

Use this information to how to troubleshoot errors that might occur during the installation of Network Manager.

The following topics describe the types of error messages that you might encounter during the installation process, and the actions you can take to resolve these issues.

Viewing the installation logs

Viewing the installation logs can be useful for troubleshooting purposes.

About this task

Information about the success of the installation process is recorded in IBM Installation Manager. To view the installation log information, proceed as follows. You can find more information in the IBM Installation Manager Knowledge Center at: https://www.ibm.com/support/knowledgecenter/SSDV2W/im_family_welcome.html.

Note: For general settings of the Network Manager installation, such as information about the version of Network Manager installed, the home location of components, and database connection information, see the `NCHOME/etc/itnm.cfg` file.

Procedure

1. Review the installation history. In IBM Installation Manager click **File** > **Installation History**.
   a) In IBM Installation Manager click **File** > **Installation History**.
   b) Review the status of the installed packages.
      The **Package Group Name** column lists the installed packages, and the **Status** column lists the results of the installation of that package.
   c) Click **View Log** to view the log file for any selected package.
2. You can also filter the various log files by event type. To do this, in IBM Installation Manager click **File** > **View Log**.
   You can filter information in the log files by the following event types:

© Copyright IBM Corp. 2006, 2021

- Error
- Warning
- Note
- Information

3. If necessary, you can export a full set of log data for analysis by IBM. To do this, in IBM Installation Manager click **Help** > **Export Data for Problem Analysis**.

Checking login URL and default ports

If you have trouble logging in, make sure you check the URL format and the ports you use after installation.

URL format

Check that your URL format entered is as follows (shows default ports):

- `https://localhost:16311/ibm/console` (secure access).
- `http://localhost:16310/ibm/console` (nonsecure access).

Where *localhost* is the fully-qualified host name or IP address of the Dashboard Application Services Hub server.

Default ports

16310 is the default nonsecure port number and 16311 is the default secure port number. If your environment was configured during installation with a port number other than the default, enter that number instead.

Troubleshooting the Dashboard Application Services Hub

You can find information on troubleshooting the Dashboard Application Services Hub on the Jazz for Service Management Knowledge Center.

The Jazz for Service Management Knowledge Center is located at https://www.ibm.com/support/knowledgecenter/SSEKCU

Troubleshooting Web Applications

Use this troubleshooting information to help you resolve common problems that might occur when you administer the Web applications.

Device not found

A `Device Not Found` error might occur when you right-click an event in the Tivoli Netcool/OMNIbus Web GUI and click **Find in Hop View**.

This error appears for one of the following reasons:

- There is no corresponding device in the topology. If this is the case, you should check that:
  - You have configured the scope of the discovery so that it includes this device.
  - You have run the appropriate Discovery Agent to discover this device.
  - The device is a supported network device.
  - The device has been discovered, as it may have recently come online and need to be discovered.
- The event came from a probe that has not been configured to include the fields that Network Manager requires to locate the device. This is the most likely cause of the error if the device is in the topology.

Topoviz screen is blank

If Topoviz fails to start, or starts with a blank screen, refresh the browser window. If the Network Manager splash screen does not appear, check the topology database access settings.

Related tasks

Updating NCIM access settings for the Web applications

If you changed the NCIM settings, you must configure access to NCIM for the Network Manager Web applications.

Unable to access domain

If the **Domain** drop-down list does not show the expected domain, check your topology database access settings. Also check the files in the `$NMGUI_HOME/profile/logs/tnm/` directory for relevant information about the GUI, especially the `ncp_topoviz.0.log` and `ncp_topoviz.0.trace` files for relevant information about Topoviz.

Unable to execute right-click tools in Event Viewer

If the **Show Root Cause** or **Show Suppressed Events** context menu options fail to execute and return an error message, this might be because the CGI scripts, which are run when you select these menu options, are unable to find the path to Perl.

If you have installed Perl in a nonstandard location, ensure that you have specified the correct path to Perl in all CGI scripts.

Device in Topoviz appears as generic node

If a device is known to be a switch or router but appears in the **Network Hop View** or in the **Network Views** as a generic node icon, then the device may not be correctly mapped to an icon in its active object class (AOC) file.

Cause

Device was discovered correctly and is mapped to an AOC file. One way to check this is to make sure that in the **Network Views**, the device can be located in one of the Device Class network views.

Resolving the problem

Certain AOC files do not give a visual icon but rather use the statement `visual_icon = ' ';`. In this case the AOC file (and the corresponding device) takes the visual_icon from the super_class of the AOC.

Example

An example is the Extreme.aoc and ExtremeSummit.aoc file. The super_class for Extreme.aoc is Device.aoc file, which uses the 'Device' icon. If you would like any device instantiated as an Extreme.aoc to be seen in **Network Hop View** or in the **Network Views** as a switch or router, edit the AOC file and use the statement `visual_icon = 'Switch';` or `visual_icon = 'Router;` in place of `visual_icon = ' ';`.

Performance impact in network views when many MPLS-TE tunnels are present

The `Path View Engine` process runs as a background process to build all required views for the network topology. When the topology contains many MPLS-TE tunnels, the `Path View Engine` process is highly active while the process is building the views, and in some cases the process might impact the performance of the Network Manager GUI.

If necessary, you can disable the `Path View Engine` process. To disable the `Path View Engine` process, go to `$NMGUI_HOME/profiles/etc/topoviz.properties` and set the `topoviz.pathviewengine.enabled` property to FALSE.

GUIs not appearing

If you integrate Network Manager web applications into your own product, and certain web applications are not appearing, then this might be because of the clickjacking filter. This filter is applied by default to the Database Access GUI and the Discovery Configuration GUI.

About this task

To resolve this issue, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

Not authorized error message when logged into Network Manager

You might be unexpectedly logged out of ITNM, or your might receive unexpected error messages about not being authorised when you are logged in, for example, when you try to access Network Manager online help. One solution to this issue is to change the name of the WebSphere Application Server single sign-on (SSO) cookie to ensure that users logged into Network Manager are always recognized as being authenticated with Network Manager, regardless of any other SSO cookies set by concurrent login sessions to other servers, such as Intranet login sessions.

Perform the following steps to change the name of the WebSphere Application Server single sign-on (SSO) cookie:

1. Log in to the WebSphere admin console as the smadmin user.

   ```
   https://hostname:port/ibm/console
   ```

   Where:

   - *hostname* is the name of the WebSphere Application Server.
   - *port* is the port number associated with the WebSphere Application Server. By default this is the port number of the Dashboard Application Services Hub server + 5, which is 16316.
2. In the navigation tree on the left-hand side, click **Security** > **Global Security**.
3. In the **Global Security** page, click **Web and SIP Security**, and then click **Single Sign-On (SSO)**.
4. In the **LTPA V2 cookie name** field, type the desired SSO cookie name; for example, `ITNMLtpaToken`.

   By default this field is blank, and if left blank the cookie name value defaults to `LtpaToken2`. Hence it is important to change the cookie name to anything other than `LtpaToken2`.
5. Click **OK**.
6. Click **Save directly to the master configuration**.
7. Restart the Dashboard Application Services Hub server.

Troubleshooting reporting

If you have problems with Cognos Analytics, check the troubleshooting information.

Polled data reports contain null values

If reports on polled data contain null values, you might need to clear the polling data.

About this task

Over time, with the addition and removal of entities, the ncmonitor.monitoredInstance table can contain instances of entityIds that have since been removed from the ncim.entity table. This can result in polled data reports containing null information. Complete the following steps to fix this problem:

Procedure

1. Run the following queries to determine whether your database contains null values:

```
select count(*) from ncpolldata.monitoredInstance where instanceType =
'ifIndex' and entityId is null;

select count(*) from ncpolldata.monitoredInstance where instanceType =
'ifIndex' and entityId not in (select entityId from ncim.interface);
```

2. If any values are returned from the above query, use `itnm_stop` to stop the domain and run the following script to delete the polled data for a domain:

```
$NCHOME/precision/bin/ncp_perl $NCHOME/precision/scripts/perl/scripts/domain_drop.pl -domain
domain -clearPollData
```

   Where *domain* is the domain for which you want to delete polled data.

Changing the logging level for reporting

For information on how to set logging and on how to change the logging level for Cognos Analytics, see the Cognos Analytics Knowledge Center at https://www.ibm.com/support/knowledgecenter/SSEP7J.

Related information

Cognos Business Intelligence welcome pageFor more information on setting logging for reporting, search the Cognos Business Intelligence online documentation.

Troubleshooting database access

To troubleshoot database access, check user access or run the database troubleshooting script.

Database access troubleshooting script

In the event of problems with access to the topology database, historical polling database, or polling database, run the **ncp_db_access.pl** script. This script checks database setup and determines whether access to the databases is being prevented by firewalls.

About this task

The **ncp_db_access.pl** script checks database setup and firewall issues for the following databases:

- NCIM topology database
- NCMONITOR polling database
- MIB historical polling database

Procedure

1. Change to the `$NCHOME/precision/scripts/perl/scripts` directory and locate the ncp_db_access.pl program.
2. Issue the following command.

   `perl ncp_db_access.pl -domain` *`domain_name`*
   Where:

   - *`domain_name`* is the name of the required domain.

   For each database the script indicates whether connection is in order or if there are access problems.

Related tasks

Changing the NCIM access details

If you change the host name, port, password, or database name of the NCIM database, you must complete some configuration tasks to enable Network Manager to connect to the database.

Troubleshooting the ITM Agent

To troubleshoot a problem with IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition, gather information about the problem for IBM Software Support, use logging data, and consult the lists of identified problems and workarounds.

For general troubleshooting information, see the *IBM Tivoli Monitoring Troubleshooting Guide*.

For known issues with this version of IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition, refer to the *IBM Tivoli Network Manager IP Edition Release Notes*.

You can resolve some problems by ensuring that your system matches the system requirements. The most up-to-date requirements are in the https://www.ibm.com/software/reports/compatibility/clarity/ (http://www.ibm.com/software/reports/compatibility/clarity/index.jsp).

Gathering product information for IBM Software Support

Before contacting IBM Software Support about a problem you are experiencing with this product, gather the information shown in Table 9 on page 82.

Table 9. Information to gather before contacting IBM Software Support

Information type	Description
Log files	Collect trace log files from failing systems. Most logs are located in a `logs` subdirectory on the host computer.
Network Manager information	Version number and patch level
Operating system	Operating system version number and patch level
Messages	Messages and other information displayed on the screen
Version numbers for IBM Tivoli Monitoring	Version number and patch levels for IBM Tivoli Monitoring and IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition.
Screen captures	Screen captures of incorrect output, if any.
(UNIX systems only) Core dump files	If the system stops on UNIX systems, collect the core dump file from the *`install_dir`*`/bin` directory, where *`install_dir`* is the directory where you installed the monitoring agent.

For information about working with IBM Software Support, see https://www.ibm.com/support/home/ (https://www.ibm.com/support/servicerequest/Home.action).

Using logging

Logging is the primary troubleshooting feature in the monitoring agent. *Logging* refers to the text messages and trace data that is generated by the agent. Messages and trace data are sent to a file.

Trace data captures transient information about the current operating environment when a component or application fails to operate as designed. IBM Software Support personnel use the captured trace information to determine the source of an error or unexpected condition.

Consulting the lists of identified problems and workarounds

Known problems are organized into types such as those in the following list to make them easier to locate:

- Installation, configuration, uninstallation
- Remote deployment
- Agent
- Workspace
- Situation
- Take Action commands

Trace logging

Trace logs are used to capture information about the operating environment when component software fails to operate as designed.

The principal log type is the RAS (Reliability, Availability, and Serviceability) trace log. These logs are in the English language only. The RAS trace log mechanism is available for all components of IBM Tivoli Monitoring. Most logs are in a `logs` subdirectory on the host computer.

Note: The documentation refers to the RAS facility in IBM Tivoli Monitoring as "RAS1."

IBM Software Support personnel use the information captured by trace logging to trace a problem to its source or to determine why an error occurred. All components in the IBM Tivoli Monitoring environment have a default tracing level. The tracing level can be changed on a per-component level to adjust the type of trace information collected, the degree of trace detail, the number of trace logs to be kept, and the amount of disk space used for tracing.

Overview of log file management

Knowing the naming conventions for log files helps you to find the files.

Agent log file naming conventions

IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition log file names adhere to the following naming convention:

Linux and UNIX systems

*hostname*_np_*program_HEXtimestamp-nn*.log

Location of trace log files

Trace log files are located on various systems.

The following tables contain locations, file names, and descriptions of some of the trace logs that can help determine the source of problems with agents.

For more information about the complete set of trace logs that are maintained on the monitoring server, see the *IBM Tivoli Monitoring Installation and Setup Guide*.

Table 10. Trace log files located on the Tivoli Enterprise Monitoring Server	
File name and path	**Description**
• **Windows:** The `IBM Tivoli Monitoring` *timestamp*`.log` file in the *install_dir* `\InstallITM` path • **UNIX:** The `candle_installation.log` file in the *install_dir*`/logs` path • **Linux:** The `candle_installation.log` file in the *install_dir*`/logs` path	Provides details about products that are installed. **Note:** Trace logging is enabled by default. A configuration step is not required to enable this tracing.
The name of the RAS log file is as follows: • **Windows:** *install_dir*`\logs \`*hostname*`_ms_`*timestamp-nn*`.log` • **UNIX:** *install_dir*`/logs/` *hostname*`_ms_`*timestamp-nn*`.log` • **Linux:** *install_dir*`/logs/` *hostname*`_ms_`*timestamp-nn*`.log` **Note:** File names for RAS1 logs include a hexadecimal time stamp. Also on UNIX systems, a log with a decimal time stamp is provided: *hostname*`_np_`*timestamp*`.log` and *hostname*`_np_`*timestamp*`.pid`*nnnnn* in the *install_dir*`/logs` path, where *nnnnn* is the process ID number.	Traces activity on the monitoring server.
The name of the RAS log file is as follows: • **Windows:** *install_dir*`\logs\` *hostname*`_cq_HEXtimestamp-nn.log` • **UNIX:** *install_dir*`/logs/` *hostname*`_cq_HEXtimestamp-nn.log` • **Linux:** *install_dir* `/logs/` *hostname*`_cq_HEXtimestamp-nn.log` **Note:** File names for RAS1 logs include a hexadecimal time stamp. Also on UNIX systems, a log with a decimal time stamp is provided: *hostname*`_np_`*timestamp*`.log` and *hostname*`_np_`*timestamp*`.pid`*nnnnn* in the *install_dir*`/logs` path, where *nnnnn* is the process ID number.	Traces activity on the portal server.

The following table lists trace log files that are located on the Tivoli Enterprise Portal server.

Table 11. Trace log files located on the Tivoli Enterprise Portal server	
File name and path	**Description**
The name of the RAS log file is as follows: • **Windows:** *install_dir*\logs\ *hostname*_cq_HEXtimestamp-nn.log • **UNIX:** *install_dir*/logs/ *hostname*_cq_HEXtimestamp-nn.log • **Linux:** *install_dir* /logs/ *hostname*_cq_HEXtimestamp-nn.log **Note:** File names for RAS1 logs include a hexadecimal time stamp. Also on UNIX systems, a log with a decimal time stamp is provided: *hostname*_np_*timestamp*.log and *hostname*_np_*timestamp*.pid*nnnnn* in the *install_dir*/logs path, where *nnnnn* is the process ID number.	Traces activity on the portal server.

Table 12. Trace log files located on the Network Manager server	
File name and path	**Description**
The RAS1 log files are as follows: • **UNIX:** *hostname*_np_*instance_name*_ knpagent_ HEXtimestamp-nn.log in the *install_dir*/ logs directory • Linux **Linux:** *hostname*_np_*instance_name*_ knpagent_ HEXtimestamp-nn.log in the *install_dir*/ logs directory These logs are in the following directories: • **UNIX:** *install_dir*/logs • Linux **Linux:** *install_dir*/logs On Linux systems, the following additional logs are provided: – *hostname*_np_timestamp.log – *hostname*_np_timestamp. pidnnnnn in the *install_dir*/logs path, where *nnnnn* is the process ID number	Traces activity of the monitoring agent.

Table 12. Trace log files located on the Network Manager server (continued)

File name and path	Description
The agent operations log files are as follows: *instance_hostname_* *NP*.LG0 is the current log created when the agent is started. *instance_hostname_* *NP*.LG1 is the backup of the previous log. These logs are in the following directory depending on the operating system that you are using: • Linux **Linux:** *install_dir*/logs • **UNIX:** *install_dir*/logs	Shows whether the agent could connect to the monitoring server. Shows which situations are started and stopped, and shows other events while the agent is running. A new version of this file is generated every time the agent is restarted. One backup copy of the *.LG0 file is generated with the tag .LG1. View the .LG1 tag to learn the following details regarding the *previous* monitoring session: • Status of connectivity with the monitoring server • Situations that were running • The success or failure status of Take Action commands
The Take Action command log files are as follows: • *host*_np_*instance_takeactioncommand*.log The logs are in the following directories: • **UNIX:** *install_dir*/logs • Linux **Linux:** *install_dir*/logs	Traces activity each time a Take Action command runs. For example, when a hypothetical **start_command** Take Action command runs, a start_command.log file is generated.
The Take Action command log files are as follows: • knp_data_provider_actions_*instance_n*.log The logs are in the following directories: • **UNIX**: *install_dir*/logs • Linux **Linux**: *install_dir*/logs	Traces activity each time a Take Action command runs. All predefined Take Action commands are logged into this file.
The Take Action log files are as follows: • knp_*instance_takeactioncommand*.log • knp_*instance*_control_*domainscheduler*.log	Traces activity each time a Take Action command runs. For example, when a **stop_scheduler** Take Action command runs, the following log messages are generated: • kp9_*instance*_stop_log This log contains some basic information about how the Take Action command subsequently invoked a process called Control Scheduler. • kp9_*instance*_control_scheduler.log This log contains the log output of all of the PeopleSoft psadmin CLI output collected when executing specific Take Actions commands.

Table 12. Trace log files located on the Network Manager server (continued)	
File name and path	**Description**
Data provider log files: • `knu_data_provider_instance_startup.log` • `knu_data_provider_instance_n.log` The logs are in the following directories: • **Windows:** *install_dir*`\tmaitm6\logs` • **Linux:** *install_dir*`/logs`	Traces monitoring agent data provider state and operations.

Definitions of variables:

- *timestamp* is a time stamp with a format that includes year (y), month (m), day (d), hour (h), and minute (m), as follows: **yyyymmdd hhmm**
- *HEXtimestamp* is a hexadecimal representation of the time at which the process was started.
- *install_dir* represents the directory path where you installed the relevant component. *install_dir* can represent a path on the computer that hosts the monitoring system, the monitoring agent, or the portal.
- *instance* refers to the name of the database instance that you are monitoring.
- *instance_name* refers to the name of the agent instance.
- *hostname* refers to the name of the computer on which the relevant component runs.
- *nn* represents the circular sequence in which logs are rotated. This value includes a range from 1 - 5, by default. The first is always retained because it includes configuration parameters.

For more information about the complete set of trace logs that are maintained on the monitoring server, see the *IBM Tivoli Monitoring Installation and Setup Guide*.

Examples: Using trace logs

You can open trace logs in a text editor to learn some basic facts about your IBM Tivoli Monitoring environment.

The following examples are from the Tivoli Enterprise Monitoring Server log.

Example one

This excerpt shows the typical log for a failed connection between a monitoring agent and a monitoring server with the host name **server1a**:

```
(Thursday, August 11, 2005, 08:21:30-{94C}kdcl0cl.c,105,"KDCL0_ClientLookup")
 status=1c020006, "location server unavailable", ncs/KDC1_STC_SERVER
_UNAVAILABLE (Thursday, August 11, 2005, 08:21:35-{94C}kraarreg.cpp,
1157,"LookupProxy") Unable to connect to broker at ip.pipe:: status=0,
 "success", ncs/KDC1_STC_OK (Thursday, August 11, 2005, 08:21:35-{94C}
kraarreg.cpp,1402,"FindProxyUsingLocalLookup") Unable to find running
 CMS on CT_CMSLIST <IP.PIPE:#server1a>
```

Example two

The following excerpts from the trace log *for the monitoring server* show the status of an agent, identified here as "Remote node." The name of the computer where the agent is running is **SERVER5B**:

```
(42C039F9.0000-6A4:kpxreqhb.cpp,649,"HeartbeatInserter")
 Remote node SERVER5B:NP is ON-LINE.
. . .
(42C3079B.0000-6A4:kpxreqhb.cpp,644,"HeartbeatInserter")
 Remote node SERVER5B:NP is OFF-LINE.
```

See the following key points about the preceding excerpts:

- The monitoring server appends the two-character product code to the server name to form a unique name (for example, SERVER5B:vm) for this instance of the agent. By using this unique name, you can distinguish multiple monitoring products that might be running on **SERVER5B**.
- The log shows when the agent started (ON-LINE) and later stopped (OFF-LINE) in the environment.
- For the sake of brevity, an ellipsis (...) represents the series of trace log entries that were generated while the agent was running.
- Between the ON-LINE and OFF-LINE log entries, the agent was communicating with the monitoring server.
- The ON-LINE and OFF-LINE log entries are always available in the trace log.

RAS trace parameters

Pinpoint a problem by setting detailed tracing of individual components of the monitoring agent and modules

See "Overview of log file management" on page 83 to ensure that you understand log rolling and can reference the correct log files when you manage log file generation.

Setting RAS trace parameters by using the GUI

On Windows systems, you can use the graphical user interface to set trace options.

About this task

The default RAS1 trace level is ERROR, but you can change this. To change the RAS1 trace settings, complete the following steps:

Procedure

1. Open the **Manage Tivoli Enterprise Monitoring Services** window.
2. Select **Advanced** > **Edit Trace Parms**. The **Tivoli Enterprise Monitoring Server Trace Parameters** window is displayed.
3. Select a new trace setting in the pull-down menu in the **Enter RAS1 Filters** field or type a valid string.
   - General error tracing. KBB_RAS1=ERROR
   - Intensive error tracing. KBB_RAS1=ERROR (UNIT:knp ALL)
   - Maximum error tracing. KBB_RAS1=ERROR (UNIT:knp ALL) (UNIT:kra ALL)

   Note: As this example shows, you can set multiple RAS tracing options in a single statement.
4. Modify the value for Maximum Log Size Per File (MB) to change the log file size (changes LIMIT value).
5. Modify the value for Maximum Number of Log Files Per Session to change the number of log files per startup of a program (changes COUNT value).
6. Modify the value for Maximum Number of Log Files Total to change the number of log files for all startups of a program (changes MAXFILES value).
7. Optional: Click Y (Yes) in the **KDC_DEBUG Setting** menu to log information that can help you diagnose communications and connectivity problems between the monitoring agent and the monitoring server. The **KDC_DEBUG** setting and the **Maximum error tracing** setting can generate a large amount of trace logging. Use these settings only temporarily, while you are troubleshooting problems. Otherwise, the logs can occupy excessive amounts of hard disk space.
8. Click **OK**. You see a message reporting a restart of the monitoring agent so that your changes take effect.

What to do next

Monitor the size of the `logs` directory. Default behavior can generate a total of 45 - 60 MB for each agent that is running on a computer. For example, each database instance that you monitor can generate 45 - 60 MB of log data.

Regularly prune log files other than the RAS1 log files in the `logs` directory. Unlike the RAS1 log files that are pruned automatically, other log types can grow indefinitely, for example, the logs that include a process ID number (PID).

Use collector trace logs as an additional source of troubleshooting information.

Note: The **KDC_DEBUG** setting and the **Maximum error tracing** setting can generate a large amount of trace logging. Use these settings only temporarily while you are troubleshooting problems. Otherwise, the logs can occupy excessive amounts of hard disk space.

Manually setting RAS trace parameters

You can manually edit the RAS1 trace logging parameters.

About this task

The default RAS1 trace level is ERROR. To change the trace settings, complete the following steps.

Procedure

1. Open the trace options file:

   install_dir `/config/np.config`
2. Edit the line that begins with **KBB_RAS1=** to set trace logging preferences. For example, if you want detailed trace logging, set the **Maximum Tracing** option: `KBB_RAS1=ERROR (UNIT:knp ALL) (UNIT:kra ALL)`
3. Edit the line that begins with **KBB_RAS1_LOG=** to manage the generation of log files:
   - **MAXFILES**: The total number of files that are to be kept for all startups of a specific program. When this value is exceeded, the oldest log files are discarded. The default value is 9.
   - **LIMIT**: The maximum size, in megabytes (MB) of a RAS1 log file. The default value is 5.
   - IBM Software Support might guide you to modify the following parameters:
     - **COUNT**: The number of log files to keep in the rolling cycle of one program startup. The default is 3.
     - **PRESERVE**: The number of files that are not to be reused in the rolling cycle of one program startup. The default value is 1.

     Note: The **KBB_RAS1_LOG** parameter also provides for the specification of the log file directory, log file name, and the inventory control file directory and name. Do not modify these values or log information can be lost.
4. Restart the monitoring agent so that your changes take effect.

What to do next

Monitor the size of the `logs` directory. Default behavior can generate a total of 45 - 60 MB for each agent that is running on a computer. For example, each database instance that you monitor can generate 45 - 60 MB of log data.

Regularly prune log files other than the RAS1 log files in the `logs` directory. Unlike the RAS1 log files that are pruned automatically, other log types can grow indefinitely, for example, the logs that include a process ID number (PID).

Use collector trace logs as an additional source of troubleshooting information.

Note: The **KDC_DEBUG** setting and the **Maximum error tracing** setting can generate a large amount of trace logging. Use these settings only temporarily while you are troubleshooting problems. Otherwise, the logs can occupy excessive amounts of hard disk space.

Problems and workarounds

The known problems and workarounds are organized into types of problems that might occur with an agent, for example installation and configuration problems and workspace problems.

You can resolve some problems by ensuring that your system matches the requirements for installing the monitoring agent. See *Requirements for installing the monitoring agent* in the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide* for more information.

Installation and configuration troubleshooting

Problems can occur during installation, configuration, and uninstallation of the agent.

The following tables provide information about these problems and solutions.

Table 13. Problems and solutions for installation and configuration

Problem	Solution
(*UNIX only*) During a command-line installation, you choose to install a component that is currently installed, and you see the following warning: `WARNING - you are about to install the SAME version of "component_name"` where *component_name* is the name of the component that you are attempting to install. This problem affects UNIX command-line installations.	You must exit and restart the installation process. You cannot return to the list where you selected components to install. When you run the installer again, do not attempt to install any component that is currently installed.
A problem can occur when you install and configure a new monitoring agent on a computer where other agents are running as described in this example: • Agents are running on a computer and communicating with a Tivoli Enterprise Monitoring Server, called **TEMS1**. • You install a new agent on the same computer and you want this agent to communicate with a different monitoring server, called **TEMS2**. • When you configure the new agent to communicate with **TEMS2**, all the existing agents are reconfigured to communicate with **TEMS2**.	You must reconfigure the previously existing agents to restore their communication connection with **TEMS1**. For example, you can right-click the row for a specific agent in the Manage Tivoli Enterprise Monitoring Services, and select **Reconfigure**. For more information about reconfiguration, see the *IBM Tivoli Monitoring Installation and Setup Guide*.
A message similar to "`Unable to find running CMS on CT_CMSLIST`" in the log file is displayed.	If a message similar to "`Unable to find running CMS on CT_CMSLIST`" is displayed in the log file, the agent cannot connect to the monitoring server. Confirm the following points: • Do multiple network interface cards (NICs) exist on the system? • If multiple NICs exist on the system, find out which one is configured for the monitoring server. Ensure that you specify the correct host name and port settings for communication in the IBM Tivoli Monitoring environment.

Table 13. Problems and solutions for installation and configuration (continued)	
Problem	**Solution**
The system is experiencing high CPU usage.	**Agent process:** View the memory usage of the KNPCMA process. If CPU usage seems to be excessive, restart the monitoring agent. **Network cards:** The network card configurations can decrease the performance of a system. Each stream of packets that a network card receives (assuming that it is a broadcast or destined for the under-performing system) must generate a CPU interrupt and transfer the data through the I/O bus. If the network card in question is a bus-mastering card, work can be offloaded and a data transfer between memory and the network card can continue without using CPU processing power. Bus-mastering cards are 32-bit and are based on PCI or EISA bus architectures.

Table 14. General problems and solutions for uninstallation	
Problem	**Solution**
On Windows systems, uninstallation of IBM Tivoli Monitoring fails to uninstall the entire environment.	Be sure that you follow the general uninstallation process described in the *IBM Tivoli Monitoring Installation and Setup Guide*: 1. Remove Tivoli Enterprise Monitoring Server Application support by completing the following steps: a. Use Manage Tivoli Enterprise Monitoring Services. b. Select **Tivoli Enterprise Monitoring Server**. c. Right-click and select **Advanced**. d. Select **Remove TEMS application support**. e. Select the agent to remove its application support. 2. Uninstall the monitoring agents first, as in the following examples: • Uninstall a single monitoring agent for a specific database. -OR- • Uninstall all instances of a monitoring product, such as IBM Tivoli Monitoring for Databases. 3. Uninstall IBM Tivoli Monitoring.

Table 14. General problems and solutions for uninstallation (continued)	
Problem	**Solution**
The way to remove inactive managed systems (systems whose status is OFFLINE) from the Navigator tree in the portal is not obvious.	Use the following steps to remove, but not uninstall, an offline managed system from the Navigator tree: 1. Click the **Enterprise** icon in the Navigator tree. 2. Right-click, and then click **Workspace** > **Managed System Status**. 3. Right-click the offline managed system, and select **Clear offline entry**. To uninstall the monitoring agent, use the procedure described in the *IBM Tivoli Monitoring Installation and Setup Guide*.
After the remote removal from the Tivoli Enterprise Portal of a running instance, the instance name is still listed in the Start List.	Bring up the configure list to remove the instance name from the Start list.

Table 14. General problems and solutions for uninstallation (continued)	
Problem	**Solution**
IBM Tivoli Monitoring might not be able to generate a unique name for monitoring components because of the truncation of names that the product automatically generates.	If the agent supports multiple instances, IBM Tivoli Monitoring automatically creates a name for each monitoring component by concatenating the subsystem name, host name, and product code separated by colons (*subsystem_name*:*hostname*:KNP). **Note:** When you monitor a multinode system, such as a database, IBM Tivoli Monitoring adds a subsystem name to the concatenated name, typically a database instance name. The length of the name that IBM Tivoli Monitoring generates is limited to 32 characters. Truncation can result in multiple components having the same 32-character name. If this problem happens, shorten the *hostname* portion of the name as follows: 1. Open the configuration file for the monitoring agent, which is located in the following path: • **On UNIX and Linux:** `itm_home/config/`*`product_code`*`.ini` and *`product_code`*`.config`. For example, the file names for the Monitoring Agent for UNIX OS is `ux.ini` and `ux.config`. 2. Find the line that begins with `CTIRA_HOSTNAME=`. 3. Type a new name for host name that is a unique, shorter name for the host computer. The final concatenated name including the subsystem name, new host name, and KNP, cannot be longer than 32 characters. **Note:** You must ensure that the resulting name is unique with respect to any existing monitoring component that was previously registered with the Tivoli Enterprise Monitoring Server. 4. Save the file. 5. Restart the agent.

Remote deployment troubleshooting

Problems can occur with remote deployment and removal of agent software using the Agent Remote Deploy process.

The following table contains problems and solutions related to remote deployment.

Table 15. Remote deployment problems and solutions	
Problem	**Solution**
While you are using the remote deployment feature to install the monitoring agent, an empty command window is displayed on the target computer. This problem occurs when the target of remote deployment is a Windows computer. (For more information about the remote deployment feature, see the *IBM Tivoli Monitoring Installation and Setup Guide*.)	Do not close or modify this window. It is part of the installation process and is dismissed automatically.
The removal of a monitoring agent fails when you use the remote removal process in the Tivoli Enterprise Portal desktop or browser.	This problem might occur when you attempt the remote removal process immediately after you restart the Tivoli Enterprise Monitoring Server. You must allow time for the monitoring agent to refresh its connection with the Tivoli Enterprise Monitoring Server before you begin the remote removal process.

Troubleshooting workspaces

Problems can occur with general workspaces and agent-specific workspaces.

The following table contains problems and solutions related to workspaces.

Table 16. Workspace problems and solutions

Problem	Solution
The process application components are available, but the Availability status shows PROCESS_DATA_NOT_ AVAILABLE.	This problem occurs because the PerfProc performance object is disabled. When this condition exists, IBM Tivoli Monitoring cannot collect performance data for this process. Use the following steps to confirm that this problem exists and to resolve it: 1. In the Windows **Start** menu, click **Run**. 2. Type `perfmon.exe` in the **Open** field of the **Run** window. The **Performance** window is displayed. 3. Click the plus sign (+) in the toolbar. The **Add Counters** window is displayed. 4. Look for **Process** in the **Performance object** menu. 5. Complete one of the following actions: • If you see **Process** in the menu, the PerfProc performance object is enabled and the problem is coming from a different source. You might need to contact IBM Software Support. • If you do not see **Process** in the menu, use the Microsoft utility from the Microsoft.com Operations website to enable the PerfProc performance object. The **Process** performance object becomes visible in the **Performance object** menu of the **Add Counters** windows, and IBM Tivoli Monitoring is able to detect Availability data. 6. Restart the monitoring agent.
The name of the attribute does not display in a bar chart or graph view.	When a chart or graph view that includes the attribute is scaled to a small size, a blank space is displayed instead of a truncated name. To see the name of the attribute, expand the view of the chart until sufficient space is available to display all characters of the attribute name.
At the end of each view, you see the following Historical workspace KFWITM220E error: `Request failed during execution`.	Ensure that you configure all groups that supply data to the view. In the Historical Configuration view, ensure that data collection is started for all groups that supply data to the view.
When you use a long process name in the situation, the process name is truncated.	Truncation of process or service names for situations in the Availability table in the portal display is the expected behavior. The maximum name length is 100 bytes.
Regular (non-historical) monitoring data fails to be displayed.	Check the formation of the queries you use to gather data. For example, look for invalid SQL statements.

Table 16. Workspace problems and solutions (continued)	
Problem	**Solution**
No row of data for 64-bit applications is displayed in the workspaces when the monitoring agent is running on a 64-bit operating system.	The Tivoli Enterprise Portal shows data only for 32-bit applications. No solution is available for this problem at this time.

Troubleshooting situations

Problems can occur with situations and situation configuration.

The following table contains problems and solutions for situations.

Table 17. Situation problems and solutions	
Problem	**Solution**
Monitoring activity requires too much disk space.	Check the RAS trace logging settings. For example, trace logs grow rapidly when you apply the `ALL` logging option.
A formula that uses mathematical operators appears to be incorrect. For example, if you were monitoring a Linux system, the formula that calculates when **Free Memory** falls under 10 percent of **Total Memory** does not work: `LT #'Linux_VM_Stats.Total_Memory' / 10`	This formula is incorrect because situation predicates support only logical operators. Your formulas cannot have mathematical operators. **Note:** The Situation Editor provides alternatives to math operators. In the example, you can select the **% Memory Free** attribute and avoid the need for math operators.
You want to change the appearance of situations when they are displayed in the navigation tree.	1. Right-click an item in the navigation tree. 2. Click **Situations** in the menu. The **Situation Editor** window is displayed. 3. Select the situation that you want to modify. 4. Use the **State** menu to set the status and appearance of the Situation when it triggers. **Note:** The **State** setting is not related to severity settings in the IBM Tivoli Enterprise Console.
When a situation is triggered in the Event Log attribute group, it remains in the Situation Event Console as long as the event ID entry is present in the Event Log workspace. When this event ID entry is removed from the Event Log workspace on the Tivoli Enterprise Portal, the situation is also cleared even if the actual problem that caused the event is not resolved, and the event ID entry is also present in the Windows Event Viewer.	A timeout occurs on the cache of events for the NT Event Log group. Increase the cache time of Event Log collection to meet your requirements by adding the following variable and timeout value to the K*pc*ENV file for the agent (where *pc* is the two-letter product code): `CDP_NT_EVENT_LOG_CACHE_TIMEOUT=3600` This variable determines how long events from the NT Event Log are kept.
The situation for a specific agent is not visible in the Tivoli Enterprise Portal.	Open the Situation Editor. Access the All managed servers view. If the situation is not displayed, confirm that the monitoring server has been seeded for the agent.
The monitoring interval is too long.	Access the Situation Editor view for the situation that you want to modify. Check the **Sampling interval** area in the **Formula** tab. Adjust the time interval as required.

Table 17. Situation problems and solutions (continued)

Problem	Solution
The situation did not activate at startup.	Manually recycle the situation as follows: 1. Right-click the situation and select **Stop Situation**. 2. Right-click the situation and select **Start Situation**. **Note:** You can permanently avoid this problem by selecting the **Run at Startup** check box of the Situation Editor view for a specific situation.
The situation is not displayed.	Click the **Action** tab and check whether the situation has an automated corrective action. This action can occur directly or through a policy. The situation might be resolving so quickly that you do not see the event or the update in the graphical user interface.
An Alert event did not occur even though the predicate was correctly specified.	Check the logs, reports, and workspaces.
A situation fires on an unexpected managed object.	Confirm that you distributed and started the situation on the correct managed system.
The product did not distribute the situation to a managed system.	Click the **Distribution** tab and check the distribution settings for the situation.

Table 17. Situation problems and solutions (continued)

Problem	Solution
The situation does not fire.	This problem can be caused when incorrect predicates are present in the formula that defines the situation. For example, the managed object shows a state that normally triggers a monitoring event, but the situation is not true because the wrong attribute is specified in the formula. In the **Formula** tab, analyze predicates as follows: 1. Click the **fx** icon in the **Formula** area. The **Show formula** window is displayed. a. Confirm the following details in the **Formula** area of the window: • The attributes that you intend to monitor are specified in the formula. • The situations that you intend to monitor are specified in the formula. • The logical operators in the formula match your monitoring goal. • The numeric values in the formula match your monitoring goal. b. (Optional) Select the **Show detailed formula** check box to see the original names of attributes in the application or operating system that you are monitoring. c. Click **OK** to dismiss the **Show formula** window. 2. (Optional) In the **Formula** area of the **Formula** tab, temporarily assign numeric values that immediately trigger a monitoring event. The triggering of the event confirms that other predicates in the formula are valid. **Note:** After you complete this test, you must restore the numeric values to valid levels so that you do not generate excessive monitoring data based on your temporary settings.
Situation events are not displayed in the Events Console view of the workspace.	Associate the situation with a Navigator item. **Note:** The situation does not need to be displayed in the workspace. It is sufficient that the situation is associated with any Navigator item.

Table 17. Situation problems and solutions (continued)	
Problem	**Solution**
You do not have access to a situation.	**Note:** You must have administrator privileges to complete these steps. 1. Click **Edit** > **Administer Users** to access the **Administer Users** window. 2. In the **Users** area, select the user whose privileges you want to modify. 3. In the **Permissions** tab, **Applications** tab, and **Navigator Views** tab, select the permissions or privileges that correspond to the user role. 4. Click **OK**.

Troubleshooting Take Action commands

Problems can occur with Take Action commands.

The following table contains problems and solutions that can occur with Take Action commands.

Table 18. Take Action commands problems and solutions	
Problem	**Solution**
Take Action commands often require several minutes to complete.	Allow several minutes. If you do not see a message advising you of completion, try to run the command manually.
Situations fail to trigger Take Action commands.	Attempt to manually run the Take Action command in the Tivoli Enterprise Portal. If the Take Action command works, look for configuration problems in the situation.

ITM Agent reference

The IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition agent provides predefined workspaces, attributes, and situations. It can be extended with Take Action commands and policies.

Workspaces reference

A workspace is the working area of the Tivoli Enterprise Portal application window. The Navigator contains a list of the workspaces provided by the agent.

About workspaces

Use the Navigator to select the workspace you want to see. As part of the application window, the status bar shows the Tivoli Enterprise Portal Server name and port number to which the displayed information applies and the ID of the current user.

When you select an item in the Navigator, a default workspace is displayed. When you right-click a navigator item, a menu that includes a Workspace item is displayed. The Workspace item contains a list of workspaces for that navigator item. Each workspace has at least one view. Some views have links to other workspaces. You can also use the Workspace Gallery tool as described in the *Tivoli Enterprise Portal User's Guide* to open workspaces.

The workspaces in the Navigator are displayed in a Physical view that shows your enterprise as a physical mapping or a dynamically populated logical view that is agent-specific. You can also create a Logical view. The Physical view is the default view.

This monitoring agent provides predefined workspaces. You cannot modify or delete the predefined workspaces, but you can create new workspaces by editing them and saving the changes with a different name.

The IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition provides various default workspaces. These workspaces are displayed in the Navigator under the following nodes and subnodes for this monitoring agent:

IBM Tivoli Monitoring for Tivoli Network Manager
Corresponds to an agent instance and contains agent instance-level workspaces.

When a single instance of the monitoring agent is defined on a system, the top-level node is IBM Tivoli Monitoring for Tivoli Network Manager - *Instance*:*Hostname*:NP. The IBM Tivoli Monitoring for Tivoli Network Manager workspace is defined at this node. When multiple instances of the monitoring agent are defined on a system, the top-level node becomes IBM Tivoli Monitoring for Tivoli Network Manager. The IBM Tivoli Monitoring for Tivoli Network Manager workspace is undefined at this node. A node for each instance is created called *Instance*:*Hostname*:NP. A workspace that is called *Instance*:*Hostname*:NP is associated with the instance node. This workspace is comparable to the IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition workspace.

Workspace views can be any combination of query-based views, event views, and special purpose views.

Additional information about workspaces

For more information about creating, customizing, and working with workspaces, see "Using workspaces" in the *Tivoli Enterprise Portal User's Guide*.

Some attribute groups for this monitoring agent might not be represented in the predefined workspaces or views for this agent.

Predefined workspaces

The IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition agent provides predefined workspaces, which are organized by navigator item.

Agent-level navigator items

- IBM Tivoli Monitoring for Tivoli Network Manager navigator item
  - IBM Tivoli Monitoring for Tivoli Network Manager workspace
- Availability navigator item
  - Availability workspace.
- Discovery navigator item
  - Discovery workspace.
- Network navigator item
  - Network workspace.

Workspace descriptions

Each workspace description provides information about the workspace such as the purpose and a list of views in the workspace.

Workspaces are listed under navigator items.

IBM Tivoli Monitoring for Tivoli Network Manager navigator item
The workspace descriptions are organized by the navigator item to which the workspaces are relevant.

IBM Tivoli Monitoring for Tivoli Network Manager workspace
This workspace displays the overall state of the Tivoli Network Manager application.

This workspace contains the following views:

Availability
Displays the state of the Tivoli Network Manager processes.

Discovery
Displays the state of the Tivoli Network Manager Discovery.

Network
Displays the Tivoli Network Manager Network metrics.

Availability navigator item
The workspace descriptions are organized by the navigator item to which the workspaces are relevant.

Availability workspace
The Availability workspace displays the overall health of the application.

This workspace contains the following views:

Availability
Displays the state of each component in the application. Each process is displayed using a descriptive name, the name of the running process, and the state of the process (UP, DOWN, or PROCESS_DATA_NOT_AVAILABLE). When the state of the component is DOWN (for a process, or service) it is highlighted with a red background.

Processor
Displays the amount of CPU used by each process that is a component of the application. This displays the 2 main components of CPU usage, privileged time which is time spent in the kernel on behalf of the process and user mode time, which is the time spent running the process code.

Threads
Displays the number of threads used by each process that is a component of the application.

Memory
Displays the amount of memory being consumed by each process that is a component of the application. This total (virtual) size of the process and the size of the process in memory (working set) are displayed.

Discovery navigator item
The workspace descriptions are organized by the navigator item to which the workspaces are relevant.

Discovery workspace
The Discovery workspace displays the health of the network discovery done by Tivoli Network Manager and gathers the metrics used to generate reports.

This workspace contains the following views:

Current_Discovery
This view displays a snapshot of the current discovery. The values are all set to zero, no discovery is available at the moment.

Last Discovery Phase Duration
Displays a snapshot of the duration of each phase of the last completed discovery.

Last Discovery Memory Usage in KBytes
Displays a snapshot of the memory usage to process the last completed discovery.

Objects Discovered
Displays a snapshot of the objects discovered during the last completed discovery.

Agent Status
Displays a snapshot of the status of each agent configured in the system.

Monitoring navigator item
The workspace descriptions are organized by the navigator item to which the workspaces are relevant.

Monitoring workspace
This workspace displays information related to Tivoli Network Manager monitoring. The workspace gathers network monitoring metrics used to monitor the Tivoli Network Manager pollers and to generate reports.

This workspace contains the following views:

Displays the total number of MIB objects retrieved per second.

Number of Devices Being Polled by Poll Definition Type
Displays the total number of devices being polled by poll definition type.

Packets Sent by Poller
Displays the total number of packets sent by the poller.

Packets Processed by Poller
Displays the total number of packets processed by the poller.

SNMP Errors and Timeout Responses
Displays the total number of SNMP errors and timeouts.

Network navigator item
The workspace descriptions are organized by the navigator item to which the workspaces are relevant.

Network workspace
This workspace displays information related to a network monitored by Tivoli Network Manager. The workspace gathers network discovery metrics used to monitor the network and to generate reports.

This workspace contains the following views:

Network elements
Displays the total number of network elements listed by type in a monitored network.

Devices with and without SNMP access
Displays the total number of devices with and without SNMP access.

Interfaces Up and Down
Displays a counter of the interface Operational status that can be UP or DOWN.

Attributes reference

Attributes are the application properties that are being measured and reported by the agent.

About attributes

Attributes are organized into attribute groups. Attributes in an attribute group relate to a single object such as an application, or to a single kind of data such as status information.

Attributes in a group can be used in queries, query-based views, situations, policy workflows, take action definitions, and launch application definitions. Chart or table views and situations are two examples of how attributes in a group can be used:

- Chart or table views

  Attributes are displayed in chart and table views. The chart and table views use queries to specify which attribute values to request from a monitoring agent. You use the Properties editor to apply filters and set styles to define the content and appearance of a view based on an existing query.
- Situations

  You use attributes to create situations that monitor the state of your operating system, database, or application. A situation describes a condition you want to test. When you start a situation, the values you assign to the situation attributes are compared with the values collected by the agent and registers

an *event* if the condition is met. You are alerted to events by indicator icons that are displayed in the Navigator.

Additional information about attributes

For more information about using attributes and attribute groups, see the *Tivoli Enterprise Portal User's Guide*.

Attribute groups for the monitoring agent

The agent contains the following attribute groups.

- Attribute group name: Agent Status
  - Table name: KNPAGTSTS
- Attribute group name: Availability
  - Table name: KNPAVAIL
- Attribute group name: Current Discovery
  - Table name: KNPCURDISC
- Attribute group name: Devices Being Polled
  - Table name: KNPDEVPOLL
- Attribute group name: Devices With SNMP Access
  - Table name: KNPSNMPAC
- Attribute group name: Entities On Model DB
  - Table name: KNPTOTENT
- Attribute group name: Interfaces Up and Down
  - Table name: KNPNODETO
- Attribute group name: Last Discovery
  - Table name: KNPLSTDISC
- Attribute group name: MIB Objects Retrieved
  - Table name: KNPMIBOBJ
- Attribute group name: Network Elements
  - Table name: KNPNETELEM
- Attribute group name: Objects Discovered
  - Table name: KNPOBJDISC
- Attribute group name: Packets Sent And Processed By Poller
  - Table name: KNPPACPROC
- Attribute group name: Performance Object Status
  - Table name: KNPPOBJST
- Attribute group name: Polling Capacity
  - Table name: KNPCAPPACT
- Attribute group name: SNMP Errors And Timeouts
  - Table name: KNPSNMPERR
- Attribute group name: Work Item Queue
  - Table name: KNPWORKQUE

Attributes in each attribute group

Attributes in each attribute group collect data that the agent uses for monitoring.

Agent Status attribute group

This view displays a snapshot of the status of each agent configured in the system.

Attribute descriptions

The following list contains information about each attribute in the Agent Status attribute group:

Agent Connections attribute

Description

Displays the total number of agent connections.

Type

Integer (32-bit gauge)

Agent Name attribute This attribute is a key attribute.

Description

Displays the agent name configured in the system.

Type

String

Agent Status attribute

Description

Displays the status of the agents used by discovery.

Type

Integer with enumerated values. The following values are defined: Inactive state (0), Yet to be started (1), Being started (2), Receiving and sending data (3), Active but finished processing (4), Was active but has died (5). Any value that does not have a definition here is displayed in the User Interface

Node attribute This attribute is a key attribute.

Description

The managed system name of the agent.

Type

String

Timestamp attribute

Description

The local time at the agent when the data was collected.

Type

String

Availability attribute group

The Availability data set contains the availability data for all processes and services that make up this application.

Attribute descriptions

The following list contains information about each attribute in the Availability attribute group:

Application Component attribute This attribute is a key attribute.

Description

The descriptive name of a part of the application.

Type
string

Command Line attribute

Description
The program name and any arguments specified on the command line when the process was started. For Service or Functionality test, this attribute has the value N/A.

Type
String with enumerated values. The following values are defined: N/A (N/A). Any value that does not have a definition here is displayed in the User Interface

Full Name attribute

Description
The full name of the process that includes the path.

Type
string with enumerated values. The following values are defined: N/A (N/A). Any value that does not have a definition here is displayed in the User Interface

Functionality Test Message attribute

Description
The text message that corresponds to the Functionality Test Status. This attribute is only valid for functionality tests.

Type
String with enumerated values. The following values are defined: N/A (N/A). Any value that does not have a definition here is displayed in the User Interface

Functionality Test Status attribute

Description
The return code of the functionality test. When the monitored application is running correctly, 'SUCCESS' is displayed. 'NOT_RUNNING' is displayed when it is not running correctly. 'N/A' is displayed when the row does not represent a functionality test.

Type
Integer with enumerated values. The following values are defined: SUCCESS (0), N/A (1), GENERAL ERROR (2), WARNING (3), NOT RUNNING (4), DEPENDENT NOT RUNNING (5), ALREADY RUNNING (6), PREREQ NOT RUNNING (7), TIMED OUT (8), DOESNT EXIST (9), UNKNOWN (10), DEPENDENT STILL RUNNING (11), INSUFFICIENT USER AUTHORITY (12). Any value that does not have a definition here is displayed in the User Interface

Name attribute

Description
The name of the process, service, or functionality test. This name matches the executable name of the process, the service short name or the name of the process used to test the application.

Type
String with enumerated values. The following values are defined: N/A (N/A). Any value that does not have a definition here is displayed in the User Interface

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type

String

Page Faults per Sec attribute

Description
The rate of page faults for the process measured in faults per second. This attribute contains only valid data for processes.

Type
Integer (32-bit gauge)

Percent Privileged Time attribute

Description
The percentage of the available CPU time being used by this process for privileged operation.

Type
Integer (32-bit gauge)

Percent Processor Time attribute

Description
The percentage of the elapsed time that this process used the processor to execute instructions.

Type
Integer (32-bit gauge)

Percent User Mode Time attribute

Description
The percentage of the available CPU time being used by this process for user mode operation.

Type
Integer (32-bit gauge)

PID attribute

Description
The process ID associated with the process. This attribute contains only valid data for processes.

Type
Integer (32-bit gauge)

Status attribute

Description
The status of the application component.

- For processes 'UP', 'DOWN', 'WARNING', or 'PROCESS_DATA_NOT_AVAILABLE': 'PROCESS_DATA_NOT_AVAILABLE' is displayed for a process when the matching process is running but the resource use information cannot be collected for that process.
- For services 'UP', 'DOWN', or 'UNKNOWN': 'UNKNOWN' is displayed when the service is not installed.
- For functionality tests: 'PASSED' or 'FAILED' is displayed.

Type
Integer with enumerated values. The following values are defined: DOWN (0), UP (1), WARNING (2), UNKNOWN (3), PASSED (4), FAILED (5), PROCESS DATA NOT AVAILABLE (6). Any value that does not have a definition here is displayed in the User Interface

Thread Count attribute

Description
The number of threads currently allocated by this process. This attribute contains only valid data for processes.

Type
Integer (32-bit gauge)

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Type attribute

Description
The type of the application component. Components are processes, services, or functionality tests.

Type
Integer with enumerated values. The following values are defined: PROCESS (0), SERVICE (1), FUNCTIONALITY TEST (2). Any value that does not have a definition here is displayed in the User Interface

Virtual Size attribute

Description
The virtual size (in MB) of the process.

Type
Integer (32-bit gauge)

Working Set Size attribute

Description
The working set size of the process in MB. This attribute contains only valid data for processes.

Type
Integer (32-bit gauge)

Current Discovery attribute group
This view displays a snapshot of the current discovery. If there is no current discovery, all values are set to zero.

Attribute descriptions
The following list contains information about each attribute in the Current Discovery attribute group:

Blackout State attribute

Description
The blackout state

Type
Integer with enumerated values. The following values are defined: False (0), True (1). Any value that does not have a definition here is displayed in the User Interface

Cycle Count attribute

Description
The discovery cycle count

Type
Integer (32-bit gauge)

Discovery Mode attribute

Description
The discovery mode

Type
Integer with enumerated values. The following values are defined: Full (0), Partial (1). Any value that does not have a definition here is displayed in the User Interface

Discovery Phase attribute

Description
The discovery phase

Type
Integer with enumerated values. The following values are defined: Phase 0 (Not Running) (0), Phase 1 (1), Phase 2 (2), Phase 3 (3), Phase 4 (4). Any value that does not have a definition here is displayed in the User Interface

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Processing Needed attribute

Description
The discovery processing needed

Type
Integer with enumerated values. The following values are defined: No (0), Yes (1). Any value that does not have a definition here is displayed in the User Interface

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Devices Being Polled attribute group
This view displays the total number of devices being polled by poll definition type.

Attribute descriptions
The following list contains information about each attribute in the Devices Being Polled attribute group:

Addresses attribute

Description
Displays the total number of addresses being polled.

Type
Integer (32-bit gauge)

Entities attribute

Description
Displays the total number of entities being monitored

Type
Integer (32-bit gauge)

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Pollicy Name attribute

Description
Displays the policy name that is polling the devices.

Type
String

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Devices With SNMP Access attribute group
Displays the total number of devices with and without SNMP Access.

Attribute descriptions
The following list contains information about each attribute in the Devices With SNMP Access attribute group:

No SNMP Access attribute

Description
Displays the total number of devices without SNMP access.

Type
Integer (32-bit gauge)

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

SNMP Access attribute

Description
Displays the total number of devices with SNMP access.

Type
Integer (32-bit gauge)

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Entities On Model DB attribute group
Displays the total number of entities defined in the model database.

Attribute descriptions
The following list contains information about each attribute in the Entities On Model DB attribute group:

Entities attribute

Description
Displays the total number of entities defined in the model database.

Type
Integer (32-bit gauge)

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Interfaces Up and Down attribute group
Displays a counter of the Interfaces operational status, which can be UP and DOWN.

Attribute descriptions
The following list contains information about each attribute in the Interfaces Up and Down attribute group:

Down attribute

Description
Displays the total number of Interfaces Down

Type
Integer (32-bit gauge)

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Up attribute

Description
Displays the total number of Interfaces Up

Type
Integer (32-bit gauge)

Last Discovery attribute group
This view displays a snapshot of the duration and memory usage for each phase of the last completed discovery.

Attribute descriptions
The following list contains information about each attribute in the Last Discovery attribute group:

Completion memory attribute

Description
Displays the discovery completion memory.

Type
Integer (32-bit gauge)

Completion time attribute

Description
Displays the discovery completion time.

Type
String

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Phase one memory attribute

Description
Displays the phase one memory usage.

Type
Integer (32-bit gauge)

Phase one start attribute

Description
Displays the phase one start time.

Type
String

Phase three memory attribute

Description
Displays the phase three memory usage.

Type
Integer (32-bit gauge)

Phase three start attribute

Description
Displays the phase three start time.

Type
String

Phase two memory attribute

Description
Displays the phase two memory usage.

Type
Integer (32-bit gauge)

Phase two start attribute

Description
Displays the phase two start time.

Type
String

Processing start time attribute

Description
Displays the data processing start time.

Type
String

Starting memory attribute

Description
Displays the discovery starting memory size.

Type
Integer (32-bit gauge)

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Total discovery time attribute

Description
Displays the total time spent by discovery.

Type
String

MIB Objects Retrieved attribute group
Displays the total number of MIB objects retrieved per second.

Attribute descriptions
The following list contains information about each attribute in the MIB Objects Retrieved attribute group:

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Total attribute

Description
Displays the total number of MIB objects retrieved per second

Type
Integer (32-bit gauge)

Network Elements attribute group
This view displays the total number of network elements listed by type in a monitored network.

Attribute descriptions
The following list contains information about each attribute in the Network Elements attribute group:

Card attribute

Description
Displays the total number of Cards.

Type
Integer (32-bit gauge)

Chassis attribute

Description
Displays the total number of chassis.

Type
Integer (32-bit gauge)

Interfaces attribute

Description
Displays the total number of interfaces.

Type
Integer (32-bit gauge)

Logical Interface attribute

Description
Displays the total number of logical interfaces.

Type
integer (32-bit gauge)

Module attribute

Description
Displays the total number of Modules.

Type
Integer (32-bit gauge)

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

PSU attribute

Description
Displays the total number of PSU.

Type
Integer (32-bit gauge)

Subnet attribute

Description
Displays the total number of Subnets.

Type
Integer (32-bit gauge)

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Unknown attribute

Description
Displays the total number of Unknown.

Type
Integer (32-bit gauge)

VLAN Objects attribute

Description
Displays the total number of VLAN objects.

Type
Integer (32-bit gauge)

Objects Discovered attribute group
This view displays a snapshot of the objects discovered during the last completed discovery.

Attribute descriptions
The following list contains information about each attribute in the Objects Discovered attribute group:

Devices attribute

Description
Displays the total number of devices.

Type
Integer (32-bit gauge)

Interfaces attribute

Description
Displays the total number of interfaces.

Type
Integer (32-bit gauge)

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Routers attribute

Description
Displays the total number of routers.

Type
Integer (32-bit gauge)

Switches attribute

Description
Displays the total number of switches.

Type
Integer (32-bit gauge)

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Packets Sent And Processed By Poller attribute group
Displays the number of packets, per second, sent and processed by the poller.

Attribute descriptions
The following list contains information about each attribute in the Packets Sent And Processed By Poller attribute group:

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Processed attribute

Description
Displays the total number of packets, per second, processed by the poller.

Type
Integer (32-bit gauge)

Sent attribute

Description
Displays the total number of packets, per second, sent by the poller.

Type
Integer (32-bit gauge)

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Performance Object Status attribute group
The Performance Object Status data set contains information that reflects the status of other data sets so you can see the status of all performance objects that make up this application all at once. Each of these other performance data sets is represented by a row in this table (or other type of view). The status for a data set reflects the result of the last attempt to collect data for that data set, so you can see whether the agent is collecting data correctly. Unlike other data sets, the Performance Object Status data set does not reflect the state of the monitored application. This data set is most often used to determine why data is not available for one of the performance data sets.

Attribute descriptions
The following list contains information about each attribute in the Performance Object Status attribute group:

Average Collection Duration attribute

Description
The average duration of all data collections of this group in seconds.

Type
Real number (32-bit counter) with two decimal places of precision with enumerated values. The following values are defined: NO DATA (-100). Any value that does not have a definition here is displayed in the User Interface

Cache Hit Percent attribute

Description
The percentage of external data requests for this group that were satisfied from the cache.

Type
Real number (32-bit counter) with two decimal places of precision

Cache Hits attribute

Description
The number of times an external data request for this group was satisfied from the cache.

Type
Integer (32-bit counter)

Cache Misses attribute

Description
The number of times an external data request for this group was not available in the cache.

Type
Integer (32-bit counter)

Error Code attribute

Description
The error code associated with the query.

Type
Integer with enumerated values. The following values are defined: NO ERROR (0), GENERAL ERROR (1), OBJECT NOT FOUND (2), COUNTER NOT FOUND (3), NAMESPACE ERROR (4), OBJECT CURRENTLY UNAVAILABLE (5), COM LIBRARY INIT FAILURE (6), SECURITY INIT FAILURE (7), PROXY SECURITY FAILURE (9), NO INSTANCES RETURNED (10), ASSOCIATOR QUERY FAILED (11), REFERENCE QUERY FAILED (12), NO RESPONSE RECEIVED (13), CANNOT FIND JOINED QUERY (14), CANNOT FIND JOIN ATTRIBUTE IN QUERY 1 RESULTS (15), CANNOT FIND JOIN ATTRIBUTE IN QUERY 2 RESULTS (16), QUERY 1 NOT A SINGLETON (17), QUERY 2 NOT A SINGLETON (18), NO INSTANCES RETURNED IN QUERY 1 (19), NO INSTANCES RETURNED IN QUERY 2 (20), CANNOT FIND ROLLUP QUERY (21), CANNOT FIND ROLLUP ATTRIBUTE (22), FILE OFFLINE (23), NO HOSTNAME (24), MISSING LIBRARY (25), ATTRIBUTE COUNT MISMATCH (26), ATTRIBUTE NAME MISMATCH (27), COMMON DATA PROVIDER NOT STARTED (28), CALLBACK REGISTRATION ERROR (29), MDL LOAD ERROR (30), AUTHENTICATION FAILED (31), CANNOT RESOLVE HOST NAME (32), SUBNODE UNAVAILABLE (33), SUBNODE NOT FOUND IN CONFIG (34), ATTRIBUTE ERROR (35), CLASSPATH ERROR (36), CONNECTION FAILURE (37), FILTER SYNTAX ERROR (38), FILE NAME MISSING (39), SQL QUERY ERROR (40), SQL FILTER QUERY ERROR (41), SQL DB QUERY ERROR (42), SQL DB FILTER QUERY ERROR (43), PORT OPEN FAILED (44), ACCESS DENIED (45), TIMEOUT (46), NOT IMPLEMENTED (47), REQUESTED A BAD VALUE (48), RESPONSE TOO BIG (49), GENERAL RESPONSE ERROR (50), SCRIPT NONZERO RETURN (51), SCRIPT NOT FOUND (52), SCRIPT LAUNCH ERROR (53), CONF FILE DOES NOT EXIST (54), CONF FILE ACCESS DENIED (55), INVALID CONF FILE (56), EIF INITIALIZATION FAILED (57), CANNOT OPEN FORMAT FILE (58), FORMAT FILE SYNTAX ERROR (59), REMOTE HOST UNAVAILABLE (60), EVENT LOG DOES NOT EXIST (61), PING FILE DOES NOT EXIST (62), NO PING DEVICE FILES (63), PING DEVICE LIST FILE MISSING (64), SNMP MISSING PASSWORD (65), DISABLED (66), URLS FILE NOT FOUND (67), XML PARSE ERROR (68), NOT INITIALIZED (69), ICMP SOCKETS FAILED (70), DUPLICATE CONF FILE (71). Any value that does not have a definition here is displayed in the User Interface

Intervals Skipped attribute

Description
The number of times a background data collection for this group was skipped because the previous collection was still running when the next one was due to start.

Type
Integer (32-bit counter)

Last Collection Duration attribute

Description
The duration of the most recently completed data collection of this group in seconds.

Type
Real number (32-bit counter) with two decimal places of precision

Last Collection Finished attribute

Description
The most recent time a data collection of this group finished.

Type
Timestamp with enumerated values. The following values are defined: NOT COLLECTED (0691231190000000), NOT COLLECTED (0000000000000001). Any value that does not have a definition here is displayed in the User Interface

Last Collection Start attribute

Description
The most recent time a data collection of this group started.

Type
Timestamp with enumerated values. The following values are defined: NOT COLLECTED (0691231190000000), NOT COLLECTED (0000000000000001). Any value that does not have a definition here is displayed in the User Interface

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Number of Collections attribute

Description
The number of data collections for this group since the agent started.

Type
Integer (32-bit counter)

Object Name attribute

Description
The name of the performance object.

Type
String

Object Status attribute

Description
The status of the performance object.

Type
Integer with enumerated values. The following values are defined: ACTIVE (0), INACTIVE (1). Any value that does not have a definition here is displayed in the User Interface

Object Type attribute

Description
The type of the performance object.

Type

Integer with enumerated values. The following values are defined: WMI (0), PERFMON (1), WMI ASSOCIATION GROUP (2), JMX (3), SNMP (4), SHELL COMMAND (5), JOINED GROUPS (6), CIMOM (7), CUSTOM (8), ROLLUP DATA (9), WMI REMOTE DATA (10), LOG FILE (11), JDBC (12), CONFIG DISCOVERY (13), NT EVENT LOG (14), FILTER (15), SNMP EVENT (16), PING (17), DIRECTOR DATA (18), DIRECTOR EVENT (19), SSH REMOTE SHELL COMMAND (20). Any value that does not have a definition here is displayed in the User Interface.

Query Name attribute This attribute is a key attribute.

Description
The name of the attribute group.

Type
String

Refresh Interval attribute

Description
The interval at which this group is refreshed in seconds.

Type
Integer (32-bit counter)

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Polling Capacity attribute group
Displays the monitoring polling capacity.

Attribute descriptions
The following list contains information about each attribute in the Polling Capacity attribute group:

Node attribute This attribute is a key attribute.

Description
The managed system name of the agent.

Type
String

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Total attribute

Description
Displays the total number of threads available for use.

Type
Integer (32-bit gauge)

SNMP Errors And Timeouts attribute group
Displays the total number of SNMP error and timeout responses.

Attribute descriptions
The following list contains information about each attribute in the SNMP Errors And Timeouts attribute group:

Errors attribute

Description
Displays the total number of SNMP Errors per second.

Type
Integer (32-bit gauge)

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Timeouts attribute

Description
Displays the total number of SNMP timeouts per second.

Type
Integer (32-bit gauge)

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Work Item Queue attribute group
Displays the number of work items in the queue to be processed.

Attribute descriptions
The following list contains information about each attribute in the Work Item Queue attribute group:

Node attribute **This attribute is a key attribute.**

Description
The managed system name of the agent.

Type
String

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Total attribute

Description
Displays the total number of work items in the queue.

Type
Integer (32-bit gauge)

Situations reference

A situation is a logical expression involving one or more system conditions. Situations are used to monitor the condition of systems in your network. You can manage situations from the Tivoli Enterprise Portal by using the Situation Editor or from the command-line interface using the tacmd commands for situations. You can manage private situations in the private configuration XML file.

About situations

The monitoring agents that you use to monitor your system environment include a set of predefined situations that you can use as-is. You can also create new situations to meet your requirements.

Predefined situations contain attributes that check for system conditions common to many enterprises. Using predefined situations can improve the speed with which you can begin using the IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition. You can change the conditions or values being monitored by a predefined situation to the conditions or values best suited to your enterprise.

You can display predefined situations and create your own situations using the Situation editor. The Situation editor initially lists the situations associated with the navigator item that you selected. When you click a situation name or create a situation, a panel opens with the following tabs:

Formula
Formula describing the condition being tested.

Distribution
List of managed systems (operating systems, subsystems, or applications) to which the situation can be distributed. All the managed systems are assigned by default.

Expert advice
Comments and instructions to be read in the event workspace.

Action
Command to be sent to the system.

EIF
Customize forwarding of the event to an Event Integration Facility receiver. (Available when the Tivoli Enterprise Monitoring Server is configured to forward events.)

Until
Options to close the event after a period of time, or when another situation becomes true.

Additional information about situations

The *Tivoli Enterprise Portal User's Guide* contains more information about predefined and custom situations and how to use them to respond to alerts.

Predefined situations

The monitoring agent contains predefined situations, which are organized by Navigator item.

Agent level Navigator items

- IBM Tivoli Monitoring for Tivoli Network Manager
  - Not applicable
- Availability
  - Not applicable
  - KNP_NCPMONITOR_Process_Down

  - KNP_NCP_CONFIG_Process_Down
  - KNP_NCP_D_HELPSRV_Process_Down
  - KNP_NCP_G_EVENT_Process_Down
  - KNP_NCP_MODEL_Process_Down
  - KNP_NCP_POLLER_Process_Down
  - KNP_NCP_STORE_Process_Down
  - KNP_NCP_VIRTUAL_Process_Down
  - KNP_NCP_WEBTOOL_Process_Down
  - KNP_Process_CPU_Critical
  - KNP_Process_CPU_High
- Discovery
  - Not applicable
- Monitoring
  - Not applicable
  - KNP_Polling_Capacitiy_Critical
  - KNP_Polling_Capacitiy_Warning
  - KNP_Work_Item_Queue_Warning
- Network
  - Not applicable
  - KNP_Total_Entities_Critical
  - KNP_Total_Entities_Warning

Situation descriptions

Each situation description provides information about the situation that you can use to monitor the condition of systems in your network.

The situation descriptions provide the following information:

Description
Information about the conditions that the situation tests.

Formula
Syntax that contains one or more logical expressions that describe the conditions for the situation to monitor.

Distribution
Whether the situation is automatically distributed to instances of the agent or is available for manual distribution.

Run at startup
Whether the situation starts monitoring when the agent starts.

Sampling interval
Number of seconds that elapse between one sample of data that the monitoring agent collects for the server and the next sample.

Situation persistence
Whether the conditions specified in the situation evaluate to "true" for the defined number of occurrences in a row before the situation is raised. The default of one means that no persistence-checking takes place.

Severity
Severity of the predefined events: Warning, Informational, or Critical.

Clearing conditions
Controls when a true situation closes: after a period, when another situation is true, or whichever occurs first if both are selected.

IBM Tivoli Monitoring for Tivoli Network Manager navigator item
No predefined situations are included for this navigator item.

Availability navigator item
The situation descriptions are organized by the navigator item to which the situations are relevant.

KNP_NCPMONITOR_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula

```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'nco_p_ncpmonitor' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_NCP_CONFIG_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula

```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'ncp_config' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_NCP_D_HELPSRV_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula
```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'ncp_d_helpserv' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_NCP_G_EVENT_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula
```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'ncp_g_event' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_NCP_MODEL_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula
```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'ncp_model' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_NCP_POLLER_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula
```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'ncp_poller' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_NCP_STORE_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula
```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'ncp_store' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_NCP_VIRTUAL_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula

```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'ncp_virtualdomain' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_NCP_WEBTOOL_Process_Down situation

Description
The Network Manager process is not running.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula

```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ DOWN ) *AND ( *VALUE
KNP_AVAILABILITY.Application_Component *EQ 'ncp_webtool' ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_Process_CPU_Critical situation

Description
A running Network Manager process has high CPU usage.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula

```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ UP ) *AND ( *VALUE
KNP_AVAILABILITY.Percent_Processor_Time *GE 80 ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_Process_CPU_High situation

Description
CPU usage of the ITNM process is high.

The situation is evaluated for each distinct value of the COMPONENT attribute.

Formula

```
*IF ( ( *VALUE KNP_AVAILABILITY.Status *EQ UP ) *AND ( *VALUE
KNP_AVAILABILITY.Percent_Processor_Time *LT 80 ) *AND ( *VALUE
KNP_AVAILABILITY.Percent_Processor_Time *GE 20 ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Warning

Clearing conditions
The situation clears when the condition becomes false.

Discovery navigator item
No predefined situations are included for this navigator item.

Monitoring navigator item
The situation descriptions are organized by the navigator item to which the situations are relevant.

KNP_Polling_Capacitiy_Critical situation

Description
The monitor poller is close to reach the polling capacity.

The situation is evaluated for each distinct value of the TOTAL attribute.

Formula
```
*IF *VALUE KNP_POLLING_CAPACITY.Total *GE 60
```
Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_Polling_Capacitiy_Warning situation

Description
The monitor is near polling capacity.

The situation is evaluated for each distinct value of the TOTAL attribute.

Formula
```
*IF *VALUE KNP_POLLING_CAPACITY.Total *GE 50
```
Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Warning

Clearing conditions
The situation clears when the condition becomes false.

KNP_Work_Item_Queue_Warning situation

Description
The number of work items in the poller queue is high.

The situation is evaluated for each distinct value of the TOTAL attribute.

Formula
```
*IF *VALUE KNP_WORK_ITEM_QUEUE.Total *GT 0
```
Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
5 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Warning

Clearing conditions
The situation clears when the condition becomes false.

Network navigator item
The situation descriptions are organized by the navigator item to which the situations are relevant.

KNP_Total_Entities_Critical situation

Description
The total number of entities in the MODEL database is critical.

The situation is evaluated for each distinct value of the ENTITIES attribute.

Formula
```
*IF ( ( *VALUE KNP_ENTITIES_ON_MODEL_DB.Entities *GE 225000 ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
15 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Critical

Clearing conditions
The situation clears when the condition becomes false.

KNP_Total_Entities_Warning situation

Description
The total number of entities in the MODEL database is high.

The situation is evaluated for each distinct value of the ENTITIES attribute.

Formula
```
*IF ( ( *VALUE KNP_ENTITIES_ON_MODEL_DB.Entities *GE 200000 ) *AND
( *VALUE KNP_ENTITIES_ON_MODEL_DB.Entities *LE 225000 ) )
```

Distribution
This situation is automatically distributed to instances of this agent.

Run at startup
Yes

Sampling interval
15 minutes

Situation persistence
The number of times the conditions of the situation must occur for the situation to be true is 1.

Error conditions
Warning

Clearing conditions
The situation clears when the condition becomes false.

Take Action commands reference

Take Action commands can be run from the portal client or included in a situation or a policy.

About Take Action commands

When included in a situation, the command runs when the situation becomes true. A Take Action command in a situation is also referred to as *reflex automation*. When you enable a Take Action command in a situation, you automate a response to system conditions. For example, you can use a Take Action command to send a command to restart a process on the managed system or to send a text message to a cell phone.

In advanced automation, policies are used to take actions, schedule work, and automate manual tasks. A policy comprises a series of automated steps called activities that are connected to create a workflow. After an activity is completed, the Tivoli Enterprise Portal receives return-code feedback, and advanced automation logic responds with subsequent activities that are prescribed by the feedback.

A basic Take Action command shows the return code of the operation in a message box that is displayed after the action is completed or in a log file. After you close this window, no further information is available for this action.

IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition does not provide predefined Take Action commands.

Additional information about Take Action commands

For more information about working with Take Action commands, see "Take Action commands" in the *Tivoli Enterprise Portal User's Guide*.

Policies reference

Policies are used as an advanced automation technique for implementing more complex workflow strategies than you can create through simple automation. The IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition agent does not provide predefined policies, but you can create policies for any agent.

A *policy* is a set of automated system processes that can take actions, schedule work for users, or automate manual tasks. You use the Workflow Editor to design policies. You control the order in which the policy executes a series of automated steps, which are also called *activities*. Policies are connected to create a workflow. After an activity is completed, the Tivoli Enterprise Portal receives return-code feedback, and advanced automation logic responds with subsequent activities prescribed by the feedback.

For more information about working with policies, see "Automation with policies" in the *Tivoli Enterprise Portal User's Guide*.

For information about using the Workflow Editor, see the *IBM Tivoli Monitoring Administrator's Guide* or the Tivoli Enterprise Portal online help.

Chapter 11. Command reference

Use this information to understand which command line options to use when starting Network Manager processes.

By default, the master process controller, ncp_ctrl, launches and manages Network Manager processes. You can configure the options with which the processes are started using the instructions for *Configuring managed processes* in the *IBM Tivoli Network Manager IP Edition Administration Guide* and the information in this section about the command line options. You can also start processes manually if necessary using the command-line options described here

For further information on databases related to Network Manager processes, see the *IBM Tivoli Network Manager Reference*.

For information about Dashboard Application Services Hub commands, see the Dashboard Application Services Hub command reference topics on the Jazz for Service Management information center at https://www.ibm.com/support/knowledgecenter/SSEKCU

Commands to start Network Manager

Use these commands to start and stop Network Manager and all its processes.

itnm_status command line options

Use the **itnm_status** command, with optional advanced arguments, to retrieve information about the components that are running.

This command also can also be used to retrieve status on the Apache Storm real-time computation system, which us used to aggregate raw poll data into historical poll data.

Note: If you installed Network Manager as a non-root user, then you will only be able to run this script as the installing user. You will need to run one of the `setup_run_as*` scripts in order to run this script as root or as other users. For more information, see *setup_run_as* scripts* in the *IBM Tivoli Network Manager Reference*.

Example commands

The following command reports the status of all components:

```
itnm_status
```

The following command reports the status of Network Manager components:

```
itnm_status ncp
```

All command line options

The following table describes all the command line options.

© Copyright IBM Corp. 2006, 2021

Table 19. Command line options	
Command line options	**Description**
components	Optional; one or more component abbreviations. If no component is specified, the status of all components is reported. The following component abbreviations are possible: **Network Manager** ncp **Apache Storm** storm
`-verbose`	Optional; provides more information on the screen.
`-help`	Optional; displays help on screen.

itnm_start command line options

Use the **`itnm_start`** command to start Network Manager components.

You can start Network Manager and all of its component s using the **`itnm_start`** command. This command also can also be used to start the Apache Storm real-time computation system, which us used to aggregate raw poll data into historical poll data.

Note: If you installed Network Manager as a non-root user, then you will only be able to run this script as the installing user. You will need to run one of the `setup_run_as*` scripts in order to run this script as root or as other users. For more information, see *setup_run_as* scripts* in the *IBM Tivoli Network Manager Reference*.

Example commands

The following command starts all components in the domain specified during installation:

```
itnm_start
```

The following example starts the core components in domain NCOMS:

```
itnm_start ncp -domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 20. itnm_start command-line options	
Command-line options	**Description**
components	Optional; one or more component abbreviations. If no component is specified, all components are started. The following component abbreviations are possible: **Network Manager** ncp **Apache Storm** storm

Table 20. itnm_start command-line options (continued)	
Command-line options	**Description**
`-domain` *DomainName*	Optional; applies to the Network Manager component. If not used, then the default domain is used (the one specified during install).
`-verbose`	Optional; provides more information on the screen.
`-help`	Optional; displays help on screen.

itnm_stop command line options

Use the **`itnm_stop`** command to stop Network Manager components.

You can stop Network Manager and all of its components, using the **`itnm_stop`** command. This command also can also be used to stop the Apache Storm real-time computation system, which us used to aggregate raw poll data into historical poll data.

Note: If you installed Network Manager as a non-root user, then you will only be able to run this script as the installing user. You will need to run one of the `setup_run_as*` scripts in order to run this script as root or as other users. For more information, see *setup_run_as* scripts* in the *IBM Tivoli Network Manager Reference*.

Example commands

The following command stops only the Network Manager components in the domain specified during installation:

```
itnm_stop ncp
```

The following example stops the core components in domain NCOMS:

```
itnm_stop ncp -domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 21. Command line options	
Command line options	**Description**
components	Optional; one or more component abbreviations. If no component is specified, all components are stopped. The following component abbreviations are possible: **Network Manager** ncp **Apache Storm** storm
`-domain` *DomainName*	Optional; applies to the Network Manager domain. If not used, then the default domain is used (the one specified during install).
`-verbose`	Optional; provides more information on the screen.
`-help`	Optional; displays help on screen.

Restarting the Dashboard Application Services Hub server

After customization and configuration activities, you might need to restart the Dashboard Application Services Hub.

About this task

Fix Pack 7 You do not normally need to restart the Dashboard Application Services Hub server if you change a `.properties` file that belongs to Network Manager. Such changes are read automatically.

Restarting Dashboard Application Services Hub restarts all of the GUI applications that are running on your Dashboard Application Services Hub server.

The applications that are restarted include Network Manager GUI applications and Web GUI applications.

To restart the server:

Procedure

1. On the relevant Dashboard Application Services Hub, open a command window.
2. Change to the `$JazzSM_HOME/profile/bin` directory.
3. Stop the server by using the following command:

   ```
   stopServer.sh server1
   ```

   Note: You are prompted to supply the user name and password of the administrative user.
4. Wait a moment for the server to completely shut down and, confirm that all Java processes stopped running.

   The following messages confirm that the server is shut down:

   ```
   ADMU3201I: Server stop request issued. Waiting for stop status.
   ADMU4000I: Server server1 stop completed.
   ```
5. Start the server by using the following command:

   ```
   startServer.sh server1
   ```

Process control

Network Manager processes run under the control of the Master process controller, `ncp_ctrl`.

ncp_ctrl command line options

You can use the `ncp_ctrl` command to start the Master process controller. Alternatively, start the product using the `itnm_start` command. This command automatically starts up `ncp_ctrl`, which in turn starts all of the other Network Manager processes.

Example commands

The following command starts the ncp_ctrl process in the NCOMS domain:

```
ncp_ctrl –domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 22. Command line options	
Option	**Explanation**
-domain *DOMAIN_NAME*	Required. The name of the domain under which to run the ncp_ctrl process.
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
-help	Displays the command line options. Does not start the component even if used in conjunction with other arguments.
-latency *LATENCY*	The maximum time in milliseconds (ms) that the ncp_ctrl process waits to connect to another Network Manager process using the messaging bus. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic.
-logdir *DIRNAME*	Specifies the directory where log files are to be added and directs log messages for each process started by the ncp_ctrl process to a separate file in the specified directory.
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
-messagelog *PATHTOLOGFILE*	The path to the message log file.
-nologdir	Turns off log file writing and prints log messages on screen.
-slave	Indicates that this instance of the ncp_ctrl process is to be run in slave mode. The slave ncp_ctrl process must have the same domain name as the master ncp_ctrl process. When running in slave mode, ncp_ctrl accepts requests from the master process to launch services. The slave mode can be used to distribute processes.
-version	Displays the version number of the component. Does not start the component even if used in conjunction with other arguments.

Commands for processes under process control

Network Manager processes run under the Master process controller, `ncp_ctrl`. The following topics list the commands to start these processes and to start the Master process controller, `ncp_ctrl`. Note that you would not normally use these commands to start these processes, as they would all be started by `ncp_ctrl` when you start up Network Manager using the `itnm_start` command.

ncp_class command line options

Use the ncp_class command to start the Active Object Class manager.

It is best practice to configure the ncp_ctrl process, the master process controller, to launch and manage the ncp_class process.

Restriction: If you are using Network Manager with failover, you must start ncp_class using the ncp_ctrl process. The ncp_ctrl process checks the status of the ncp_class process and uses this information to generate the Health Check events used by the failover process.

Prequisites

For ncp_class to run, it requires access to the NCIM database. The connection and access to the database is performed automatically, you do not need to do anything.

The AOC files define the class hierarchy. There is a copy of this hierarchy in the `entityClass` NCIM database table. If changes are made to the AOC files, ncp_class updates the `entityClass` database table when it connects to the NCIM database to reflect any changes to the AOC hierarchy.

Note: If you create a new AOC file, you should also add a new insert to the `class.classIds` database table in the `ClassSchema.cfg` configuration file.

Example commands

The following command starts the ncp_class process manually in the NCOMS domain:

```
itnm_class -domain NCOMS
```

All command line options

The following table describes the command line options of the ncp_class command.

Table 23. `ncp_class` command line options

Option	Explanation
`-cachepercent` *PERCENTAGE_OF_CACHE_IN_MEMORY*	Enables you to specify a ratio of either 0% or 100% of the cache that is resident in memory to the cache that is resident on the hard disk. The ratio that you specify depends on the amount of memory that exists on the host machine and the number of processes it is running. The default value is 0% cache. The -cachepercent option can be used to reduce the memory required when the process responds to OQL queries that result in large numbers of records being returned. Do not use this command-line option for permanent data storage, because the cache is cleared when the process exits.
-domain *DOMAIN_NAME*	The name of the domain under which to run ncp_class.
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).

Table 23. ncp_class command line options (continued)	
Option	**Explanation**
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
-messagelog *PATH_TO_LOGFILE*	The path to the message log file.
-help	Displays the command line options. Does not start the component even if used in conjunction with other arguments.
-latency *LATENCY*	The maximum time in milliseconds (ms) that ncp_class waits to connect to another Network Manager process by means of the messaging bus. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic.
-read_aocs_from *DIRECTORY_NAME*	The full path of the directory from which to read the AOC definitions. When ncp_class is launched it obtains the AOC definitions either from this directory or from the cache files contained in `NCHOME/var/precision`. • If you omit the `-read_aocs_from` argument from the command line, ncp_class automatically reads the cache files from `NCHOME/var/precision` if they are present. If there are no valid cache files, ncp_class uses the definitions in `NCHOME/precision/aoc`. • If you specify a directory using the `-read_aocs_from` option, ncp_class initializes itself with the AOCs located in the specified directory. The ncp_class process automatically moves any existing cache files to a backup directory. Regardless of the options specified, ncp_class generates a warning if the files within the `NCHOME/precision/aoc` directory are more recent than the cache files contained within the `NCHOME/var/precision` directory. For more information about failover, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.
-version	Displays the version number of the component. Does not start the component even if used in conjunction with other arguments.

ncp_config command line options

Use the ncp_config command to start the Network Manager GUI configuration file server.

The ncp_config process provides a means for the configuration GUIs to read from and write to schema files.

The ncp_config process is started and stopped by ncp_ctrl and so in general need not be manually started.

Example commands

The following example command starts the ncp_config process in the domain NCOMS with the highest level of debugging output.

```
ncp_config -domain NCOMS -debug 4
```

All command line options

The following table describes the command line options of the ncp_config command.

Table 24. Explanation of Command Line Options

Option	Explanation
`-domain` *DOMAIN_NAME*	Required. The name of the domain under which to run the ncp_config process. Data saved in the GUI is saved to the domain name that you specify here.
`-debug` *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
`-help`	Displays the command line options. Does not start the component even if used in conjunction with other arguments.
`-latency` *LATENCY*	The maximum time in milliseconds (ms) that the ncp_config process waits to connect to another Network Manager process using the messaging bus. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic.
`-logfileusepool`	Enable log pooling. This setting is off by default. For more information about how to use the log settings, see “Avoiding process errors with large trace or log files by using log file rotation” on page 33.
`-logfilepoolsize`	Set the log pool size. Valid values are: 2 to 99. The default value is 10. For more information about how to use the log settings, see “Avoiding process errors with large trace or log files by using log file rotation” on page 33.

Table 24. Explanation of Command Line Options (continued)	
Option	**Explanation**
`-messagelevel` *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
`-messagelog` *PATHTOLOGFILE*	The path to the message log file.
`-query` *QUERY*	A query statement to pass to the OQL Service Provider. This option is designed to run the `ncp_config` process in batch mode. In batch mode, `ncp_config` processes the OQL query statement against the requested database, writes the schema files, and exits.
`-schema`	The schema file you want to use.
`-read_schemas_from` *DIRECTORY_NAME*	The full path of the directory from which the ncp_config process reads the schema files. This command line option can only be used when starting the `ncp_config` process manually. If this option is not specified, `NCHOME/etc/precision` is used as a default.
`-version`	Displays the version number of the component. Does not start the component even if used in conjunction with other arguments.
`-write_schemas_to` *DIRECTORY_NAME*	The full path to which the ncp_config process writes the schema files. This command line option can only be used when starting the `ncp_config` process manually. If no path is specified, the ncp_config process updates the files in the source directory, saving a backup of the existing schema.

ncp_disco command line options

Use the ncp_disco command to start the Discovery engine.

Example commands

The following command starts ncp_disco in the domain NCOMS:

```
ncp_disco -domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 25. Command line options	
Option	**Explanation**
`-activeOnBackupDomain`	Specifies whether to have a discovery active on the backup domain of a failover pair.  **Attention:** This option is provided for non-standard Network Manager configurations only and must not be set if you are running a standard Network Manager failover configuration. Standard failover does not work if this flag is set. Do not set this flag if you are running the ncp_virtualdomain process.
`-cachepercent` *PERCENTAGE_OF_CACHE_IN_MEMORY*	Used to specify a ratio of either 0% or 100% of the cache that is resident in memory to the cache that is resident on the hard disk. The ratio that you specify depends on the amount of memory that exists on the host server and the number of processes it is running. The default value is 0% cache. The `-cachepercent` option can be used to reduce the memory that is required when the process responds to OQL queries that return large numbers of records. Do not use this command line option for permanent data storage because the cache is cleared when the process exits.
`-domain` *DOMAIN_NAME*	Required. The name of the domain under which to run the ncp_disco process.
`-discoOnStartup {0 | 1}`	Specifies whether to have a new discovery start automatically when ncp_disco starts up. The options are: • 0 – discovery does not start automatically when ncp_disco starts up • 1 – discovery starts automatically when ncp_disco starts up By default, a new discovery does not start automatically.
`-debug` *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
`-help`	Displays the command line options. Does not start the component even if used with other arguments.
`-latency` *LATENCY*	The maximum time in milliseconds (ms) that the **ncp_ctrl** process waits to connect to another Network Manager process by using the messaging bus. This option is useful for busy networks where processes can assume that a problem occurred, when in fact the communication delay is a result of network traffic.
`-LoadCacheIntoMem` *CACHE_DOMAIN*	Loads discovery cache files from *CACHE_DOMAIN* into memory for a discovery in another domain. This option is for specialist use, as a replacement for loading cache files onto disk.

Table 25. Command line options (continued)	
Option	**Explanation**
`-messagelevel` *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • `debug` • `info` • `warn` • `error` • `fatal`
`-messagelog` *PATHTOLOGFILE*	The path to the log file.
`-version`	Displays the version number of the component. Does not start the component even if used with other arguments.

ncp_d_helpserv command line options

Use the ncp_d_helpserv command to start the Helper Server.

Example commands

The following command starts the Helper Server in the domain NCOMS:

```
ncp_d_helpserv –domain DOMAIN_NAME
```

All command line options

The following table describes all the command line options.

Table 26. Command line options	
Option	**Explanation**
-domain *DOMAIN_NAME*	Required. The name of the domain under which to run the Helper Server.
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
-help	Displays the command line options. Does not start the component even if used with other arguments.
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • `debug` • `info` • `warn` • `error` • `fatal`
-messagelog *PATHTOLOGFILE*	The path to the message log file.

Table 26. Command line options (continued)	
Option	**Explanation**
-noexpiry	Fix Pack 3 Allows cached records to be used even if they have expired. Use this option with the **-persist** option if you are using cached helper data in discovery. Fix Pack 3 You might want to use cached helper data for troubleshooting purposes if you are directed to do so by IBM Support.
-persist	Fix Pack 3 Caches helper data to disk during a discovery. Only data that is configured for caching by using the `m_HelperDoWeStore` and `m_HelperDoWeQuery` options in the `DiscoHelperServerSchema.cfg` configuration file is cached. If you run a discovery with this option enabled, cached helper data is used in that discovery. Fix Pack 3 Use this option with the **-noexpiry** option if you are replaying a helper server cache. You might want to use cached helper data for troubleshooting purposes if you are directed to do so by IBM Support.
-version	Displays the version number of the component. Does not start the component even if used with other arguments.

Starting helpers

Provided that the Helper Server is launched by CTRL, the individual helpers are automatically started as and when they are required.

The only situation in which you might need to configure an insert for an individual helper into the disco.managedProcesses table would be if you wanted to start that helper on a remote machine (that is, a machine other than the one that is running the Helper Server). In this situation, you would insert the required helper into the disco.managedProcesses table, specifying the appropriate remote host in the m_Host field.

ncp_g_event command line options

Use the **ncp_g_event** command to start the Event Gateway.

Example commands

The following command starts the Helper Server in the domain NCOMS:

```
ncp_d_helpserv –domain DOMAIN_NAME
```

Attention: If you are using Network Manager with failover, you must start the Event Gateway using the **ncp_ctrl** process. The **ncp_ctrl** process checks the status of the Event Gateway component and uses this information to generate the health check events used by the failover process.

For more information about failover, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

All command line options

The following table describes all the command line options.

Table 27. Command line options	
Option	**Description**
-domain *DOMAIN_NAME*	Required. The name of the domain under which to run **ncp_g_event**.
-debug *DEBUG*	The level of debugging output (1-4), where 4 represents the most detailed output.
-help	Prints out the command-line options for **ncp_g_event** then exits.
-latency *LATENCY*	The maximum time in milliseconds (ms) that **ncp_g_event** waits for a response to a query of another Network Manager process. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic. The default value is 10000. If you specify a lower value on the command line, it is increased to 10000. For large topologies, make sure you set a value of at least a few minutes.
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
-messagelog *PATHTOMSGLOGFILE*	The path to the message log file.
-logdir *PATHTOLOGFILE*	The directory to write log files to.
-version	Prints the version number of **ncp_g_event** then exits.

ncp_model command line options

Use the **ncp_model** command to start the topology manager.

Example commands

It is best practice to configure the **ncp_ctrl** process, the master process controller, to launch and manage the **ncp_model** process. You can configure the behavior of the **ncp_model** process when it is started by using the command line options.

The following example command starts the ncp_model process in the domain NCOMS:

```
ncp_model –domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 28. Command line options

Option	Explanation
-domain *DOMAIN_NAME*	Required. The name of the domain under which to run the **ncp_model** process.
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
-help	Displays the command line options. Does not start the component even if used in conjunction with other arguments.
-latency *LATENCY*	The maximum time in milliseconds (ms) that the **ncp_model** process waits to connect to another Precision Server process by means of the messaging bus. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic.
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
-messagelog *PATHTOLOGFILE*	The path to the message log file.
-version	Displays the version number of the component. Does not start the component even if used in conjunction with other arguments.

nco_p_ncpmonitor command line options

On UNIX operating systems, use the **nco_p_ncpmonitor** command to start and configure the Probe for Tivoli Netcool/OMNIbus.

Usage considerations

Use this command only to troubleshoot IBM Tivoli Network Manager IP Edition.

The **nco_p_ncpmonitor** command starts the Probe for Tivoli Netcool/OMNIbus independently of the domain process controller ncp_ctrl. If you are using Network Manager with failover, you must start the Probe for Tivoli Netcool/OMNIbus using ncp_ctrl. The ncp_ctrl process checks the status of the Probe for Tivoli Netcool/OMNIbus and uses this information to generate the Health Check events used by the failover process.

There are no dependencies for starting the Probe for Tivoli Netcool/OMNIbus.

Example commands

The following example command starts the nco_p_ncpmonitor process in the domain NCOMS:

```
nco_p_ncpmonitor –domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 29. Command line options

Option	Explanation
-autosaf	Enable automatic store and forward mode.
-buffer	Allows you to turn on alert buffering.
-buffersize	The size of the alert buffer to use.
-capturefile	Raw capture file to write to.
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
-domain *DOMAIN_NAME*	Required. The name of the domain under which Network Manager processes are running.
–help	Prints out a synopsis of all command line options for the component. If specified, the component is *not* started.
-latency *LATENCY*	The maximum time in milliseconds (ms) that the component waits to connect to another Network Manager process via the messaging bus. This option is useful for large and busy networks where the default settings can cause the process to assume that there is a problem when in fact the communication delay is a result of network traffic.
-manager	Manager name.
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
-messagelog *PATHTOLOGFILE*	The path to the message log file.
-name	Name of probe.
-noautosaf	Disable automatic store and forward mode.
-nobuffer	Allows you to turn off alert buffering.

Table 29. Command line options (continued)	
Option	**Explanation**
-noraw	Allows you to turn off raw capture mode.
-nosaf	Disable store and forward mode.
-propsfile	Properties file to use.
-raw	Allows you to turn on raw capture mode.
-rulesfile	Rules file to use.
-saf	Enable store and forward mode.
-server	The name of the ObjectServer to connect to. **Tip:** The `$NCHOME/etc/precision/ConfigItnm.cfg` file provides a simpler, efficient alternative for configuring failover. Use this file instead of the `-server` command-line option, to specify the ObjectServer name.
-version	Prints the version number of the component. If specified, the component is *not* started even if `-version` is used in conjunction with other arguments.

ncp_poller command line options

Use the **ncp_poller** command to start the network polling engine.

Usage considerations

It is best practice to configure the **ncp_ctrl** process, the master process controller, to launch and manage the **ncp_poller** process.

Attention: If you are using Network Manager with failover, you must start the **ncp_poller** process by using the **ncp_ctrl** process. The **ncp_ctrl** process checks the status of the **ncp_poller** process and uses this information to generate the health check events used by the failover process.

Prerequisites

Before you manually start the **ncp_poller** process, the following Network Manager processes must be running:

- The ncp_model process must be running in order to pass the network topology to the polling subsystem.
- The Probe for Tivoli Netcool/OMNIbus must be running to transfer events to the ObjectServer.

Example commands

The following example command starts the ncp_poller process in the domain NCOMS:

```
ncp_poller –domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 30. Command line options	
Option	**Explanation**
-admin | -noadmin	Use the -admin option to configure this instance of the Polling engine, ncp_poller, to perform administrative functions, such as updating the monitoring view caches, and performing monitoring of the poll data tables. Use the -noadmin option to configure this instance of ncp_poller not to perform administrative functions. If you do not specify the -admin option or the -noadmin option, the poller behaves as if the -admin option is specified. If you are using multiple pollers, start one of the pollers with the -admin option and the other pollers with the -noadmin option. For performance reasons, the poller with the smallest polling load is usually started with the -admin option.
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
-deregister	Deregisters a poller. Any polls assigned to a poller that is not registered will not run.
-domain *DOMAIN_NAME*	Required. The name of the domain under which to run this instance of ncp_poller.
-force	Forces deregistration of the poller. Deletes all polling policies assigned to the named poller.
-help	Prints out a synopsis of all command-line options for this instance of ncp_poller then exits.
-latency *LATENCY*	The maximum time in milliseconds (ms) that this instance of ncp_poller process waits for a response to a query of another Network Manager process. This option is useful for large and busy networks where the default settings can cause the process to assume that there is a problem when in fact the communication delay is a result of network traffic. For large topologies, make sure you set a value of at least a few minutes.
-logdir	The directory to write process log files to.
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
-messagelog *PATHTOLOGFILE*	The path to the message log file.
-name	Run as a named poller. Only polls assigned to this poller will be run.
-readsnmpconfig	Instructs the poller to read the configuration from the `SnmpStackSecurityInfo.cfg` file and write it to the configuration database.

Table 30. Command line options (continued)	
Option	**Explanation**
-register	Registers the poller. Pollers must be registered before polls can be assigned to them.
-version	Prints the version number of **ncp_poller** then exits.

For more information on failover, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

ncp_store command line options

Use the ncp_store command to start the ncp_store process.

Example commands

It is best practice to configure the **ncp_ctrl** process, the master process controller, to launch and manage the **ncp_store** process. You can configure the behavior of the **ncp_store** process when it is started by using the command line options. The following example command starts the ncp_store process in the domain NCOMS:

```
ncp_store –domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 31. Command line options	
Option	**Explanation**
`-cachepercent` *PERCENTAGE_OF_CACHE_IN_MEMORY*	Enables you to specify a ratio of either 0% or 100% of the cache that is resident in memory to the cache that is resident on the hard disk. The ratio that you specify depends on the amount of memory that exists on the host machine and the number of processes it is running. The default value is 0% cache. The -cachepercent option can be used to reduce the memory required when the process responds to OQL queries that result in large numbers of records being returned. Do not use this command-line option for permanent data storage, because the cache is cleared when the process exits.
-domain *DOMAIN_NAME*	Required. The name of the domain under which to run the ncp_store process.
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
-help	Displays the command line options. Does not start the component even if used in conjunction with other arguments.

Table 31. Command line options (continued)	
Option	**Explanation**
-latency *LATENCY*	The maximum time in milliseconds (ms) that the ncp_store process waits to connect to another Network Manager process by means of the messaging bus. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic.
-logdir *PATHTOLOGFILE*	The directory to write log files to.
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
-messagelog *PATHTOMSGLOGFILE*	The path to the message log file.
-version	Displays the version number of the component. Does not start the component even if used in conjunction with other arguments.

ncp_virtualdomain command line options

Use the **ncp_virtualdomain** command to start the Virtual Domain component.

Usage considerations

Virtual Domain is used when running Network Manager with failover.

It is best practice to configure the **ncp_ctrl** process, the master process controller, to launch and manage Virtual Domain. You can configure the behavior of the process when it is started by using the command line options.

Example commands

The following example command starts the ncp_virtualdomain process in the domain NCOMS:

```
ncp_virtualdomain –domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 32. Command line options	
Option	**Description**
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).

Table 32. Command line options (continued)	
Option	**Description**
`-domain` *DOMAIN_NAME*	Required. The name of the domain under which the Network Manager component is running. This name must be different for the primary and backup Network Manager servers.
`-help`	Prints out a synopsis of all command-line options for the component. If specified, the component is not started even if `-help` is used in conjunction with other arguments.
`-latency` *LATENCY*	The maximum time in milliseconds (ms) that the Virtual Domain component waits to connect to another Network Manager process using the messaging bus.
`-messagelevel` *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
`-messagelog` *PATH_TO_LOGFILE*	The path to the message log file.
`-version`	Prints the version number of the component. If specified, the component is not started even if `-version` is used in conjunction with other arguments.

ncp_webtool command-line options

Use the **ncp_webtool** command to run the Web tools on the backend server.

Usage considerations

The ncp_webtool process transfers the running of the Web tools to the backend server in environments where Topoviz is running on a different server to the Network Manager backend processes and there is a firewall between the two. This transfer makes the Web tools available across such distributed environments.

You do not need to interact directly with ncp_webtool process, because it is a sever-time application that runs without user input.

It is best practice to configure the ncp_ctrl process, the master process controller, to launch and manage the ncp_webtool process. You can configure the behavior of the process by using the command line options.

All command line options

Command line options are explained in the following table.

Table 33. Command line options	
Option	**Explanation**
-debug *debug level*	Specifies the level of debugging output (1-4, where 4 represents the most detailed output).
-domain	Required. Specifies the domain.
-help	Displays the command line options.
-latency	Specifies the network timeout value to use (in milliseconds).
-messagelevel *message level*	Specifies the message level to use for log files: • Debug • Info • Warn • Error • Fatal The default is Warn.
-messagelog	Displays the path to the message log file.
-version	Displays the version number of the component.

Commands for processes not started by default

The following topics list the commands for processes that are not started by default when Network Manager starts up.

ncp_crypt command line options

Use the ncp_crypt command to start ncp_crypt, the password encryption utility.

Example commands

The following command starts ncp_crypt and encrypts the password provided:

```
ncp_crypt -password password
```

Note: All password encryption in Network Manager is performed using FIPS 140–2 compliant algorithms.

The following command starts ncp_crypt and **decrypts** the password provided:

```
ncp_crypt -password password -decrypt
```

All command line options

The following table describes all the command line options.

Table 34. Command line options	
Option	**Explanation**
-password	Required. Specifies the password to encrypt or decrypt. By default the password is encrypted.

Table 34. Command line options (continued)	
Option	**Explanation**
-decrypt	Optional. Overrides the default and instructs ncp_crypt to decrypt the password.
-help	Optional. Displays the command line options. Does not start the component even if used in conjunction with other arguments.
-version	Optional. Displays the version number of the component. Does not start the component even if used in conjunction with other arguments.

ncp_mib command line options

Use the **ncp_mib** command to update and compare MIB files. You need to run **ncp_mib** only after you have added new MIBs.

Prerequisites

All MIBs must be valid in order to be parsed correctly.

There are no process dependencies for the **ncp_mib** process.

Example commands

The following command starts the ncp_mib process using the generic `MibDbLogin.cfg` file:

```
ncp_mib
```

The following command starts the ncp_mib process using the `MibDbLogin.NCOMS.cfg` file:

```
ncp_mib –domain NCOMS
```

The following example database query verifies that a MIB has loaded successfully. You must run **ncp_mib** before running this database query.

```
 select * from ncmib.mib_modules where moduleName ='RFC1213-MIB';
```

If the MIB loaded, a table is displayed containing a moduleName of RFC1213-MIB.

You can also verify that MIBs are loaded by running the **ncp_mib** command with the `-messagelevel info` option. A message similar to the following informs you that the MIBs are being processed:

```
09/10/14 12:41:08: Information: I-MIB-001-013: [1096571552t]
Resolving references for module 'RFC1213-MIB'
```

When processing completes, a message states that the MIBs have been committed to the database.

All command line options

The following table describes all the command line options.

Table 35. Command line options	
Option	**Explanation**
`-db`	Specifies the MIB Database ID, as defined in the `MibDbLogin.cfg` file. The default value is MIB.

Table 35. Command line options (continued)	
Option	**Explanation**
`-debug` *`debug level`*	The level of debugging output (1-4, where 4 represents the most detailed output).
`-diff`	Compares the list of MIBs in files with those in the SQL DB and show difference.
`-domain`	Optional. There is only one MIB database for all domains. The -domain option, if specified, configures ncp_mib to use a domain-specific `MibDbLogin.`*`domain`*`.cfg` file. If no -domain option is specified, the generic `MibDbLogin.cfg` file is used.
`-dryrun`	Shows what would be done by issuing the command but does not alter the SQL database in any way.
`-emptydb`	Removes all the MIB data from the database and exits.
`-force`	Attempts to resolve and insert all parsed MIB objects, regardless of whether all dependencies are met. This option allows the insertion of a partially resolved MIB module, inserting those MIB objects which have their dependencies satisfied while leaving out those MIB objects which do not.
`-help`	Displays the command line options. Does not start the component even if used in conjunction with other arguments.
`-logdir`	Specifies the directory into which log files are written.
`-messagelevel` *`message level`*	Specifies the message level to use for log files: • Debug • Info • Warn • Error • Fatal The default is Warn.
`-messagelog`	Specifies the path to the message log.
`-override`	Empties the database and then attempts to import all the MIBs. If there are errors, such as unresolved MIB modules, the transaction is rolled back and the original state of the database is restored.
`-showcontents`	Displays a list of the MIB modules that are contained in the MIB files and states which ones are resolved.
`-version`	Displays the version number of the component.

ncp_oql command line options

Use the ncp_oql command to query and update data in Network Manager management databases from the command line.

Example commands

The following example command starts the ncp_oql process (known as the OQL Service Provider) in the domain NCOMS:

```
ncp_oql –domain NCOMS
```

The following example command starts the OQL Service Provider with a username and password supplied on the command line:

```
ncp_oql –domain NCOMS -username FRED -password wilma
```

Table 36. Explanation of Command Line Options

Option	Explanation
`-dbId` *DATABASE_ID*	This option is only valid when used in conjunction with the ncim service. Choose the database from DbLogins to connect to. The default value is NCIM.
`-debug` *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).
`-displayNulls`	Optional. If specified, null values are displayed. If this option is omitted, null values are not displayed. This option takes effect only in non-tabular mode.
`-domain` *DOMAIN_NAME*	Required. The name of the domain in which the processes that you want to query are running. The process whose databases you wish to query must be running in order for the databases to be queried.
`-help`	All Network Manager components have a special `-help` option that displays the command line options. The component is not started even if `–help` is used in conjunction with other arguments.
`-history` *HISTORY_SIZE*	The size of the command line history.
`-latency` **LATENCY**	The maximum time in milliseconds (ms) that the service provider waits to connect to another Network Manager process using the messaging bus. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic. The default value is 3000 (equivalent to 3 seconds). You might want to increase this value as the default value might not be long enough to get a response from a large or busy OQL database.

Table 36. Explanation of Command Line Options (continued)

Option	Explanation
`-messagelevel` *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
`-options`	Displays the list of options allowed as query services.
`-oqldump`	Using this argument results in the databases from the application being converted into OQL create and insert statements. This can be a useful means of recording the internal state of a component for debugging or analysis at a later date.
`-outputFile`	A file to write the output to.
`-password` *PASSWORD*	Optional. A password is required in the following circumstances: • If you query the `ncim` service, use the password for the NCIM topology database, in conjunction with the NCIM username. • If you query the `objectServer` service, use the ObjectServer password in conjunction with the ObjectServer username. • If you query Network Manager core components, you need to use a password only if password authentication has been enabled in the `NCHOME/etc/precision/ConfigSchema.cfg` file. The -password option can also be used with the `-query` option, to enable OQL queries to be placed in scripts. For security reasons you must use this option carefully, because other users may be able to see your password. You can also enter your password when prompted rather than at the command line.
`-plugin`	The Event Gateway stitcher plugin to connect to. Takes the value of the subdirectory of the relevant stitcher.
`-poller`	Specifies a poller instance to connect to. If none is specified then this option defaults to the `ncp_poller_default` poller instance.
`-query` *QUERY*	A query to pass to the OQL Service Provider, designed to allow OQL statements in scripts (used in conjunction with `-password`).

Table 36. Explanation of Command Line Options (continued)	
Option	**Explanation**
`-schema` *PATH_TO_SCHEMA_FILE*	This option is only valid when used in conjunction with the ncp_config service. Specifies a schema file to use.
`-server` *SERVER_NAME*	This option is valid only with the ObjectServer service. It defaults to the current ObjectServer given in the ConfigItnm file.
`-service` *SERVICE_NAME*	Required. The service you want to interrogate. Run **ncp_oql** with the `-options` command line option to list all available services.
`-snoop`	Shows queries made on the service.
`-tabular`	Toggles tabular results display format.
`-updates`	Shows updates made on the service.
`-username` *USERNAME*	The username to use to log into the service provider. Required if the OQL Service Provider is running in authentication mode.
`-version`	All Network Manager components have a special `-version` option that displays the version number of the component. The component is not started even if `–version` is used in conjunction with other arguments.

ncp_trapmux command line options

Use the **ncp_trapmux** command to start the SNMP trap multiplexer. The SNMP trap multiplexer listens to a single port and forwards all the traps it receives to a set of host/socket pairs.

Restriction: The SNMP trap multiplexer does not forward SNMPv3 Inform messages.

Example commands

The following command starts the ncp_store process in the domain NCOMS.

```
ncp_store –domain NCOMS
```

All command line options

The following table describes all the command line options.

Table 37. Command line options	
Option	**Explanation**
-domain *DOMAIN_NAME*	Required. The name of the domain under which to run the trap multiplexer.
-debug *DEBUG*	The level of debugging output (1-4, where 4 represents the most detailed output).

Table 37. Command line options (continued)	
Option	**Explanation**
-help	Displays the command line options. Does not start the component even if used in conjunction with other arguments.
-latency *LATENCY*	The maximum time in milliseconds (ms) that the ncp_trapmux process waits to connect to another Network Manager process by means of the messaging bus. This option is useful for large and busy networks where the default settings can cause processes to assume that there is a problem when in fact the communication delay is a result of network traffic.
-messagelevel *MESSAGELEVEL*	The level of messages to be logged (the default is warn): • debug • info • warn • error • fatal
-messagelog *PATH_TO_LOGFILE*	The path to the message log file.
-version	Displays the version number of the component. Does not start the component even if used in conjunction with other arguments.

Part 2. Discovering the network

Discovering the network involves building your network topology by performing a series of initial discoveries and then ensuring that you always have an up-to-date network topology by means of regular discoveries.

About this task

The following topics describe how to discover the network and keep the discovery up to date.

© Copyright IBM Corp. 2006, 2021

Chapter 12. About discovery

Configure discovery by setting the parameters that govern how the discovery is performed.

About types of discovery

Different terms are used to describe network discovery, depending on what is being discovered and how the discovery has been configured. You can run discoveries, rediscoveries, and full and partial discoveries.

Discovery and rediscovery

Discovery

The term discovery is used generally to mean any kind of discovery. Technically, only the first discovery that is run after the discovery engine, ncp_disco, is started can properly be called a discovery, and every discovery after that is a rediscovery. Because there is no discovery data in memory yet, discoveries take slightly longer than rediscoveries.

Rediscovery

After a discovery has been run, any further discoveries that are run are rediscoveries. Rediscoveries use a different data flow to discoveries, with some different stitchers and database tables. If ncp_disco is restarted, the next discovery is again just a discovery, and further discoveries after that are rediscoveries. Unless you are running advanced discoveries or modifying the discovery data flow, the difference between a discovery and a rediscovery is not usually important, and, for ease of reading, the instructions in this information do not distinguish between discovery and rediscovery unless it is necessary.

Full and partial discovery

Full discovery

A full discovery is 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.

Partial discovery

A partial discovery is 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, the **ncp_disco** process, has not been stopped since the last full discovery. A partial discovery is, therefore, actually a type of rediscovery.

Scheduled discovery

You can schedule a full discovery to start at a certain time. For example, you can schedule a full discovery to run at the same time every night, or at the same time on the same day every week, or at the same time every day of the month.

Scopes

Define the zones of the network (that is, subnet ranges) that you want to include in the discovery, and the zones that you want to exclude. The areas of the network to be included in the discovery process, or excluded from the discovery process are collectively known as the discovery scope.

There are several benefits to limiting the discovery scope:

- It is important to limit discovery scope because the range of IP addresses considered by the default discovery process is potentially unlimited. Without a scope, the discovery attempts to recognize every network device. By limiting the scope, you can concentrate on the important areas of your network.

© Copyright IBM Corp. 2006, 2021

Attention: If there are routes out of your network to the Internet, then an unscoped discovery will find these routes and proceed to discover parts of the Internet.

- You might also want to restrict the scope of the discovery to control the discovery of sensitive devices that you do not want to poll. A device might be considered sensitive because there is a security risk involved in polling the device, or because polling might overload the device. You can specify that particular devices are discovered but not instantiated to an AOC definition (such devices are discovered but are *not* represented in the network topology and their details are *not* sent to MODEL). You can also restrict devices from being discovered (SNMP access to any such device is not attempted).
- Another reason for scoping the discovery is that it restricts the amount of data Network Manager tries to download from the routing tables of individual routers. Without this restriction, if Network Manager finds a router that knows the routing table for the whole Internet, then discovery takes a very long time to complete.

Restriction: Network Manager does not support the IPv4–mapped IPv6 format and expects all IPv6 addresses to be in standard colon-separated IPv6 format. For example, Network Manager does not support an IPv4–mapped IPv6 address such as `::ffff:192.0.2.128`. Instead this address must be entered as `::ffff:c000:280` (standard colon-separated IPv6 format).

Types of scoping

Network Manager offers several types of scoping.

You can enable the following types of scoping:

- You can include or exclude areas of your network (either subnet ranges or specific devices) in the discovery. Each configured area is referred to as a *zone*.

  Tip: If your subnet is sparsely populated, including individual routers is likely to result in a faster discovery than including the whole subnet.
- Zones can be specified within zones: within a given inclusion zone, you can specify devices or subnets that are not to be detected. These devices are not pinged by the Ping finder or interrogated by the discovery agents. For example, you can define an include scope zone consisting of the Class B subnet 1.2.0.0/16, and within that zone you can specify an exclude scope zone consisting of the Class C subnet 1.2.3.0/24. Finally, within the exclude scope zone you could specify an include scope zone 1.2.3.128/26.
- You can include or exclude non-IP devices, such as layer 1 optical devices, using a filter.
- You can configure a filter that determines whether a discovered device is interrogated for connectivity information.
- You can configure multicast scoping. This enables you to configure which multicast subnets to use as scopes for your multicast discovery.

Defining discovery zones to restrict discovery

To restrict the discovery, you must define discovery zones. You can define discovery zones in several ways.

Choose one of the following methods to define a discovery zone:

- Define discovery zones using the Discovery Configuration GUI.
- Construct zones by appending an OQL insert into the `scope.zones` table with the `DiscoScope.cfg` configuration file. This method is for more experienced users.

Note: If nothing is specified in the `scope.zones` table then everything is considered to be in scope.

For each zone you must specify the following information:

- The type of network protocol used by the zone, although currently only IP is supported. You can define as many zones as necessary. Multiple zones can also be defined within the same insert.

- The action to be taken for the zone, where `m_Action=1` means include in the discovery, and `m_Action=2` means exclude. You can define both inclusion and exclusion zones. The action to be taken in the smallest zone overrides the actions in the larger zones.
- A list of varbinds (`name=value`) that define the present discovery zone.

Examples of discovery zones

Use this information to understand how inclusion and exclusion zones define which devices are included in and excluded from the discovery. The diagrams in this section show that the smallest zone takes priority.

Inclusion zone

Use this information to understand how an inclusion zone works.

The following figure shows an inclusion zone, 172.18.1.0/24.

- An entity with IP address 172.18.1.33 is within the inclusion zone and is therefore discovered.
- An entity with IP address 172.18.2.33 is outside the inclusion zone and is therefore not discovered.

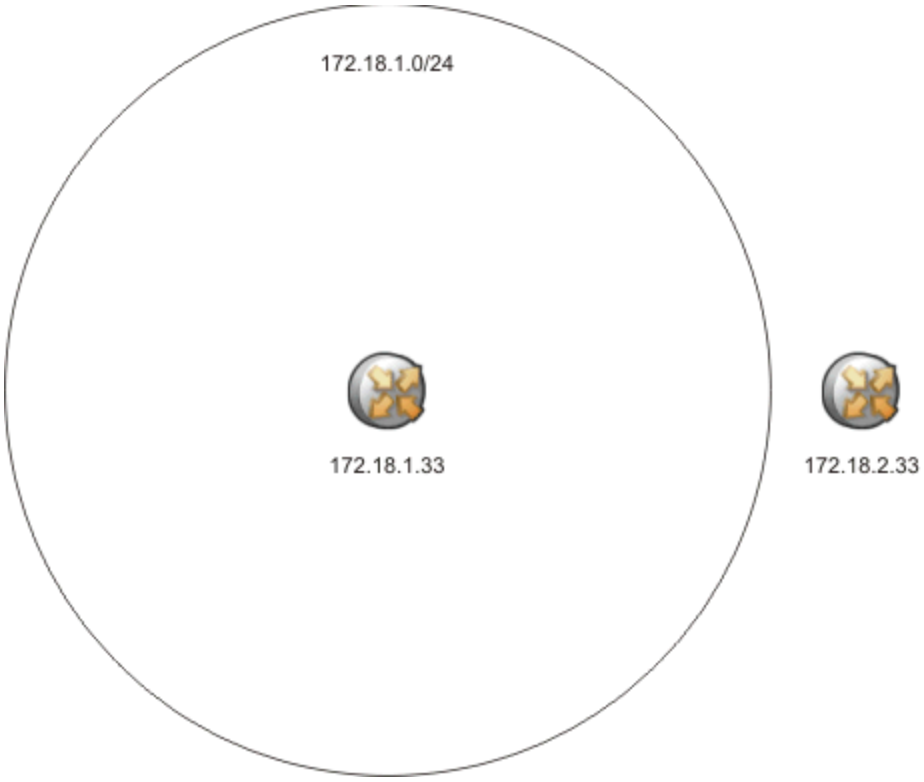


Figure 1. Inclusion zone, with device inside and outside the zone.

Exclusion zone within inclusion zone

Use this information to understand how scoping works for an exclusion zone within an inclusion zone.

The following figure shows an exclusion zone within an inclusion zone.

- An entity with IP address 172.18.1.33 is within the exclusion zone and is therefore not discovered.
- An entity with IP address 172.18.2.33 is within the inclusion zone and is therefore discovered.
- An entity with IP address 172.19.3.2 is outside of the inclusion zone and is therefore not discovered.

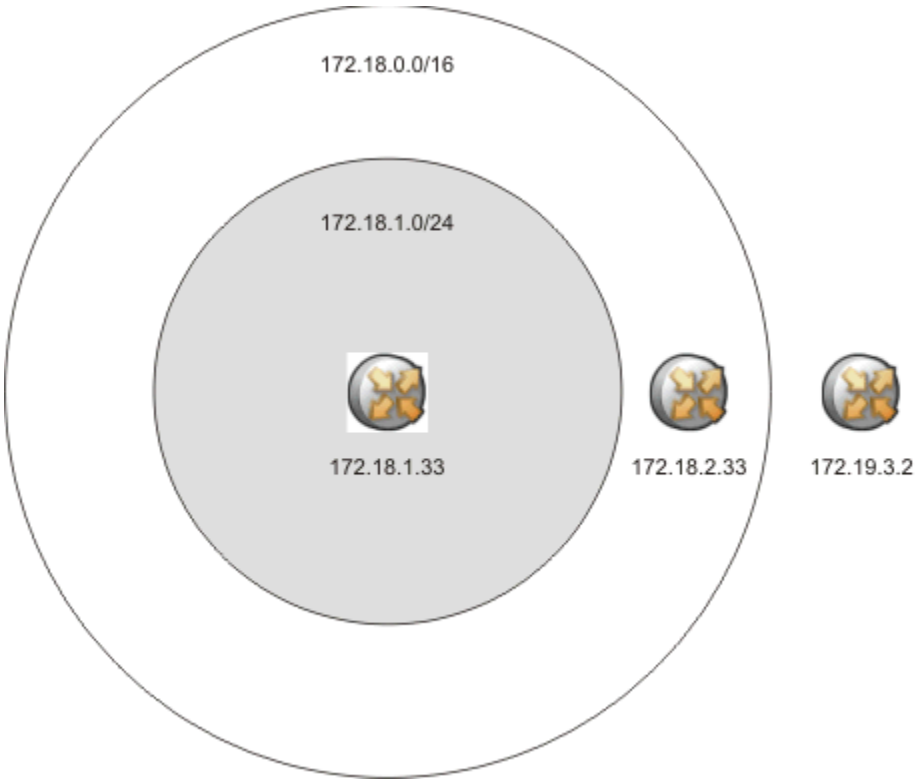


Figure 2. Exclusion zone within an inclusion zone

Inclusion, exclusion, and inclusion zone

Use this information to understand how device scoping works for an inclusion zone within an exclusion zone within an inclusion zone.

The following figure shows an inclusion zone within an exclusion zone within an inclusion zone.

- An entity with IP address 172.18.1.33 is within the inclusion zone and is therefore discovered.
- An entity with IP address 172.18.2.33 is within the exclusion zone and is therefore not discovered.
- An entity with IP address 172.19.3.2 is within the inclusion zone and is therefore discovered.
- An entity with IP address 172.19.3.2 is outside of the inclusion zone and is therefore not discovered.

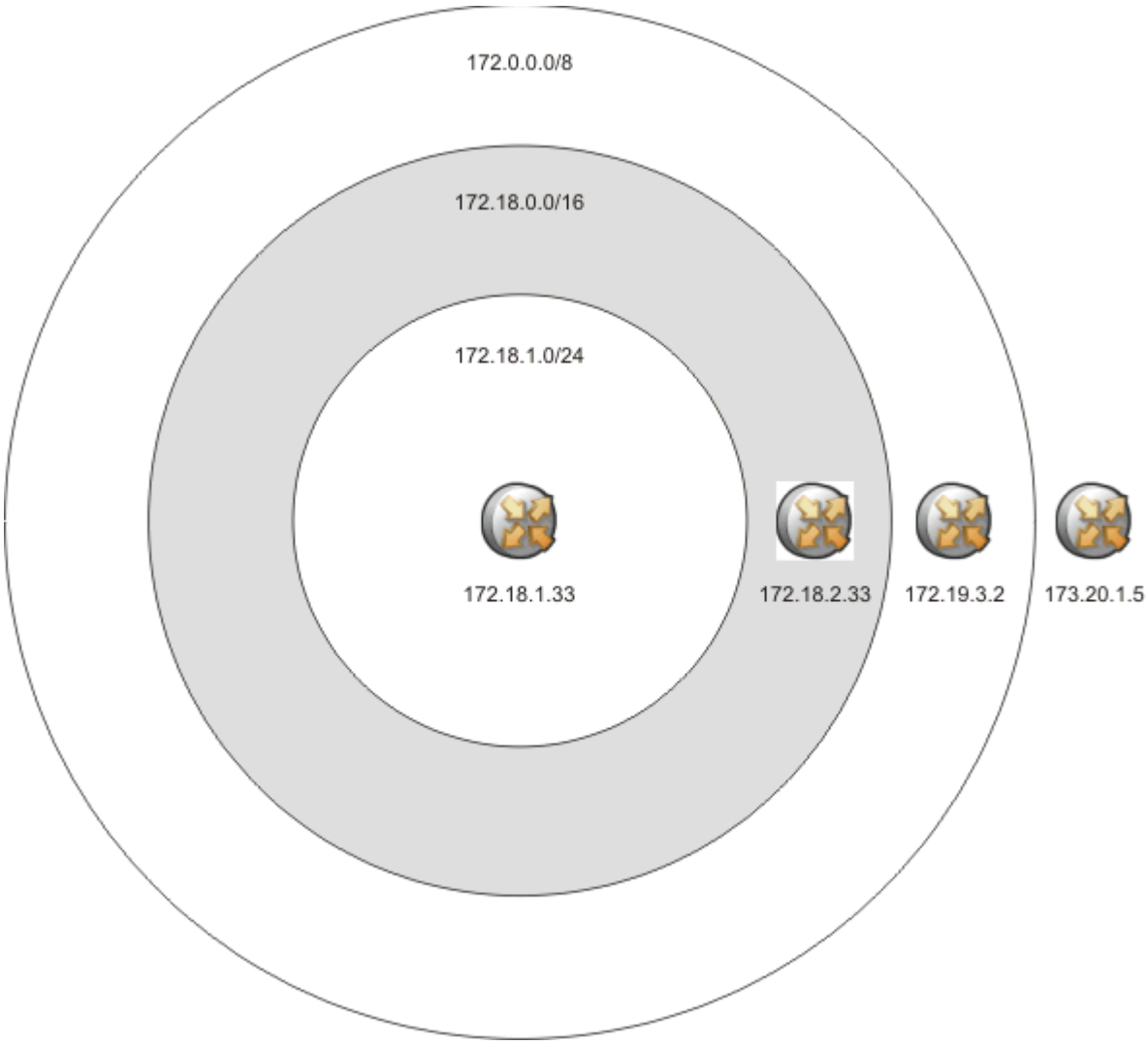


Figure 3. Inclusion zone within an exclusion zone within an inclusion zone

Exclusion zone

Use this information to understand how scoping works for an exclusion zone works.

The following figure shows an exclusion zone, 172.18.1.0/24.

- An entity with IP address 172.18.1.33 is within the exclusion zone and is therefore not discovered.
- An entity with IP address 172.18.2.33 is outside of the exclusion zone and is therefore discovered.

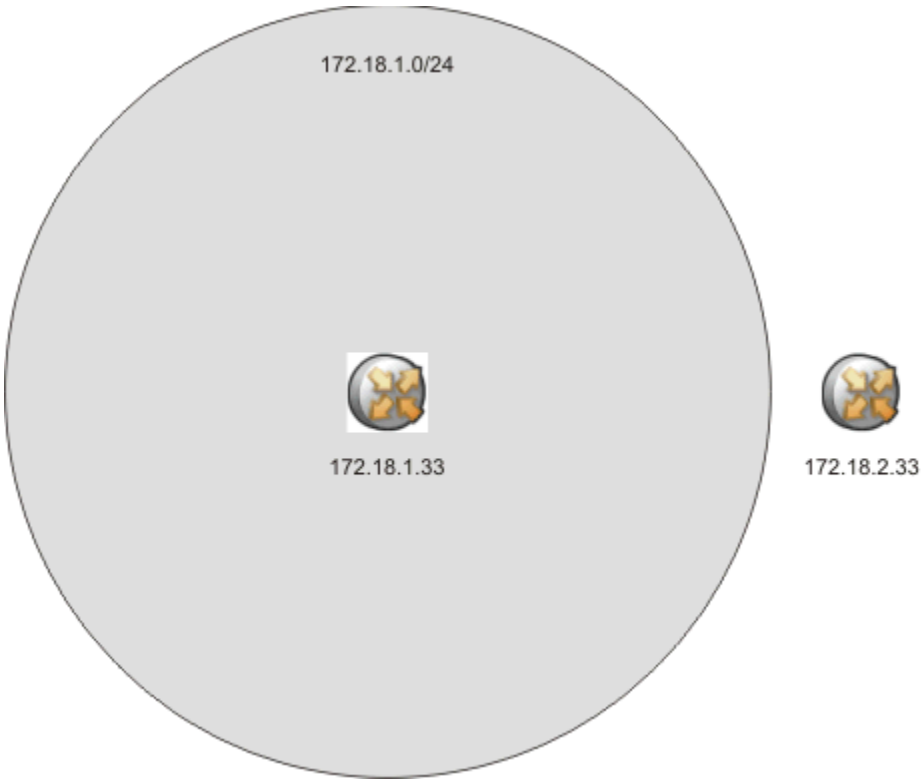


Figure 4. Exclusion zone

Inclusion zone within exclusion zone

Use this information to understand how scoping works for an inclusion zone within an exclusion zone.

The following figure shows an inclusion zone within an exclusion zone.

- An entity with IP address 172.18.1.33 is within the inclusion zone and is therefore discovered.
- An entity with IP address 172.18.2.33 is within the exclusion zone and is therefore not discovered.
- An entity with IP address 172.19.3.2 is outside of the exclusion zone and is therefore discovered.

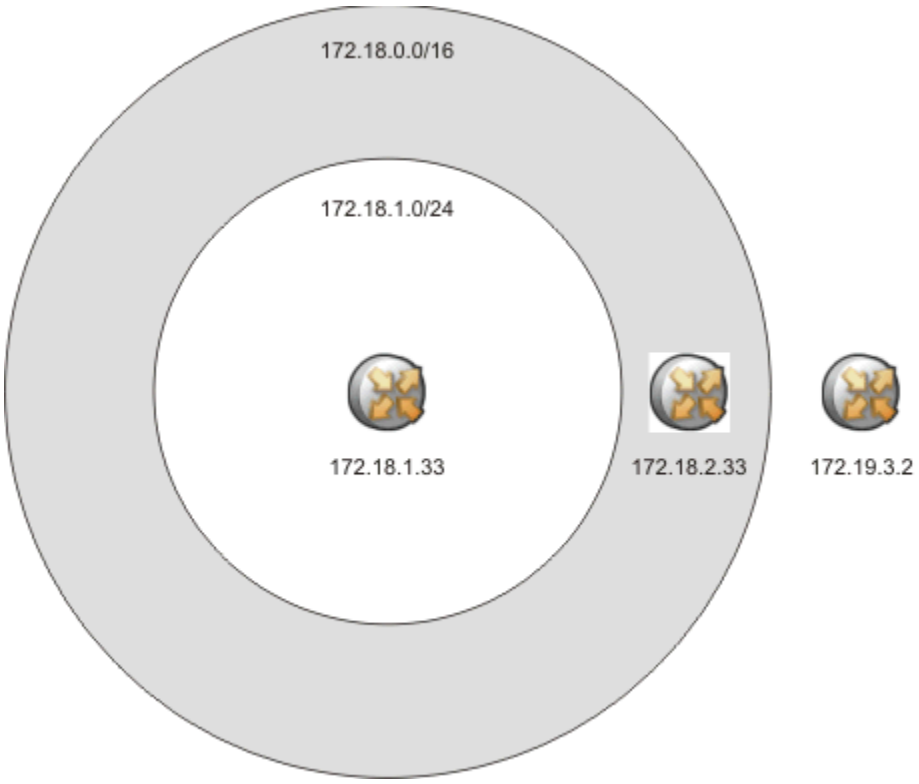


Figure 5. Inclusion zone within an exclusion zone

Exclusion, inclusion, and exclusion zone

Use this information to understand how device scoping works for an exclusion zone within an inclusion zone within an exclusion zone.

The following figure shows an exclusion zone within an inclusion zone within an exclusion zone.

- An entity with IP address 172.18.1.33 is within the exclusion zone and is therefore not discovered.
- An entity with IP address 172.18.2.33 is within the inclusion zone and is therefore discovered.
- An entity with IP address 172.19.3.2 is within the exclusion zone and is therefore not discovered.
- An entity with IP address 173.20.1.5 is outside of the exclusion zone and is therefore discovered.

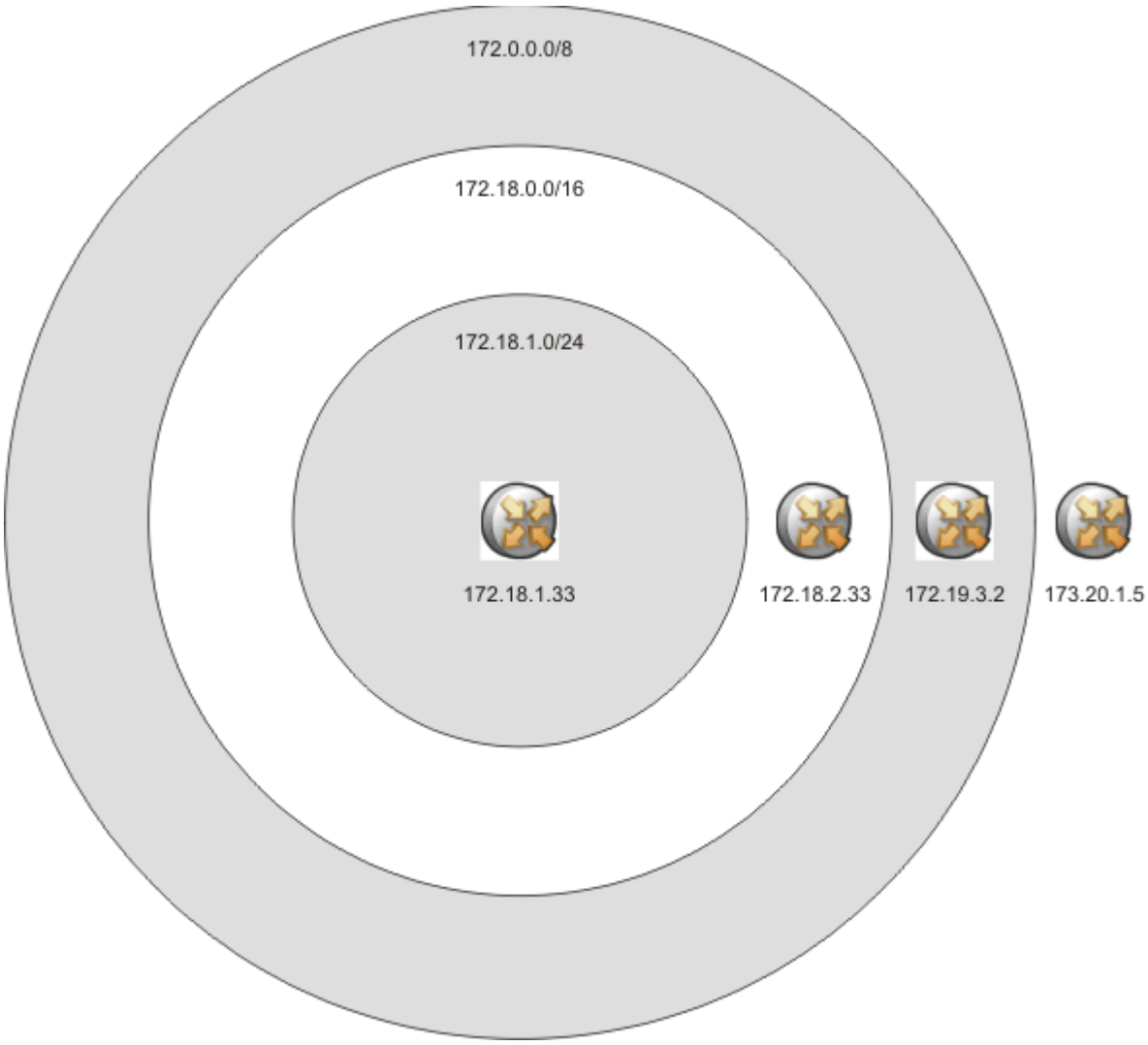


Figure 6. Exclusion zone within an inclusion zone within an exclusion zone

Defining a single inclusion zone

Use this information to understand how to define a single inclusion zone.

About this task

An inclusion zone is any zone that you want to be discovered by the Discovery engine, ncp_disco. The following example insert defines a single inclusion zone for the IP protocol and associates the zone with a subnet. This zone applies to every device within the specified subnet address.

```
insert into scope.zones
(
        m_Protocol,
        m_Action,
        m_Zones
)
values
(
        1,
        1,
        [
            {
                m_Subnet="172.16.1.0",
                m_NetMask=24
            }
        ]
);
```

The above insert defines an IP inclusion zone for the 172.16.1.0 subnet with a netmask of 24.

Defining multiple inclusion zones

You can define multiple inclusion zones in the scope.zones table.

About this task

In the following example, three different IP inclusion zones are defined within a single insert.

```
insert into scope.zones
(
    m_Protocol,
    m_Action,
    m_Zones
)
values
(
        1,
        1,
        [
              {
                    m_Subnet="172.16.1.0",
                    m_NetMask=24
              },
              {
                    m_Subnet="172.16.2.*"
              },
              {
                    m_Subnet="172.16.3.0",
                    m_NetMask=255.255.255.0
              }
        ]
);
```

The above example defines three different IP inclusion zones each using a different syntax to define the subnet mask. Network Manager discovers:

Procedure

- Any device that falls within the 172.16.1.0 subnet (with a subnet mask of 24, that is, 24 bits turned on and 8 bits turned off, which implies a netmask of 255.255.255.0).
- Any device with an IP address starting with "172.16.2", that is, in the 172.16.2.0 subnet with a mask of 255.255.255.0.
- Any device that falls within the 172.16.3.0 subnet with a mask of 255.255.255.0.

Defining exclusion zones

Use this information to understand how to define exclusion zones.

About this task

An exclusion zone is any zone that you do not want to be discovered by the Discovery engine, ncp_disco. Multiple exclusion zones can be created within the same insert in the same way as inclusion zones. The following example insert defines a single exclusion zone for the IP protocol, and associates the zone with a subnet.

```
insert into scope.zones
(m_Protocol, m_Action, m_Zones)
values (1, 2,[{m_Subnet="172.16.1.0", m_NetMask=24]);
```

Defining zones for NAT address spaces

Use this information to understand how to define zones for NAT address spaces.

About this task

Inclusion and exclusion zones can be customized for individual NAT domains, using the m_AddressSpace column of the scope.zones table. The following example insert defines an inclusion zone within a particular NAT domain.

```
insert into scope.zones
(
    m_Protocol, m_Action, m_Zones, m_AddressSpace
)
values
(
        1,
        1,
        [
            {
                m_Subnet="172.16.2.*",
            }
        ],
        "NATDomain1"
);
```

Restricting detection of specific devices

You can specify individual devices or subnets that the Ping finder is not to ping. Since the Ping finder has not detected their existence, these devices or subnets are not interrogated by the discovery agents.

Restricting interrogation of specific devices

You can exclude specific devices from the discovery by specifying that after their initial detection those devices are not to be interrogated further for connectivity information.

About this task

In order to specify devices that must not be detected, you must append an OQL insert into the scope.detectionFilter table to the DiscoScope.cfg configuration file. There must only be one insert into the scope.detectionFilter table per protocol. Multiple conditions must be defined within a single insert.

Within the detectionFilter table you must specify:

- The type of network protocol. Currently only IP is supported.
- The filter condition(s). Only devices that pass this filter, that is, for which the filter evaluates true, are further investigated. If no filter is specified, all devices are passed through the detection filter.

A stitcher tests each discovered device against the filter condition in the scope.detectionFilter table, and the outcome of this test determines whether the device is discovered. Since the process flow of the discovery is fully configurable, you can configure this stitcher to act at any time during the discovery process. By default, the stitcher performs the conditional test on the device details returned by the Details agent. Your filter must therefore be based on the columns of the Details.returns table. The following examples show how you might configure the detection filter.



Attention: Multiple examples are shown for illustrative purposes only. In practice you must ensure that there is only one insert into the scope.detectionFilter table per protocol.

Example

Preventing the Interrogation of a Device Based on the IP Address

In this example only devices that do not have the IP address 10.10.63.234 are interrogated further.

```
insert into scope.detectionFilter
(
        m_Protocol, m_Filter
)
```

```
values
(
        1,
        "( ( m_UniqueAddress <> '10.10.63.234' ) )"
);
```

Interrogation Restriction Based on Object ID

The following example shows how you would prevent the further interrogation of devices that match a given Object ID. The OQL not like clause indicates that only devices that pass the filter (that is, devices for which the OID is not like 1.3.6.1.4.1.*) are interrogated further.

```
insert into scope.detectionFilter
(
        m_Protocol,
        m_Filter
)
values
(
        1,
        "(
                ( m_ObjectId not like '1\.3\.6\.1\.4\.1\..*' )
        )"
);
```

The backslash must be used in the above insert to escape the ., which would otherwise be treated as a wildcard.

Combining Multiple Filter Conditions

The following example shows how you can combine filter conditions within a single OQL insert. This example ensures that only devices that do not have the specified OID and do not have the specified IP address are detected.

```
insert into scope.detectionFilter
(
        m_Protocol,
        m_Filter
)
values
(
        1,
        "(
                ( m_ObjectId not like '1\.3\.6\.1\.4\.1\..*' )
                AND
                ( m_UniqueAddress <> '10.10.63.234' )
        )"
);
```

Configuring the Filter Condition

Although you can configure the filter condition to test any of the columns in the Details.returns table, you may need to use the IP address as the basis for the filter if you need to restrict the detection of a particular device. If the device does not grant SNMP access to the Details agent, the Details agent may not be able to retrieve MIB variables such as the Object ID. However, you are guaranteed the return of at least the IP address when the device is detected.

Seeds

Configure seeds to specify the locations from which to begin discovering devices. Discovery seeds can be IP addresses, or subnet addresses.

You can specify seeds in several ways:

- Using the Ping finder: You specify the IP addresses or subnet addresses to discover first.
- Using the File finder: You provide one or more files that each contain a list of IP addresses or subnet addresses.
- Using the Database finder: You specify connection details for an internal or external database that contains details of devices to discover. You also specify an SQL query to run to retrieve the device details from the database. An advantage of using the Database finder over the File finder is that you can

extract device data directly from a third-party database, without having to extract the device data as a flat file.

Note: The Database finder method is only configurable from the command line

- Using the Collector finder: You specify one or more ports used to access independent collector processes.

Tip: To restrict discovery to a list of specific devices, seed the discovery with a list of devices using the File finder or the Ping finder, and disable feedback in the **Advanced** tab of the **Network Discovery Configuration GUI**.

Device access

Configure device access by specifying SNMP community strings and Telnet parameters so that the system can access network devices.

Configure device access as follows:

- Specify SNMP community strings so that Network Manager can access and interrogate the network devices that use SNMP. Network Manager supports SNMP v1, v2, and v3,
- Specify Telnet parameters so that Network Manager can access and interrogate the network devices that use Telnet.

Agents

Use discovery agents to retrieve information about devices on the network. Select the right agents for your discovery depending on your network type.

Discovery agents retrieve device details and investigate device connectivity. They can also report the existence of new devices by finding new connections when investigating device connectivity. The discovery agents can be used for specialized tasks. For example, the ARP Cache discovery agent populates the Helper Server database with IP address to MAC address mappings.

Default agents are provided for the type of discovery you want to perform, for example a layer 2 or layer 3 discovery. You can select different sets of agents for full discoveries and for partial discoveries. The agents vary because connectivity information varies with the technology of the hardware in the network.

Device filters

Use prediscovery filters to increase the efficiency of discovery.

After you have defined the scope of your discovery using the **Scope** tab, it is possible to restrict the scope using filters. For example, you might want to maintain the scope zones that you defined earlier but restrict the scope based on location (for example, New York hardware only) or based on hardware supplier (for example, Cisco devices only).

You can filter out devices based on a variety of criteria, including location, technology, and manufacturer.

By default, the discovery filters do not filter out the Network Manager host because it usually also serves as the polling station for root cause analysis. For root cause analysis to work correctly, the polling station, and hence the Network Manager host machine, must be part of the topology.

For more information on root cause analysis, see the *IBM Tivoli Network Manager IP Edition Administration Guide* and the *IBM Tivoli Network Manager User Guide*.

If you do need to filter out the Network Manager host, then you need to modify the following stitchers and remove the sections of code, indicated by comments, that prevent the Network Manager host from being filtered. The stitchers are DetectionFilter.stch and InstantiationFilter.stch.

Prediscovery filter

You might want to apply this filter to sensitive devices that you do not want to poll. A device might be considered sensitive because there is a security risk involved in polling the device, or because polling might cause the device to overload.

Prediscovery filters prevent the discovery from retrieving detailed data or connectivity data from the device and prevent discovered devices from being polled for connectivity information. Only devices matching the prediscovery filter are fully discovered. If no prediscovery filter is defined, then all devices within the scope are discovered.

Prediscovery filters provide a mechanism to base discovery on complex IP ranges that cannot be easily defined in the **Scope** tab. It can be used to filter out devices based on their sysObjectId value. Default filters exist to filter out end nodes, printers, and similar devices. You can create quite complex multiple filters, which makes this feature very powerful but try to ensure that filters are designed so that they can be easily maintained. The filter acts on the fields of the `Details.returns` OQL table in the discovery (Disco) service, so you can use fields other than IP addresses, such as m_ObjectId (equivalent to sysObjectId). A device must pass all filters to be discovered.

Important: Design the filter logic so that you do not need to modify the prediscovery filters every time you add new scopes.

You can configure the filter condition to test against any of the columns in the Details.returns table. But, you might need to use the IP address (m_UniqueAddress) as the basis for the filter to restrict the detection of a particular device. If the device does not grant SNMP access to the Details agent, the Details agent might not be able to retrieve MIB variables such as the Object ID. However, you are guaranteed the return of at least the IP address when the device is detected.

You can define multiple prediscovery filters. Filters are combined automatically using a Boolean AND expression. All criteria defined in all filters must be matched.

Post-discovery filter

The post-discovery filter is not available when discovery is running in dNCIM mode.

SNMP interface filtering

You can filter the SNMP data retrieved from devices by the discovery process by configuring SNMP interface filters. You can configure SNMP interface filters only on the command line.

Why use SNMP interface filtering

Sometimes a device or a class of devices returns too much MIB data. For example, if virtual devices have a large number of interfaces, discovering them can take a long time. In order to speed up discovery of such devices, you can use SNMP interface filters to reduce the number of interfaces that are retrieved by the SNMP helper.

How SNMP interface filtering works

When discovery agents, Perl scripts, or the **SNMP MIB Browser** request SNMP information, the SNMP helper retrieves the information from network devices. SNMP interface filters define rows in MIB tables that the SNMP helper retrieves. The SNMP helper retrieves a subset of the information that would have been returned without a filter, and sends it to the process that requested the SNMP information. SNMP interface filters can also define entire tables that are not to be retrieved by the SNMP Helper.

SNMP interface filters are only applied to requests for full walks of MIB tables. SNMP Get or GetNext requests for specific interfaces within a MIB table are not filtered.

The devices that should have the filter applied to them are defined in the *device filter*. If a device filter is defined, any request for SNMP information for a device is first checked against the device filter. Only devices which pass the filter are then checked for interface filtering.

The filter can filter on multiple rows of a table. The first time that a filtered table is accessed, one or more columns in the table are walked. All subsequent requests for SNMP walks of that table return only interfaces that match the filter.

Including multiple tables with dependent filters

You can also define *dependent filters*. If an SNMP interface filter *Filter 1* is defined on Table A, then a second, dependent filter *Filter 2* can be defined on Table B. SNMP information in Table B that relates to the same interface is also retrieved.

In order to define a dependent filter, in addition to a filter being defined on Table A, one or more of the following conditions must be true:

- Table A and Table B have equivalent indexes.
- The index of Table A is a value in Table B.

If Table A and Table B have exactly the same index, then you do not need to define a dependent filter. Information from Table B is automatically retrieved according to the filter defined on Table A.

When you can use SNMP interface filtering

You can use SNMP interface filtering on any SNMP MIB table that is keyed on ifIndex. For example, you filtering on ifTable or ifXTable allows filtering on values such as ifType and ifDescr.

Restriction: Filtering on any SNMP MIB variable other than interfaces is not supported. You can, however, block access to any table using the m_InstanceFilterTable option.

The following example extract shows the definition for the ipAddrTable from the `NCHOME/precision/mibs/RFC1213.mib` file:

```
          -- the IP address table

          -- The IP address table contains this entity's IP addressing
          -- information.

          ipAddrTable OBJECT-TYPE
              SYNTAX  SEQUENCE OF IpAddrEntry
              ACCESS  not-accessible
              STATUS  mandatory
              DESCRIPTION
                      "The table of addressing information relevant to
                      this entity's IP addresses."
              ::= { ip 20 }
```

The syntax "SEQUENCE OF" defines this as a table. You can find out what tables are defined in the MIB by searching for this string. The following example shows the output of a search run on UNIX:

```
% grep 'SEQUENCE OF'  RFC1213.mib
              SYNTAX  SEQUENCE OF IfEntry
              SYNTAX  SEQUENCE OF AtEntry
              SYNTAX  SEQUENCE OF IpAddrEntry
              SYNTAX  SEQUENCE OF IpRouteEntry
              SYNTAX  SEQUENCE OF IpNetToMediaEntry
              SYNTAX  SEQUENCE OF TcpConnEntry
              SYNTAX  SEQUENCE OF UdpEntry
              SYNTAX  SEQUENCE OF EgpNeighEntry
```

Domain Name System

Configure DNS to enable the discovery to access DNS services that are used to perform domain name lookups.

You can configure three types of Domain Name System:

DNS server
A server on the network that is dedicated to performing domain name resolution.

File
The name of a file held on the Network Manager host that contains IP addresses and host names in lookup table format.

System
The local DNS system on the Network Manager machine.

Network address translation

Configure data for NAT gateways in your network.

NAT gateways provide mappings between private IP address in your network and public device IP addresses. You can enable the system to discover devices within private address spaces by configuring data for NAT gateways.

Advanced settings

Configure advanced discovery settings to increase the speed of the discovery, and balance the speed with the load on the server. Generally, a faster discovery results in more memory usage on the server. Advanced settings control features of the discovery such as concurrent processes and timeouts.

Note: Modify the advanced settings only if you are an experienced Network Manager user.

You can configure the following advanced discovery settings:

Finder parameters:
Finders are discovery subsystems that discover devices on the network. You can configure parameters such as timeouts, number of retries, and number of threads for the finders.

Helper parameters
Helpers are discovery applications used by agents to retrieve information from devices. You can configure parameters such as timeouts, number of retries, and number of threads for the helpers.

Other parameters
You can configure complex discovery settings, such as enabling caching of discovery tables, VLAN modeling, File finder verification, and parameters that affect the speed of partial discovery.

Most of the advanced discovery parameters are optional.

Context-sensitive discovery

If you need to discover devices that support multiple contexts, you must run a context-sensitive discovery. For example, a device that uses separate SNMP contexts to provide access to its virtual routers. Context-sensitive discovery ensures correct representation of context-accessible virtual devices. Always check that your particular device type is supported for discovery.

In a context-sensitive discovery, information about a device is passed from the `returns` table of the Details agent to the `despatch` table of the relevant Context agent.

The Context agents use the filters in the `.agnt` files of the agents to determine which devices to process. This approach is true for all discovery agents. If the device is not of a type that supports context-accessible virtual devices, that is, does not need context-sensitive processing, it is passed directly to the Associated Address agent.

Helpers

The helpers are specialized applications that retrieve information from the network on demand. The default helper configuration is sufficient for most networks. However, you might decide to alter the configuration for several reasons.

Configuring the Helper System can speed up network discovery, but is recommended for experienced users.

Although the discovery agents retrieve connectivity information, they do not have any direct interaction with the network. Instead, they retrieve connectivity information through the Helper System, which consists of a Helper Server and various helpers.

Reasons to configure the helpers include:

- To speed up the discovery process, you could reduce the helper timeouts and number of retries.
- If you have a very reliable network in which devices respond quickly, you can specify a small default timeout.
- You might want to change the default timeouts for the SNMP and Telnet helpers if you have many devices that either do not respond to SNMP and Telnet or that are set up not to respond to Telnet or SNMP access. A large default timeout would therefore mean that the helpers wait for a long time for responses they never receive.
- To reduce the amount of network traffic caused by a discovery, you could increase the timeout and disable broadcast and multicast pinging.

Collectors

Collectors are like agents, except they do not discover information from devices, but collect it from Element Management Systems (EMSs). Enable and configure the right collectors for your discovery if you use an EMS.

Collectors are separate processes that are started independently of the ncp_disco process. Collectors connect to an EMS or parse files from an EMS to collect data about the devices managed by the EMS.

Each Collector that you need to run must be configured to connect to the appropriate EMS.

Specialized discoveries

You can configure the system to perform more complex discoveries, such as Multiprotocol Label Switching (MPLS) and Network Address Translation (NAT) discovery.

Specialized discoveries include:

Element Management System (EMS) discoveries
Collect topology data from Element Management Systems and integrates this data into the discovered topology.

MPLS discoveries
Discover layer 3 virtual private networks (VPNs) and enhanced layer 2 VPNs running across MPLS core networks.

NAT discoveries
Discover NAT gateway devices to retrieve data on devices in private address spaces.

Third-party discoveries:
Discover intervening provider networks as a "third-party" object on multiple networks that run across a provider network (for example, enterprise VPNs across a provider MPLS core network).

Chapter 13. Configuring network discovery

Configure how your network is discovered, including which kinds of devices you want to discover, and where the boundaries of the discovery should be.

About this task

Network Manager provides tools for discovering your network using a phased approach.

- Use the Discovery Configuration Wizard to perform initial discoveries. The wizard provides a guided discovery and makes configuration choices for you based on the answers that you provide as you work through the wizard.
- Use the Discovery Configuration GUI to perform subsequent discoveries. Using the GUI you can configure detailed discovery settings, including scope, seeds, community strings, agent selection and many other configuration settings.

Note: You can also configure a discovery using the discovery configuration files and the command line. However, you should configure discovery this way only if you are an experienced Network Manager user and you understand the different aspects of discovery, including the discovery processes, phases, stages, Helpers, agents, stitchers and traps.

For information on how to manually edit a discovered topology following discovery, see the *IBM Tivoli Network Manager User Guide*.

Planning for discovery

Before configuring and running a discovery, you should check several system settings, parameters, and requirements.

About this task

The following notes help you plan for the discovery.

Saving changes in the Network Discovery Configuration GUI
To save configuration changes that you have made during a session, remember to click the **Save** button before you log out, close the browser window, or close the Network Discovery Configuration tab. It is good practice to click **Save** as you move from tab to tab.

Operating system
Ensure that the host on which Network Manager is running is fully patched with the latest patches.

Discovery scope
Consider the following questions and points related to the discovery scope:

- Is the positioning of the Network Manager host within the network?
- Is the host positioned to interrogate all devices that you want to include in your discovery?
- Consider all necessary networks, subnetworks and determine the associated netmasks.
- Are there any parts of the network that you want to exclude?
- Gather all relevant SNMP community strings for the devices within the scope

Routing
Ensure that each of the networks and subnetworks to be discovered is reachable using the ICMP process. If necessary, add routes to Network Manager host machine using the **route add** command.

Access control lists
Network Manager uses several protocols that might need to pass through firewalls. These protocols are ICMP, SNMP, DNS, ARP, SSH, and TELNET. To ensure that Network Manager can access devices behind these firewalls, advise the relevant firewall administrators to prepare the firewalls.

© Copyright IBM Corp. 2006, 2021

Root cause analysis

To perform root cause analysis on devices within a topology, the discovery must identify all the devices on which you might want to perform root cause analysis. In addition, the Network Manager polling station must be known. 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. For more information on root cause analysis, see the *Network Manager Monitoring and RCA Guide*.

Configuring standard discoveries

You can configure discovery using the Discovery Configuration Wizard, using the Discovery Configuration GUI, or using the command line.

Discovering the network using the wizard

The discovery configuration wizard is for users who have limited experience in configuring discoveries.

About this task

Important: If you want to keep discovery configuration settings that you made previously using the GUI, do not use the wizard. The discovery configuration wizard overwrites all previous settings.

Launching the wizard

Select a domain and launch the wizard to start configuring and running a discovery.

About this task

To launch the wizard, complete these steps.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. At the top left of the Network Discovery Configuration tab, select the domain in which you want to run a discovery from the **Domain** menu.
3. Click the wizard button to the right of the **Domain** menu.

Choosing a scoped or unscoped discovery

The **Discovery Scope** window provides the option of a scoped or unscoped discovery.

About this task

To choose a scoped or unscoped discovery, complete these steps.

Restriction: Network Manager does not support the IPv4–mapped IPv6 format and expects all IPv6 addresses to be in standard colon-separated IPv6 format. For example, Network Manager does not support an IPv4–mapped IPv6 address such as `::ffff:192.0.2.128`. Instead enter this address as `::ffff:c000:280` (standard colon-separated IPv6 format).

Procedure

1. Select **Scoped** or **Unscoped**.

   Scoped

   A scoped discovery restricts the discovery to a certain part of your network. To specify a scoped discovery, tell the wizard which area of the network to restrict the discovery to, and assign IP addresses or subnets as seeds to ping to begin the discovery.

Unscoped

An unscoped discovery attempts to discover your entire network. However, you still need to assign IP addresses or subnets as seeds to ping to begin the discovery.



Attention: If there are routes out of your network to the Internet, an unscoped discovery will find these routes and start to discover parts of the Internet.

2. If you selected **Scoped**, specify which area of the network to which to restrict the discovery.

   Specify one or more subnets to use for both scoping and seeding by clicking **New** and typing an IP address and a netmask.

   Restriction: For performance reasons, only IPV4 addresses will be pinged. To ping IPV6 addresses use the Seed tab in the Discovery Configuration GUI.
3. If you selected the **Unscoped** option, specify the seeds to use for your unscoped discovery.

   Click **New...** and specify one or more IP addresses.

Configuring SNMP access using the wizard

Specify address-specific, network-specific, or global community strings on the **SNMP Community Strings** window.

About this task

For SNMP version 3, you can also specify passwords for community strings.

To configure SNMP access, complete these steps.

Procedure

1. For each SNMP community string and associated password that you want to define:
   a) Click the **New** icon above the **SNMP Community Strings** table to display the **SNMP Password Properties** window.
   b) Specify address-specific, subnet-specific, or global SNMP community strings, and supply passwords for these community strings in the case of SNMP V3.

      You might need to enter a community string more than once. For example, enter a string for SNMP V1, enter another string for SNMP V2, and another string for SNMP V3.

      Specifying community strings by subnets results in a more efficient and faster discovery.

      Restriction: It is best practice not to use the at symbol (@) in community strings. Using this symbol in a community string can cause problems connecting to devices at discovery time.
2. Use the up and down arrow keys to put the community strings in order of most frequently expected use. Put more frequently used community strings at the top.

Configuring Telnet access using the wizard

On the **Telnet Access** window, set the Telnet access parameters.

About this task

To configure Telnet access, complete these steps.

Procedure

1. After you have specified SNMP community strings, click the **New** icon on the **Telnet Access** window.
2. For each set of Telnet-accessible devices for which you want to define prompts and passwords, click **New**.

3. On the **Telnet Passwords** window, specify a set of Telnet-accessible devices (all devices, all devices within a specified subnet or a single IP address) together with prompts, login IDs, and login passwords for this set of devices.

Specifying type of discovery

On the **Discovery Type** window, specify the type of discovery: a Layer 3 or a Layer 2 discovery.

About this task

A Layer 3 discovery is quicker, but the results of a Layer 3 discovery cannot be used for root cause analysis. A Layer 2 discovery is more detailed and the results can be used for root cause analysis.

To specify discovery type, complete these steps.

Procedure

1. On the **Discovery Type** window, specify a Layer 2 or Layer 3 discovery.
2. If you selected **Layer 3**, the **End Node Discovery** window is displayed.

   On the **End Node Discovery** window, you can filter out end node devices such as workstations and printers. You can also filter out devices with no SNMP access.

   Tip: Filtering out all end nodes in networks that have large numbers of end nodes can lead to speed and performance improvements in your discovery.
3. If you selected **Layer 2 and Layer 3**, the **VLAN Modelling** window is displayed.

   In the **VLAN Modelling** window, you configure the discovery to model VLANs in the resulting topology. This enables VLANs to be considered when performing root cause analysis. VLANs are a Layer 2 concept and VLAN modelling is required for Layer 2 discoveries only. Specify whether you want to model VLANs. When you have specified an option, click **Next** to display the **End Node Discovery** window.

Optimizing the discovery

On the **Discovery Optimization** window, optimize the discovery for connectivity, richness of information, and speed.

About this task

To optimize the discovery, complete these steps.

Procedure

1. Provide varying amounts of connectivity information by selecting one of these options.

   Best possible connectivity accuracy and richness of information
   This option provides comprehensive connectivity information between switches, end nodes and routers as well as detailed information on each device discovered. However, the discovery might require a significant amount of time to complete.

   Best possible connectivity accuracy but prefer speed to richness of information
   This option provides comprehensive connectivity information. However, the discovery provides less detailed information on each device discovered in order to provide a faster discovery.

   Rich device information but prefer speed to accurate connectivity
   This option provides detailed information on each device discovered. However, the discovery provides less detailed connectivity information to provide a faster discovery. For example, the discovery might provide information on how switches are connected to each other, but it might not provide information on connectivity between switches and end nodes or between switches and routers.

Note: This option is more suitable if you want to gather inventory data instead of perform root cause analysis (RCA). RCA is dependent on accurate connectivity data.

Fastest discovery time
This option focuses on the speed of the discovery. However, limited connectivity information is provided as well as less detailed information on each single device.

2. If you select either of the first two options, this means that accurate connectivity is important. The **Network Reliability** window is displayed.
3. If you select either of the last two options, this means that you are willing to compromise on the accuracy of connectivity information to ensure a faster discovery. In this case, the wizard asks how much of your network is made up of Cisco devices. If a large proportion of the network is made up of Cisco devices, then the wizard can switch off agents that discover connectivity for non-Cisco devices, thereby speeding up the discovery significantly. The **Cisco Hardware** window is displayed.
   a) Specify how much of your network is made up of Cisco hardware by selecting one of these options.

   All of it
   This option directs the wizard to run the Cisco Discovery Protocol (CDP).

   Most of it, Some of it, Don't know
   This option directs the wizard to run the CDP. However, if you chose a Layer 2 and Layer 3 discovery or if you indicated that you want to exclude end nodes from the discovery, then this option invokes the Spanning Tree Protocol (STP) as well as CDP.

   None of it
   This option specifies that neither the CDP nor the STP protocol is used.

   b) When you have selected one of these options, click **Next**.
   c) If your response to the Cisco hardware question was **All of it** or **None of it**, then the **Network Reliability** window is displayed.
   d) If your response to the Cisco hardware question was **Most of it**, **Some of it**, or **Don't know**, then the **Spanning Tree Protocol** window is displayed, on which you specify whether the spanning tree protocol is enabled on all network switches.

Indicating the reliability of your network

On the **Network Reliability** window, select a description of the reliability of your network in responding to pings and SNMP requests. The description directs the wizard to establish the length of timeouts.

About this task

To describe the reliability of your network, choose one of these options that correspond to the reliability of your network when responding to ping and SNMP requests.

Very reliable
This description states that the network must be reliable when responding to ping and SNMP requests. Select this option to allow the wizard to apply very short timeouts, without any retries. This option is appropriate for a very reliable network and results in fast discoveries. If you requested Fastest possible discovery time in the Discovery Optimization window, then the timeouts set by this option are even shorter.

Quite reliable
This description states that the network must be reliable for the most part when responding to ping and SNMP requests. Select this option to allow the wizard to apply slightly longer timeouts, with a single retry for both SNMP and ping requests.

Not very reliable
This description states that the network does not necessarily need to be reliable when responding to ping and SNMP requests. Select this option to allow the wizard to apply longer timeouts and two retries for SNMP requests and ping requests. Longer timeouts are suitable for less reliable networks.

Reviewing the configuration

On the **Configuration Summary** window, review your settings. You can also save the settings here, and, optionally, start the discovery with the settings that you configured.

About this task

To review your configuration settings, complete these steps.

Procedure

1. Review the settings on the **Configuration Summary** window.

   Click any of the links to return to the relevant window to modify the settings as appropriate.
2. When you are satisfied with the discovery settings, select one of these options.
   - Select **Start Discovery**, to use the discovery configuration settings that you specified, and then click **Finish** to start the discovery.
   - If you do not select **Start Discovery**, the discovery settings are saved when you click **Finish**.

Discovering the network using the GUI

To perform a custom discovery, complete the tabs on the **Network Discovery Configuration** page. On these tabs, you can configure more complex discovery parameters than by using the **Discovery Configuration Wizard**.

Before you begin

Remember: To save configuration changes that you have made during a session, click the **Save** button before you log out, close the browser window, or refresh or navigate away from the Network Discovery Configuration tab. It is good practice to click **Save** as you move from tab to tab.

About this task

The parameters that you can set on the tabs of the **Network Discovery Configuration** are described in the topics that follow.

Most of the parameters that you can set on the **Network Discovery Configuration** page are optional.

For the discovery to run, at a minimum you must specify the following parameters:

- One seed device
- The correct SNMP community strings for the network to be discovered.

If any of the tabs contain data, this data is from earlier configurations. The data is held in the relevant discovery configuration file.

Scoping discovery

To scope the discovery for IP-based devices and subnets, define the zones of the network (that is, subnet ranges) that you want to include in the discovery, and the zones that you want to exclude.

About this task

You can define as many zones as necessary. You can add new zones, or edit or delete existing zones. Zones can be specified within zones: within a given inclusion zone, you can specify devices or subnets that are not to be detected. These devices are not pinged by the Ping finder or interrogated by the discovery agents. For example, you can define an include scope zone consisting of the Class B subnet 1.2.0.0/16, and within that zone you can specify an exclude scope zone consisting of the Class C subnet 1.2.3.0/24. Finally, within the exclude scope zone you could specify an include scope zone 1.2.3.128/26.

Note: This procedure can only be used to configure scoping for IP-based entities. Non-IP entities such as layer 1 optical devices and certain radio access network devices must be scoped by configuring a prediscovery filter in the **Filters** tab.

To scope the discovery:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click **Scope**.
4. To add a new scope zone, click **New**  .
   The **Scope Properties** page is displayed.
5. Complete the fields as follows and then click **OK**.

   Action
   Define the subnet range as an inclusion zone or exclusion zone. If the subnet range is an inclusion zone that you intend to ping during the discovery, click **Add to Ping Seed List**. Clicking this option automatically adds the devices in the scope zone as a discovery seed devices.

   Restriction: The `Add to Ping Seed List` option is not available for IPv6 scope zones. This prevents ping sweeping of IPv6 subnets, which can potentially contain billions of devices to be pinged. Ping sweeping of IPv6 subnets can therefore result in a non-terminating discovery.
6. To edit an existing scope zone, click the required row. On the **Scope Properties** page, edit the properties as described in step "5" on page 183.
7. To delete an existing scope zone, select the **Select** check box next to the required row or rows and click **Delete**  .
8. Click **Save**  .

What to do next

If you are performing NAT address mapping, you must configure the NAT gateways and return to the **Scope** tab to set the address mapping.

Seeding discovery

To seed the discovery, provide the starting points from which to look for devices.

Before you begin

For the discovery to run, at a minimum you must specify the following parameters:

- One seed device
- The correct SNMP community strings for the network to be discovered.

About this task

Use the following methods to seed the discovery:

Ping finder
Seed the Ping finder with a device or subnet address at which the finder begins looking for devices. You can specify seeds for the Ping finder and save these seeds. You can separately decide whether to activate the Ping finder for the discovery.

File finder
Seed the File finder with a text file on the Network Manager host to which you have read access. This file must be a structured text file that contains the seeds in the form of IP addresses and device

names in columns. You usually use a file that already exists on the Network Manager host. However, if you want to create a new file to hold the seeds, you need to have write permissions for the directory where you want to write the file.

Note: It is also possible to seed the discovery using the Database finder; however, this method is only available from the command line. Seed the Database finder by specifying a query that reads a database in order to retrieve a list of devices to find on the network.

When running an IPv6 discovery, ensure that the following conditions are met:

- There is at least one IPv6 seed device within each IPv6 scope.
- If you specify an IPv6 subnet as a seed, then ensure that the subnet is small by specifying a high value for the netmask.

By default, the Ping finder and File finder are switched on.

To seed the discovery:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click **Seed**.
4. Optional: To switch off the Ping finder or File finder clear the **Use Ping Finder in Discovery** or **Use File Finder in Discovery** check boxes.
5. Add or edit a ping seed:

   - To add a new ping seed, click **New**  .
   - To edit an existing ping seed, click the required entry in the list.

   The **Ping Seed Properties** page is displayed.
6. Complete the fields as follows and click **OK**.

   Seed by:
   Select one of the following options:

   IP
   Type an IP address.

   Subnet
   Specify a subnet and type the number of netmask bits. The **Netmask** field is automatically updated.

   Restriction: Network Manager does not support the IPv4–mapped IPv6 format and expects all IPv6 addresses to be in standard colon-separated IPv6 format. For example, Network Manager does not support an IPv4–mapped IPv6 address such as `::ffff:192.0.2.128`. Instead enter this address as `::ffff:c000:280` (standard colon-separated IPv6 format).

   Timeout
   Specify the time in milliseconds to wait for a reply from a pinged address before timing out.

   Retries
   Specify the number of times a device is to be repinged.
7. To delete an existing ping seed, select the **Select** check box next to the required row and click **Delete**  .
8. Add or edit a file seed:

   - To add a new file seed to the File finder, click **New**  .
   - To edit an existing file seed, click the required entry in the list.

   The **File Seed Properties** page is displayed.

9. Complete the fields as follows and click **OK**.

   Filename
   Specify the path to the file on the host workstation that contains the seed data.

   Delimiter
   Specify the column delimiter. Use a regular expression if required. For example, if the Name and IP columns are separated by one or more tabs, then insert `[ tab_space ]+`, where *tab_space* is an actual tab character. To produce this tab character, create a tab in a text editor, copy the tab and paste it into the field.

   IP Column
   Type the column number of the column that contains the IP addresses of the seed devices.

   Name Column
   Type the column number of the column that contains the device names of the seed devices.
10. To delete an existing file seed, select the **Select** check box next to the required row and click **Delete**  .
11. Click **Save**  .

What to do next

You can also seed a discovery by using the *Collector finder*. The collector finder retrieves topology data from an EMS. Topology data is collected by EMS collectors, which are software modules that retrieve topology data held in an EMS database, convert this data to XML format and pass this data to Network Manager to stitch into the topology. You must seed the Collector finder to enable Network Manager to find one or more EMS collectors.

IPv6 subnet mask sizes

There are potentially billions of devices to be pinged within a single IPv6 subnet. To ensure that discovery completes, you must specify a sufficiently large netmask if you specify an IPv6 subnet as a ping seed.

The following table provides examples of IPv6 subnet mask sizes configured within ping seeds and the corresponding estimated time required to ping the devices in the subnet. The estimated times are based on spacing the pings by 100 ms between pings. This table shows that it is best to limit the size of IPv6 subnet masks in your subnet seeds.

Table 38. Ping response times for IPv6 subnet masks

IPv6 subnet mask size	Number of IPv6 addresses in subnet	Estimated ping time for subnet
120	256	26 seconds
112	65536	1 hour 48 minutes
100	268 million	Approximately 8.5 years

The time estimates shown in the table refer to time taken to ping all the seeds in a subnet seed specified for the Ping finder. It would take longer to complete the discovery as there will be many more devices to ping within the discovery scope.

Configuring device access

Specify SNMP community strings and Telnet access information to enable helpers and Network Manager polling to access devices on your network.

Before you begin

Note the following information about the SNMP helper and Telnet helper:

SNMP helper
You must specify SNMP community strings for the SNMP helper and polling operations to access devices on your network. You might need to enter a community string more than once. For example, once for SNMP V1, once for SNMP V2, and once for SNMP V3.

Telnet helper
Enter the relevant device prompts, login ID, and password for the Telnet helper and the discovery agents that use Telnet. You can configure Telnet-privileged access properties. The privileged access mode allows commands to be run that might change the configuration of the device. By default, when the discovery accesses the device using Telnet, access is granted in user mode. This mode allows the running of basic commands only, such as those commands that show the status of the system. This default access mode is a safety feature to prevent the discovery making any device configuration modifications without an explicit change to privileged mode.

Community strings and Telnet access data can be *global*, which means that the discovery tries the community string for every device it encounters, or restricted to specific subnets (that is, used only on devices within a specific subnet), or even restricted to specific devices. Specifying community strings and Telnet access data by subnet results in a more efficient and faster discovery. In general, the more specific the credentials, the faster the discovery will determine the correct credentials.

Note: Speed of discovery related to community string settings in the GUI only affects the initial discoveries. Once Network Manager has identified the correct community strings, it stores this information in the NCMONITOR relational database. Subsequent discoveries access this database for SNMP cmmunity strings and other SNMP-related device access information.

About this task

For the discovery to run, at a minimum you must specify the following parameters:

- One seed device
- The correct SNMP community strings for the network to be discovered.

To configure device access:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click **Passwords**.
4. To add a new SNMP community string, click **New**  .
   The **SNMP Password Properties** page is displayed.
5. Complete the fields as follows and then click **OK**:

   Community String
   Type a name. When you save the community string, the name is encrypted, but on the GUI, the value is always displayed unencrypted. For speed of discovery, order the SNMP strings by frequency, with the most common strings first.

   Restriction: It is best practice not to use the at symbol (@) in community strings. Using this symbol in a community string can cause problems connecting to devices at discovery time.

   Apply to
   The discovery completes more quickly if you specify the correct scope of the community strings. Select one of the following options:

   All Devices
   Select this option if the community string is global.

   IP Address
   Select this option if the community string is specific to an IP address, and type the IP address.

Subnet
Select this option if the community string is specific to a subnet. Type the required subnet and specify the number of netmask bits. The **Netmask** field is automatically updated.

SNMP Version
Specify the version of SNMP for this SNMP community. If you specify SNMP V3, complete the following additional fields:

Security Name
Type a name.

Level
Specify the required level of authentication and privacy.

NoAuthNoPriv,
Select this option for SNMP communities that have no authentication or private key. In this case there is no need to specify any passwords.

AuthNoPriv
Select this option for SNMP communities that have an authentication key but no private key. Then specify a password in the **Auth Password** field.

AuthPriv
Select this option for SNMP communities that have both an authentication and a private key. Then specify passwords in the **Auth Password** and **Private Password** fields.

Auth Type
Specify the type of encryption for the authentication password.

Restriction:

The MD5 encryption option is not available if you are running a FIPS 140–2 installation of Network Manager.

By default, the GUI does not use any cryptographic routines that are excluded from a FIPS140-2 installation, regardless of the installation status of the core server. If you want to configure SNMP discovery options to enable MD5 and DES, set `tnm.fips.mode=false` in the `tnm.properties` file.

Priv Type
Specify the type of encryption for the privacy password.

Restriction: The DES encryption option is not available if you are running a FIPS 140–2 installation of Network Manager.

SNMP Port
Specify the required port.

Timeout
Specify the time in milliseconds to wait for a reply before timing out.

Note: The administrator can control the maximum timeout that can be set using this field, by configuring the `discoconfig.oobl.passwords.snmp.timeout.max` property in the `discoconfig.properties` file.

Retries
Specify how many times you want the SNMP helper and polling operations to attempt to access a device.

Note: The administrator can control the maximum number of retries that can be set using this field, by configuring the `discoconfig.oobl.passwords.snmp.retries.max` property in the `discoconfig.properties` file.

6. Click **Move Up** and **Move Down** to arrange the SNMP community strings. Put the most frequently used strings at the top of the list.
7. Click **Save**.

8. To add Telnet access information, click **New.** 

   The **Telnet Password Properties** page is displayed.
9. Complete the fields as follows:

   Apply to
   Select one of the following options:

   All devices
   Select this option if the data applies globally.

   IP address
   Select this option if the string is specific to a device, and type the IP address of the device.

   Subnet
   Select this option if the string is specific to a subnet. Type the required subnet and specify the number of netmask bits. The **Netmask** field is automatically updated.

   Username prompt
   Type the prompt that you want to be displayed at login. If you do not know the exact format of the prompt. use a regular expression.

   Username
   Type the user name.

   Password prompt
   Type the prompt that you want to be displayed when the password is required at login. If you do not know the exact format of the prompt, use a regular expression.

   Password
   Type the password.

   Console prompt
   Type the prompt that is displayed when you log in. If you do not know the exact format of the prompt, use a regular expression.

   Access port
   Specify the port on which the Telnet helper and discovery agents attempt to access devices.

   Timeout
   Specify the time in milliseconds to wait for a reply before timing out.

   Note: The administrator can control the maximum timeout that can be set using this field, by configuring the `discoconfig.oobl.passwords.telnet.timeout.max` property in the `discoconfig.properties` file.

   Use SSH
   Select this option to configure the Telnet Helper to use the Secure Shell (SSH) program.
10. Optional: To configure Telnet-privileged access mode properties:
    a) Click **Advanced**.

    The **Telnet Privileged Access Mode Properties** page is displayed.

    b) Complete the fields as follows and then click **OK**:

    Command
    Type the command required to enter Telnet-privileged access mode. This command is typically `enable`.

    Password Prompt
    Type the prompt that you want to be displayed when the password is required at login. If you do not know the exact format of the prompt, use a regular expression.

    Password
    Type the required password for privileged mode.

    Console Prompt
    Type the prompt that is displayed when you log in. If you do not know the exact format of the prompt, use a regular expression.

Commands requiring mode:
Specify the commands that you want to make accessible from privileged mode. To add new commands, click **New...** and type the command in the **Priv command** field. The following commands are required to run in `enable` mode:

- **`show run`**
- **`show mac-address-table`**
- **`show ip nat translation`**

11. Click **OK**. Click **Save**  .

Results

When you save the Telnet password settings, the following passwords are automatically encrypted:

- Telnet password
- Telnet privileged mode password (if specified)

When you save the password settings, the following passwords are automatically encrypted:

- SNMP community string
- SNMP authentication password
- SNMP private password

What to do next

If required, change the SNMP and Telnet encryption settings. For example, you can change the encryption key file, or switch off encryption.

Activating agents

You must enable the appropriate agents for the discovery you want to perform. You can specify agents for a full discovery or for a partial discovery.

About this task

You can speed up the time taken for a partial discovery by selecting only those agents essential to discover the new or modified devices. You might want to run a partial discovery if you know that new devices have been added to the network or that engineers have been working on a device and have added or removed components of this device.

Note: The more agents you run, the more data is retrieved from the network, and the slower the discovery.

To activate agents:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click one of the following tabs, based on your requirements:

Tab	Description
Full Discovery Agents	Select agents from this tab to run a full discovery.
Partial Discovery Agents	Select agents from this tab to run a partial discovery.

Tab	Description
	Note: The **Reset** button in the **Partial Discovery Agents** window sets the partial agents to match the settings defined in the **Full Discovery Agents** window.

The **Agents List** is displayed, showing all available discovery agents for the selected discovery option.

4. Select the check boxes next to the required agents.

   For descriptions of the agents, select an agent name.

   To select all agents required for a layer 3 discovery, select the **Layer 3** checkbox. To select all agents required for a layers 2 and 3 discovery, select the **Full Layer 2 and Layer 3 Discovery** checkbox.

5. Click **Save**  .

   If you have selected a invalid combination of agents, or a combination that might result in an inefficient discovery, a warning is displayed.

6. If applicable, follow the steps displayed in the warning:

   - If you selected an agent that must be run in conjunction with another agent or agents, the warning indicates that the additional agents will be selected as applicable. Click **OK** to select the agents, or click **Cancel**.
   - If you selected an agent that cannot be run in conjunction with another agent or agents, the warning indicates that the redundant agents will be automatically deselected. Click **OK** to deselect the recommended agent or click **Cancel**.

Setting discovery filters

Use filters to filter out devices either before discovery or after discovery. You can filter out devices based on a variety of criteria, including location, technology, and manufacturer. Filters provide additional restrictions to those defined in the scope zones.

About this task

A filter is made up of one or more filter conditions. Filter conditions are defined in Object Query Language (OQL). You can add the following types of filter:

Prediscovery filters

Prediscovery filters prevent discovered devices from being polled for connectivity information.

Note: Use prediscovery filters to configure scope for non-IP devices in the network, as shown in the examples below.

Post discovery filters

Post-discovery filters prevent discovered devices from being passed to MODEL.

Note: To ensure that alerts are not raised for interfaces that are excluded by the post discovery filter, you must set the `RaiseAlertsForUnknownInterfaces` variable. To this, perform the following steps:

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

```
update config.properties set RaiseAlertsForUnknownInterfaces = 0;
```

The steps for adding, editing, and deleting filters are identical for both types.

To set the discovery filters:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.

2. From the **Domain** list, select the required domain.
3. Click **Filters**.
4. To use a filter in the discovery, select a filter from the **Available filters** list and click **Select Filter**.

   The filter is added to the **Selected Pre-discovery Filter** field or the **Selected Post-discovery Filter** field, depending on the type of filter.
5. To remove a filter, select a filter from the **Selected Pre-Discovery Filters** or **Selected Post-Discovery Filters** list and click **Deselect Filter**.
6. To add a new filter, or edit an existing filter, click **Filter Library**.

   The **Filter Library** page is displayed.
7. Add or edit the filter as follows:

Action	Instructions
Add a new filter	Click **Add** and type the required name in the **Name** field.
Edit an existing filter	Select the required filter from the list.

8. On the **Basic tab**, build the filter conditions as follows:
   a) Select the required field and comparator.
   b) Type the value for comparison with the selected field.

      See “Configuring discovery filters” on page 191 for an example.
   c) Click **Add New Row** or **Delete This Row** to add or remove rows.
   d) Select **All** to combine multiple conditions in an AND relationship, or **Any** combine the conditions in an OR relationship.
   e) Click **Save**.
9. Optional: On the **Advanced** tab, type the required SQL WHERE clauses. For multiple conditions, use an AND or an OR relationship as appropriate. Click **Save**.

   Note: The filter is actually based on standard OQL formatting, though the GUI refers to the SQL clause.
10. Click **Close** to close the **Filter Library**, then click **Save** to save your filter settings.

Configuring discovery filters

You can use a prediscovery filter to refine scoping. The prediscovery filter applies to chassis and interfaces, whereas the postdiscovery filter applies to all entities. Take care when defining postdiscovery filters so that you do not inadvertently filter out objects such as VPNs, cards, or subnets.

Only pass non-IP devices to discovery

The following example filter is provided by default. It ensures that only non-IP devices are passed to discovery and interrogated further. This filter would exclude all IP-based devices.:

```
m_Protocol = 4
```

Only pass non-IP devices to discovery and exclude the non-IP devices with a specified unique key

The following example insert ensures that only non-IP devices are passed to discovery and the non-IP devices that are passed must not include the specified string in their unique Element Management System (EMS) key.

```
( m_Protocol = 4 )
    AND
( m_UniqueAddress NOT LIKE 'LONDON' )
```

Exclude devices with a specified object ID

The following example shows a filter condition for a prediscovery filter that excludes devices with a specified object ID.

```
m_ObjectId not like '1\.3\.6\.1\.4\.1\.2\.3\.1\.'
```

Restricting instantiation of a chassis based on entity name

The following example postdiscovery filter restricts instantiation of a chassis and its contents.

```
BASENAME != 'jane'
```

Restricting instantiation of multiple chassis

The following example postdiscovery filter restricts instantiation of a chassis and its contents.

```
snmpSystem->SYSDESCR NOT LIKE ' device'
```

What to do next

For more information on OQL syntax, see the *IBM Tivoli Network Manager Reference*.

Available filter values

Use this reference information to familiarize yourself with the permissible values when you set discovery filters on the **Network Discovery Configuration** page.

Prediscovery filter values

When constructing a prediscovery filter, you can filter based on any of the fields in the Details.returns table. These fields are as follows:

m_AddressSpace

The name of the NAT address space to which the device belongs. This value is set in the translations.NATAddressSpaceIds table. If the discovery is not using NAT, or if the device is in the public domain, this value is NULL.

m_Description

Value of `sysDescr` MIB variable of the entity.

m_ExtraInfo

Any extra information.

m_HaveAccess

Flag indicating whether there is SNMP access to the device:

- 1: Have access
- 0: No access

m_LastRecord

A flag indicating whether this is the last record for this entity (that is, whether the entity has been completely processed):

- 1: True
- 0: False

m_ManagerId

Identifies the manager of the device. Takes the value "" if device is accessed directly.

m_Name

Unique name of an entity on the network.

m_ObjectId

Textual representation of the device class (an ASN.1 address).

m_Protocol

An integer representation of the IP protocol used by the presently-defined zone:

- 1: IPv4
- 2: IPv4 that has been through network address translation (NAT)
- 3: IPv6

m_UniqueAddress

The IP address of the discovered network entity.

m_UpdAgent

The agent that updated this device.

In addition, by using the **Advanced** tab, you can construct filter rows using any of the fields from within the m_ExtraInfo field.

Postdiscovery filter values

When constructing a post-discovery filter, you can filter based on any of the fields in the ncimCache database.

Configuring Domain Name System

You can specify the methods that the Domain Name System (DNS) Helpers use to perform domain name lookups.

About this task

Helpers are specialized applications that retrieve information from and about network devices for the discovery agents.

Each of method that you specify uses one of the following three domain methods:

DNS Server
A server on the network that is dedicated to performing domain name resolution.

File
The name of a file held on the Network Manager host that contains IP addresses and host names in lookup table format.

System
The local DNS system on the Network Manager host.

Tip: You can define as many methods as is necessary. You can change the order in which these methods are retrieved by the DNS Helper so that the most commonly accessed method is retrieved first. This enables a more effective use of resources during the discovery.

To configure DNSs:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click the **DNS** tab.
4. Add a new DNS helper, or edit an existing helper as follows:

   - To add a new DNS helper, click **New**  .
   - To edit an existing helper, click the name of the required helper.

   The **DNS Service Properties** page is displayed.

5. Complete the fields as follows and then click **OK**.

   Service Name
   Type the name of the method.

   Type
   Select one of the following options:

   System
   Choose this option to use the local DNS system on the Network Manager server. This is the default option.

   DNS Server
   Type the IP address of the required DNS server. In the **Timeout** field, specify the number of seconds to wait for a response from the DNS Server before timing out.

   File
   Type the name of the file that contains the domain lookup information. Specify the order in which this information appears in the lookup table by selecting one of the radio buttons:

   - **Name then IP**
   - **IP then Name**

   Domain Suffix
   Specify the suffix to append to each device name after the name is looked up. The specified domain suffix is only added if no domain suffix is present in the device name.

   Note: If you expect the discovery to return some or all devices names with domain suffixes already appended, then you can specify a list of expected domain suffixes. The domain suffix value specified in the **Domain Suffix** field is not appended to any device names returned by the discovery with these expected suffixes. To specify a list of expected domain suffixes, you must configure the DiscoDNSHelperSchema.cfg configuration file from the command line.
6. Repeat steps “4” on page 193 to “5” on page 194 to add or edit the required methods.
7. In the **Move** column, click **Move Up** and **Move Down** to arrange the methods in the order of most frequently-expected use, with the most frequently-used methods at the top.
8. Click **Save**.

Configuring NAT translation

To configure NAT translation to discover NAT environments, map the address-space identifier for a NAT domain to the IP address of the associated NAT gateway device.

About this task

After activating NAT, you must map the discovery scope zones to the NAT address spaces. You do this on the **Scope** tab.

If you select **Enable Network Address Translation (NAT) Support** in the **NAT** tab then you must configure at least one NAT gateway.

To configure NAT gateways:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click **NAT**.
4. Add a new NAT gateway, or edit an existing gateway:

   - To add a new NAT gateway, click **New**.

- To edit an existing NAT gateway, click the IP address in the required row.

The **NAT Gateway** page is displayed.

5. Complete the fields as follows and click **OK**:

   IP Address
   Type the public IP address of the NAT gateway device.

   Address Space
   Type the address space identifier that you want to use for the associated NAT domain.

6. Click **Save**  .
7. To activate NAT translation for the discovery, select **Enable Network Address Translation (NAT) Support**. Click **Save** and then map the discovery scope zones to the NAT address spaces:
   a) Click **Scope**.
   b) Click a scope zone to edit it.
      The **Scope Properties** page is displayed.
   c) In the **Address Space** field, enter the NAT address space and click **OK**.
      The **Address Space** field appears on the **Scope Properties** only after the **Enable Network Address Translation (NAT) Support** has been selected.
   d) Repeat the previous two steps for all the required scope zones.
   e) Click **Save**  .

   NAT Address Spaces Dynamic Distinct view is created automatically if **Enable Network Address Translation (NAT) Support** is turned on. Once discovery is complete, use the **Network Views** to visualize the NAT Address Spaces network view.

Configuring a multicast discovery

Configure a multicast discovery by enabling the required agents and scoping the discovery.

Enabling the multicast agents

To discover multicast groups, you must enable the appropriate agents and add the relevant SNMP community strings.

About this task

To enable the agents, complete the following steps.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click the **Full Discovery Agents** tab.
   The **Agents List** is displayed, showing all available discovery agents for the selected discovery option.
4. Click **Full Layer 2 and Layer 3 Discovery** > **Multicast**.
5. Select the check box next to the agents you want to enable.
   a) Enable the StandardPIM agent to discover protocol-independent multicast groups compliant with the RFC2934 PIM MIB.
   b) Enable the StandardIPMRoute agent to discover IP multicasting networks compliant with the RFC2932 IPMRoute MIB.
   c) Enable the StandardIGMP agent to discover multicast groups running the Internet Group Membership Protocol (IGMP).
6. Click **Save**  .

7. If you want to rediscover multicast groups, also enable the appropriate agents for partial discoveries.
8. Ensure that the SNMP community strings are configured correctly to access the devices in the multicast groups.

Scoping a multicast discovery

Configure which multicast groups and sources to discover using the **Multicast** tab.

About this task

To configure a multicast discovery, complete the following steps.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click **Multicast**.
4. In the **Multicast Groups** section, create a new multicast group or edit an existing group:
   - To create a new group to discover, click **New**  .
   - To edit an existing group, click the group name.

   The **Multicast Group Properties** page is displayed.
5. Define the scope properties using the following fields:

   Group Name
   Specify a name for this multicast group.

   PIM Mode
   Select whether to include or exclude Protocol Independent Multicast (PIM) data from the discovery. By default, PIM data is included.

   IPM Route Mode
   Select whether to include or exclude Internet Protocol Multicast (IPM) group data from the discovery. By default, IPM Group data is included.

   IGMP Mode
   Select whether to include or exclude Internet Group Management Protocol (IGMP) data from the discovery. By default, IGMP data is included.

   Protocol
   Only IPv4 is supported.

   Specify which Group subnets to add to Multicast Groups
   Use the following fields and buttons to add and delete group subnets:

   Subnet
   Enter a subnet and netmask for a group subnet to add to the multicast groups.

   Add
   Click **Add** the add this group.

   Delete
   Select a group subnet from the adjacent list and click **Delete** to delete the selected group.

   Note: Reserved multicast addresses are excluded from the scope by default.
6. Click **OK**.
7. To delete one or more groups, select the groups you want to delete and click the **Delete** button  .

   To select or deselect all groups, click the **Select All**  or **Deselect All**  button.
8. In the **Multicast Sources** section, create a new multicast source or edit an existing source.
   - To create a new source to discover, click **New**  .

- To edit an existing source, click the source name.

The **Multicast Source Properties** page is displayed.

9. Define the source properties using the following fields:

IPM Route Mode
Select whether to include or exclude the group:

- **Unknown - use default**
- **Include source**
- **Exclude source**

Protocol
Only IPv4 is supported.

Specify which Group subnets to add to Multicast Sources
Use the following fields and buttons to add and delete group subnets:

Subnet
Enter a subnet and netmask for a group subnet to add to the multicast sources.

Add
Click **Add** the add this group.

Delete
Select a group subnet from the adjacent list and click **Delete** to delete the selected group.

Specify which Source subnets to add to Multicast Sources
Use the following fields and buttons to add and delete group subnets:

Subnet
Enter a subnet and netmask for a sources subnet to add to the multicast sources.

Add
Click **Add** the add this group.

Delete
Select a source subnet from the adjacent list and click **Delete** to delete the selected soource.

10. Click **OK**.
11. To delete one or more groups, select the groups you want to delete and click the Delete button  .

    To select or deselect all groups, click the Select All  or Deselect All  button.
12. Click **Save**  .

Advanced discovery parameters

Advanced settings control features of the discovery such as concurrent processes and timeouts. Use these parameters to increase the speed of the discovery, but balance the speed with the load on the server. Generally, a faster discovery results in more memory usage on the server.

Set the advanced parameters on the **Advanced** tab of the **Network Discovery Configuration** page. After you have set the advanced parameters, click **Save**  .



Attention: Modify the advanced settings only if you are an experienced Network Manager user. If you modify the advanced parameters and the discovery does not work as expected, click **Reset** to restore the default settings.

"Advanced Finder Configuration" on page 198
"Advanced Ping Finder Configuration" on page 198
"Advanced Telnet Helper Configuration" on page 198
"Advanced SNMP Helper Configuration" on page 199
"Advanced DNS Helper Configuration" on page 199
"Advanced Discovery Configuration" on page 199

Advanced Finder Configuration

To set advanced parameters for the File finder, use the following field:

Concurrent File Finders
Specify the number of threads to be used by the File finder. Each thread can process a different seed file simultaneously. If you have many seed files and spare resources on the discovery server, more threads might result in a faster discovery. If you have only one seed file, increasing the number of threads has no effect.

Advanced Ping Finder Configuration

To set the advanced parameters for the Ping finder, use the following fields:

Concurrent Ping Finders
Specify the number of threads to be used by the Ping finder. Each thread processes one pingFinder.pingRules insert at a time. Increasing the number of threads does not speed up a single large ping sweep but might speed up feedback of many addresses. However, you must balance speed against the resources of your machine and the ability of the ping receiver to process the ping responses in a timely manner. If the number of threads is too high, the ping receiver will fall behind, resulting in false ping failures and a loss of device discovery.

Studies have shown that the default number of 10 threads is optimal for most situations. You can gradually increase the number of threads and monitor the number of ping failures and make a note of time savings. Depending on the resources available, at a certain point the benefits start to decrease as resources are overloaded.

Default Timeout
Specify the maximum time, in milliseconds, to wait for a reply from a pinged address. If you know that network latency is low, a reduced wait time might result in a faster discovery. A value that is too low for your network might result in devices not being discovered.

Default Number of Retries
Specify the number of times a device is to be pinged again following a failed initial ping.

Inter-Ping Time
Specify the interval in milliseconds between ping attempts made against the devices contained in a list or subnet. If network traffic resulting from the discovery is not an issue, this value can be reduced.

Allow Broadcast Pinging
To enable broadcast address pinging, select this check box.

Allow Multicast Pinging
To enable multicast address pinging, select this check box.

Advanced Telnet Helper Configuration

To set advanced parameters for the Telnet helper, use the following fields:

Concurrent Telnet Helpers
Specify the number of threads to be used by the Telnet helper. If you have many devices from which you want to access data using Telnet or SSH then increasing this value might result in a faster discovery. Typical examples of such devices are Catalyst switches, MPLS devices, and NAT gateways. If you change this value, be sure that your system is configured to allow at least this number of concurrent Telnet sessions.

Default Timeout
Specify the maximum time, in milliseconds, to wait for access to a device.

Number of Retries
Specify the number of times to try to connect to the device following a failed initial connection attempt.

Tip: You can also configure some other advanced settings in the `DiscoTelnetHelperSchema.cfg` file.

Advanced SNMP Helper Configuration

To advanced parameters for the SNMP helper, use the following fields:

Concurrent SNMP Helpers
Specify the number of threads to be used by the helper. If you have many devices with SNMP access and spare resources on the discovery server, more threads might result in a faster discovery. If you change this value, make sure that your system is configured to allow at least this number of concurrent SNMP sessions. This value must be more than the number of threads used by the Details discovery agent.

Timeout
Specify the maximum time, milliseconds, to wait for access to a device.

Number of Retries
Specify the number of attempts to retrieve one or more SNMP variables from a device after a failed initial attempt.

GetNext Slowdown
Specify the delay, in milliseconds, between each SNMP GetNext request. The **m_GetNextSlowDown** parameter is applied when the number of separate GetNext requests issued in order to retrieve a non-scalar SNMP variable exceeds the value of the **m_GetNextBoundary** parameter.

GetNext Boundary
Specify the minimum number of GetNext requests to be issued when a non-scalar SNMP variable is retrieved from a device. The **m_GetNextBoundary** parameter is applied before the delay specified by the **m_GetNextSlowDown** parameter is introduced.

Advanced DNS Helper Configuration

To set advanced parameters for the DNS helper, use the following fields:

Concurrent DNS Helpers
Specify the number of threads to be used by the helper. If you change this value, be sure that your system is configured to allow at least this number of concurrent DNS sessions.

Default Timeout
Specify the maximum time, in milliseconds, to wait for a response from a device.

Advanced Discovery Configuration

To specify advanced feedback control, ping verification, and further advanced discovery parameters, use the following fields:

Enable Feedback Control
Specify whether to enable feedback control. Feedback means that the data returned by agents is used by the discovery to find other devices. Examples of feedback data include the IP addresses of remote neighbors, and addresses in the subnet within which a local neighbor exists.

No Feedback
Feedback is switched off for all discoveries and rediscoveries. Only the devices specified to the finders are discovered. This option means that discoveries and rediscoveries complete in the quickest possible time. However, the resulting network topology is incomplete unless you specify all devices that you want to discover as seeds.

Tip: Switch feedback off if you want to discover only a list of certain devices. Specify the devices you want to discover as seeds.

Feedback
Feedback is switched on for full discoveries, full rediscoveries, and partial rediscoveries. This option provides a complete topology in all situations but takes the longest.

Feedback Only on Full
This setting is on by default. Feedback is switched on for full discoveries and full rediscoveries, but switched off for partial rediscoveries.

Enabling Ping Verification
Specify whether the discovery checks for pingable interfaces. If a device is not pingable, the device is not polled for alerts.

Don't Check Pingability
None of the discovered interfaces are checked for whether they can be pinged. Interfaces are polled regardless of whether they are pingable at discovery.

Check Pingability
After discovery, every discovered interface is checked for whether they can be pinged. The check is run against the details.returns table. Interfaces that have an entry in this table are pingable. Interfaces that do not have an entry in this table are not pingable. The pingable interfaces are marked to be polled.

Detect Best Setting
This setting is on by default. If feedback control has been enabled, after discovery, every discovered interface is checked for whether they can be pinged. The check is run against the details.returns table. Interfaces that have an entry in this table are pingable. Interfaces that do not have an entry in this table are not pingable. The pingable interfaces are marked to be polled.

Restriction: This option works only when you select one of the following options from the **Enable Feedback Control** list: `Feedback` or `Feedback only on Full`.

Enable 'Allow Virtual'
Specify how you want the discovery to handle virtual IP addresses: [1].

Don't Allow Virtual
Does not discover virtual IP addresses.

Allow Virtual
Discovers virtual IP addresses. This setting is on by default.

Allow if in scope.special
Discovers virtual IP addresses only if the address is defined in the scope.special table. This table defines management IP addresses.

Enable VLAN Modelling
Enable this setting to model VLANs in this discovery. If you enable VLAN modeling, then you can partition discovered topologies based on VLAN membership. Disabling VLAN modeling reduces discovery time.

Enable SysName Naming
Enable this setting to name devices using the value of the SNMP sysName variable as the main source of naming information. The sysName variable must be set and must be unique within the network. Enabling this setting has no impact on the discovery time, because the sysName variable is retrieved by the Details agent by default.

Important:

If devices have previously been discovered with SysName naming disabled, duplicate devices can be displayed on the next discovery. For example, the same device can appear twice, once with the IP address and once with the new name. To avoid duplicate device entries, set the LingerTime of all devices in the topology to zero before running your next discovery. Log into the OQL Service Provider using the following command:

```
ncp_oql -domain NCOMS -service Model
```

[1] Devices are typically discovered using IP addresses retrieved by the AssocAddress agent. If a device is discovered using an IP address that was not retrieved by the AssocAddress agent, then the IP address is probably non-standard. This type of IP address is called a *virtual IP address*. Examples of virtual IP addresses are HSRP and VRRP addresses, which are shared by multiple devices for fault tolerance. Other examples include certain management interfaces that might be on a single device but do not appear in the IP table for security reasons. Virtual IP addresses include management addresses. A management address is an IP address whose only role is to manage the device. Management addresses are often on a separate network isolated from the customer traffic. These addresses are defined in the scope.special table.

Run the following command to set the LingerTime to zero:

```
update ncimCache.lingerTime set lingerTime = {LINGERTIME=0};
go
```

Enable Caching of Discovery Tables

Enable this setting to cache data during the discovery process in order to enable data recovery if the Discovery engine, ncp_disco, fails. A discovery running in this mode is slower than a standard discovery, because of the extra time required to store data on the disk throughout the discovery process.

An administrator can hide this option from the **Network Discovery Configuration** page by setting the `discoconfig.allow.table.caching` property to `false` in the `discoconfig.properties`. If the option is not present in the GUI, an administrator can still control whether discovery tables are cached by setting the `disco.config.m_WriteTablesToCache` property to `true` in the `DiscoConfig.cfg` file, or the domain-specific version of that file.

Enable File Finder Verification

Enable this setting to use the Ping finder to verify the existence of devices specified in the files used by the File finder. If you enable this setting, ensure that the Ping finder is enabled. Enable this setting if you are not sure that the devices are still connected to the network. For example, you might want to enable this setting if your network is rapidly changing.

Enable Rediscovery Rebuild Layers

Enable this setting to rebuild the topology layers following a partial rediscovery. If you specify that topology layers are rebuilt following partial rediscovery, the result is an accurate topology showing all connectivity. However, the process of adding new devices takes longer.

Tip: To configure a partial rediscovery to run as quickly as possible, disable this option.

Enable Rediscovery of Related Devices

By default the remote neighbors of a device are not rediscovered, even if the rediscovery of that device indicates that the remote neighbors have changed. The remote neighbors can be rediscovered on the next full rediscovery. Enable this setting if you want to change this default behavior and rediscover any changed remote neighbors when rediscovering that device. If new neighboring devices are discovered during the blackout phase, another partial rediscovery is triggered after the blackout phase. If your network connectivity changes often, for example if you are rediscovering devices with switch forwarding database tables, enabling rediscovery of related devices can cause several discoveries to run one after another. Multiple consecutive discoveries are more likely to happen if the partial discovery takes longer, for example, if the **Enable Rediscovery Rebuild Layers** setting is enabled.

Tip: To configure a partial rediscovery to run as quickly as possible, disable this option.

Enable ifName/ifDescr Interface Naming

Changes the default naming convention for discovered interfaces. Names interfaces using data from the SNMP interfaces table ifName and ifDescr fields as appropriate. For example, `Fa0/0, Gi 1.0.2:0, Gigabit Ethernet 4/1`. If you change the default naming convention for discovered interfaces, you must change the BuildInterfaceName stitcher to specify your naming convention.

Tip: Some devices might report interface names and descriptions that are too long to display properly in the topology display. If there are devices that report long or incorrect interface names and descriptions, disable this setting.

Enable Inference of PEs using BGP data on CEs

Discovers intervening provider networks as a "third-party" object on multiple networks that run across a provider network. Examples of this type of network include enterprise VPNs across a provider MPLS core network. Select this option if you want to link all your networks in a single topology and perform root cause analysis (RCA) across your networks.

This option infers the existence of inaccessible provider-edge (PE) devices by using the BGP data on the customer-edge (CE) devices that point to the PE devices. In order to discover this BGP data, the

BGP discovery agents must be enabled. If you want to use the cross-domain discovery function, clear this option. If this option is selected, errors are generated during the cross-domain discovery.

You can also optionally specify which of the inferred PE devices are valid devices, by populating the scope.inferMPLSPEs table, using standard format scope entries, as in the scope.zones table. If populated this table allows you to define which IP addresses you see on CE devices that you consider valid PE devices. Use this option when you have inaccessible devices that are connected by BGP but which are not actually PE devices.

Enable Inference of MPLS CE routers on /30 subnets

Generates Service-Affected Events on customer VPNs. Select this option if you are a service provider without access to customer CE routers.

You do not normally need to enable inference of both CE and PE devices.

Starting a discovery

After you configure a discovery, you can start and, if necessary, stop the discovery.

Before you begin

Make any required discovery configuration changes before you launch the discovery.

About this task

You can start the following types of discovery:

Discovery
Run a full discovery to discover your network for the first time, or to refresh the network topology if you know the network has changed.

Partial discovery
Run a partial discovery if you know that the changes to your network are limited to a small number of devices. You need to configure scoping and seeding as part of starting each partial discovery. If the relationship of the devices that are in scope with their neighboring devices has changed, then the neighboring devices may also be discovered. If the partial discovery needs to discover a large amount of devices based on connectivity information, then a full discovery is started.

Note: If you stop a running discovery, you must then do a full discovery before you are able to do a partial discovery.

To start a discovery, complete the following steps.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Status**.
2. Select the domain in which you want to run a discovery from the **Domain** menu. You can start to type the name of the domain, and matching domains are listed below the **Domain** field.
3. Start a full or partial discovery:

   - To start a full discovery, click **Start Discovery**  only. The discovery starts.
   - To start a partial discovery, click the downward-facing arrow next to the **Start Discovery** button  and select **Start Partial Discovery** from the menu (if a full discovery has not been run since the last time that the discovery engine, **ncp_disco**, was started, the option to start a partial discovery is grayed out). The **Partial Discovery** window is displayed. Specify the IP addresses and subnets that contain the devices to be discovered:

   a) Under **Partial Discovery**, select the required nodes and subnets.

   b) To add a new subnet or node, click **New.**

c) Complete the fields as follows and click **OK**:

Discover
Select one of the following options:

Identifier
Type the required IP address.

Subnet
Type the required subnet and specify the number of netmask bits. The **Netmask** field is automatically updated.

EMS Device Name
Type the name of a device that was discovered through an Element Management System (EMS). You can type the nativeID, entity name, sysName, or display name.

Note: When running a partial discovery of an EMS collector from the **Discovery Status GUI**, within the **New Partial Discovery Node/Subnet** window you must specify the EMS identifier value in the **EMS Device Name** field and not in the **Identifier** field. The **Identifier** field only accepts IP addresses.

d) To add new scope zones, click **Scope**.

e) To add a new discovery scope zone click **New**  . To edit an existing scope zone, click the required entry in the list.

f) Complete the fields as follows and click **OK**:

Action
Define the subnet range as an inclusion zone or exclusion zone. If the subnet range is an inclusion zone that you intend to ping during the discovery, click **Add to Ping Seed List**. Clicking this option automatically adds the devices in the scope zone as a discovery seed devices.

Restriction: The `Add to Ping Seed List` option is not available for IPv6 scope zones. This prevents ping sweeping of IPv6 subnets, which can potentially contain billions of devices to be pinged. Ping sweeping of IPv6 subnets can therefore result in a non-terminating discovery.

g) Click **OK** then click **Go**.

When a full or partial discovery is running, the **Start Discovery** button is toggled off  .

4. To stop a discovery, click **Stop Discovery**  . The discovery might take a short time to stop, during which time both the **Start Discovery** and **Stop Discovery** buttons are toggled off. If you stop a discovery, you cannot then do a partial discovery until after the next full discovery.

Note: When you stop a discovery, the discovery cache is lost. This is why you must wait for the completion of the next full discovery before being able to perform a partial discovery. It is possible to configure the Discovery engine to save the discovery cache as the discovery is running, which would enable you to run a partial discovery immediately following the manual stop of a discovery. You can configure the Discovery engine to save the discovery cache by clicking **Enabling Caching of Discovery Tables** in the **Advanced** tab.

Results

While the discovery is running, you can monitor the progress of the discovery.

Note: You cannot stop the discovery from the GUI in the correlating connectivity phase. Stopping the discovery process from the command line while the topology is being created can corrupt the discovery.

What to do next

After the discovery is complete, the **Start Discovery** button is toggled on, and you can run another full or partial discovery at any time. If the Event Gateway Disco plug-in is enabled, then a new discovery can be triggered automatically when a reboot event (event ID of `NmosSnmpReboot` triggered by the rebootDetection poll policy) is received.

Schemas and tables for GUI discovery parameters

Use this reference information to learn to which schemas and table the settings made on the tabs of the **Network Discovery Configuration** page are saved.

The following table describes the tables to which the settings made on each tab of the **Network Discovery Configuration**page are saved. In these tables, *DOMAIN_NAME* represents the name of the network domains in your deployment, for example NCOMS.

Table 39. Schemas and tables to which the discovery parameters are mapped

Network Discovery Configuration tab	Description	Schema or table name
Scope	The zones of the network (that is, subnet ranges) that you want to include in the discovery, and the zones that you want to exclude.	`DiscoScope.`*`DOMAIN_NAME`*`.cfg`
Seed	The location from which to begin discovering devices. This might be one or more IP addresses, or subnet addresses. To seed the discovery, the following *finders* are used: Ping finder and File finder.	Ping finder: `DiscoPingFinderSeeds.`*`DOMAIN_NAME`*`.cfg` File finder: `DiscoFileFinderParseRules.`*`DOMAIN_NAME`*`.cfg`
Full Discovery Agents and **Partial Rediscovery Agents**	The discovery agents to be used to investigate device connectivity. Default agents are provided for the type of discovery you want to perform, for example a layer 2 or layer 3 discovery. You can select different set of agents for full discoveries and for partial discoveries. The agents vary because connectivity information varies with the technology of the hardware in the network.	`DiscoAgents.`*`DOMAIN_NAME`*`.cfg`
Device Access	SNMP community strings and Telnet parameters that Network Manager uses to interrogate devices that use SNMP and Telnet.	SNMP community strings: `SnmpStackSecurityInfo.cfg` Telnet access: `TelnetStackPasswords.cfg`
Filters	Use filters to filter out devices either before discovery or after discovery. You can filter out devices based on a variety of criteria, including location, technology, and manufacturer. Prediscovery filters prevent discovered devices from being polled for connectivity information. Post-discovery filters prevent discovered devices from being passed to MODEL.	`DiscoSchema.`*`DOMAIN_NAME`*`.cfg`
DNS	Access to DNS services that are used to perform domain name lookups.	`DiscoDNSHelperSchema.cfg`
NAT	The data that provides the discovery mappings between address space data and real device IP addresses to facilitate further discovery.	`DiscoSchema.`*`DOMAIN_NAME`*`.cfg`

Table 39. Schemas and tables to which the discovery parameters are mapped (continued)		
Network Discovery Configuration tab	**Description**	**Schema or table name**
Multicast	Multicast groups and sources used by the Discovery engine to configure multicast scopes.	`DiscoScope.`*`DOMAIN_NAME`*`.cfg`
Advanced	Advanced settings control features of the discovery such as concurrent processes and time-outs. Use these parameters to increase the speed of the discovery, but balance the speed with the load on the server. Generally, a faster discovery results in more memory usage on the server.	`DiscoSchema.`*`DOMAIN_NAME`*`.cfg`

Discovering the network using the command-line interface

As an experienced user, you can configure and track a network discovery using configuration files and database queries.

Discovery configuration files

Experienced users can configure the discovery by editing the discovery configuration files.

To configure the discovery using the command-line interface (command line), edit the discovery configuration files and create or edit inserts into the databases of the discovery processes.

Note: The DiscoSchema.cfg configuration file contains the schemas for all the discovery databases. Unlike the files listed below, the DiscoSchema.cfg configuration file contains no insert statements. You can view this file but it must not be edited.

When ncp_disco is running, it periodically scans the agents and stitchers directories and loads any new or modified stitcher and discovery agent definitions.

Table 40 on page 205 shows which configuration files to edit to configure the discovery, and whether the configuration can also be done using the **Discovery Configuration** GUI.

Table 40. User-editable discovery configuration files		
Discovery configuration taska	**Configuration file**	**GUI tab**
Scoping discovery		
Defining inclusion and exclusion zones	DiscoScope.cfg	**Scope**
Ignoring discovery scope	DiscoScope.cfg	**Scope**
Seeding discovery		
Using and configuring the Ping finder	DiscoPingFinderSeeds.cfg	**Seed**
Running multiple instances of a finder		
Configuring broadcast and multicast address pinging	DiscoPingFinderSeeds.cfg	**Advanced**

Table 40. User-editable discovery configuration files (continued)

Discovery configuration taska	Configuration file	GUI tab
Using and configuring the File finder	DiscoFileFinderParseRules.cfg	**Seed**
Enabling File finder device verification	DiscoConfig.cfg	**Advanced**
Enabling Ping verification	DiscoConfig.cfg	
Using and configuring the Database finder	DiscoDBEntryFinderQueries.cfg	
Using and configuring the Collector finder	DiscoCollectorFinderSeeds.cfg	
SNMP		
Configuring SNMP community strings and passwords	SnmpStackSecurityInfo.cfg	**Passwords**
Configuring the SNMP helper	DiscoSnmpHelperSchema.cfg	**Advanced**
Overriding SNMP helper settings for specific devices and subnets		
Telnet		
Configuring Telnet access to network devices	TelnetStackPasswords.cfg	**Passwords**
Configuring the Telnet helper	DiscoTelnetHelperSchema.cfg	**Advanced**
Configuring a context-sensitive discovery	DiscoConfig.cfg	
Agents		
Enabling and disabling discovery agents	DiscoAgents.cfg	**Full Discovery Agents Partial Rediscovery Agents**
Filtering devices sent to the agents	Discovery agent definition files	**Filters**
Filtering topology data returned by an agent	Discovery agent definition files	
Filtering topology data returned by all agents	DiscoAgentReturns.filter	
Changing the number of threads used by an agent	DiscoAgents.cfg	
Enabling multi-threaded operation for Perl agents	Discovery agent definition files	
Enabling and disabling partial matching	IpForwardingTable.agnt agent definition file (for modern devices that use RFC2096) IpRoutingTable.agnt agent definition file (for older devices that use RFC1213).	

Table 40. User-editable discovery configuration files (continued)		
Discovery configuration taska	**Configuration file**	**GUI tab**
Restricting discovery		
Restricting device detection	DiscoScope.cfg DiscoPingFinderSeeds.cfg	**Scope** **Seed**
Restricting device interrogation	DiscoScope.cfg	
Restricting device instantiation		
SNMP interface filtering	DiscoSnmpHelperFilters.cfg	
Configuring the DNS helper services	DiscoDNSHelperSchema.cfg	**DNS**
Configuring a NAT discovery	NATTextFileAgent agent NATGateway agent	**NAT**
Setting advanced configuration		
Advanced File Finder Configuration Advanced Ping Finder Configuration Advanced DNS Helper Configuration Advanced SNMP Helper Configuration Advanced Telnet Helper Configuration	DiscoFileFinderParseRules.cfg DiscoPingFinderSeeds.cfg DiscoDNSHelperSchema.cfg DiscoSnmpHelperSchema.cfg DiscoTelnetHelperSchema.cfg **Note:** As an experienced user, you can set more advanced configuration parameters in the configuration files than are available in the Advanced tab of the GUI.	**Advanced**

Discovery agent definition files

The discovery agent definition files define the operation of the discovery agents.

Filtering devices using the definition files

Note: Network Manager kills all discovery agents at the end of data collection stage 3. This ensures that the next discovery restarts the agents and forces the agents to reread their configuration files at the beginning of a discovery, thereby detecting any changes to the configuration files.

You can apply a filter to a discovery agent by editing the supported devices filter within the DiscoAgentSupportedDevices( ); section of the discovery agent definition file ($NCHOME/precision/disco/agents/*.agnt). All discovery agents have a definition file in this directory, regardless of whether the agent is text-based or precompiled.

The supported devices filter is a filter against the attributes of the agentTemplate.despatch table.

The DiscoAgentSupportedDevices( ); section accepts full OQL comparison tests using comparison operators such as `like`, < , > , = , and <>. Detailed information about comparison tests in OQL can be found in the *IBM Tivoli Network Manager Reference*.

Tip: Altering agent definition files can introduce parse errors. To check your agent for parse errors, run the agent in debug mode and examine the debug output.

Example: filtering devices sent to the CDP agent

The following example shows the DiscoAgentSupportedDevices(); section of the `CDP.agnt` agent definition file. Only network entities that match the specified Object IDs are processed by the CDP agent,

that is, only devices that use the Cisco Discovery Protocol. The CDP agent does not process devices with the Object ID 1.3.6.1.4.1.9.1.226.

```
DiscoAgentSupportedDevices
(
            " (
                        ( m_ObjectId like '1\.3\.6\.1\.4\.1\.9\..*' )
                        AND
                        ( m_ObjectId <> '1.3.6.1.4.1.9.1.226' )
            ) "
);
```

Example: using wildcards in device filters

The following example shows the use of wild cards in the IP address column. The agent only accepts devices with an IP address beginning 10.10.2.

```
DiscoAgentSupportedDevices
(
            " ( m_UniqueAddress like '10\.10\.2\..*' ) "
);
```

Example: using multiple device filter conditions

The following example shows the combination of multiple filter conditions. The agent accepts only devices that have the Object ID 1.3.6.1.4.1.9.5.7.. have an IP address starting with 10.10.. and do not have the name clandestine.

```
DiscoAgentSupportedDevices
(
            "(
                    ( m_ObjectId = '1.3.6.1.4.1.9.5.7' )
                    AND
                    ( m_UniqueAddress like '^10\.10\..*' )
                    AND
                    ( m_Name not like '.*[cC]landestin[eE].*' )
            )"
);
```

Enabling multi-threaded operation for Perl discovery agents

The number of threads used by discovery agents is set in the `DiscoAgents.cfg` configuration file. Perl agents must have multi-threaded operation enabled before the setting in the `DiscoAgents.cfg` configuration file has any effect.

To enable multi-threaded operation for a Perl discovery agent, add the following line to its definition file:

```
DiscoAgentDefaultThreads( 10 );
```

The insert above specifies that the agent uses 10 threads by default. If you set a different number of threads in the `DiscoAgents.cfg` configuration file, that value overrides the value in the agent definition file.

Restriction: Many of the add-on CPAN modules often used with Perl are not thread safe. Perl discovery agents using such modules might need to be restricted to a single thread.

Filtering topology data returned by a discovery agent

To filter topology data returned by a single agent, define a filter within the relevant agent (.agnt) file.

Example: filtering out subscriber cable-modem interfaces

The CMTS.agnt agent file retrieves data from cable modems connected to a cable modem terminating services device. This example describes a filter added to the CMTS.agnt file which filters out subscriber

cable modem interfaces from topology data returned for the CMTS devices. The example filter is as follows:

```
DiscoAgentReturnsFilterList
{
      DiscoReturnsFilter
      {
            "(
             m_LocalNbr->m_IfType = 229
            )"
      }
};
```

Sample: defining multiple topology filters

The following example illustrates how to define multiple topology data filters within an agent. The first filter specifies that each time a record is returned where the interface ifIndex value is 4, then the m_Name, m_HaveAccess, m_LocalNbr->m_SubnetMask, and m_RemoteNbr->m_RemoteNbrPhysAddr fields must be deleted from the record. The second filter deletes records returned when the interface ifIndex value is 5.

```
DiscoAgentReturnsFilterList
{
      DiscoReturnsFilter
      {
            "(
             m_LocalNbr->m_IfIndex = 4
            )"
            DiscoDeleteFields {
                  "m_Name",
                  "m_HaveAccess",
                  "m_LocalNbr->m_SubnetMask",
                  "m_RemoteNbr->m_RemoteNbrPhysAddr",
            }
      }
      DiscoReturnsFilter
      {
            "(
             m_LocalNbr->m_IfIndex = 5
            )"
      }
};
```

Example: Disabling partial matching

The following example could be appended to the IpForwardingTable.agnt definition file to ensure that if a router with m_ObjectId='1.3.6.1.4.1.9.1.48' is discovered (that is, a Cisco 7505 router), partial matching is attempted only when the router is running IOS version 12.2 or higher.

```
            DiscoRouterPartialMatchRestrictions
            (
                        "(m_ObjectId='1.3.6.1.4.1.9.1.48', m_OSVersion>='12.2',
                        m_MibVar='sysDescr')"
            );
```

Example: Disabling partial matching using wildcards

The following example ensures that partial matching is not used on Cisco 2600 routers, Cisco 7505 routers running an IOS revision lower than 12.2, and Redstone routers.

```
            DiscoRouterPartialMatchRestrictions
            (
                        "(m_ObjectId='1.3.6.1.4.1.9.1.209'),
                        (m_ObjectId='1.3.6.1.4.1.9.1.48', m_OSVersion>='12.2',
                         m_MibVar='sysDescr'),
                        (m_ObjectId like '1\.3\.6\.1\.4\.1\.2773\..*')"
            );
```

DiscoAgents.cfg configuration file

The DiscoAgents.cfg configuration file defines which agents run during a discovery.

Database table used

The DiscoAgents.cfg configuration file can be used to configure inserts into the disco.agents database table.

Sample: enabling the IpRoutingTable discovery agent

The following example activates the IpRoutingTable discovery agent.

```
insert into disco.agents
(
        m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence
    )
values
(
        'IpRoutingTable', 1, 0, 0, 2
    );
```

Sample: enabling the Details and Associated Address agents

The following example OQL inserts activate the Details and Associated Address agents.

```
insert into disco.agents
(
        m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence
    )
values
(
        'Details', 1, 0, 0, 1
    );

insert into disco.agents
(
        m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence
    )
values
(
        'AssocAddress', 1, 0, 0, 2
    );
```

Sample: enabling the ARP cache agent

The ARP Cache agent assists in MAC address-to-IP address resolution during the discovery. You must enable this agent to run during a layer 2 discovery. The following example shows how to ensure that the ARP Cache agent runs during a discovery.

```
insert into disco.agents
(
        m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence
)
values
(
        'ArpCache', 1, 0, 0, 2
);
```

Sample: deactivating the StandardSwitch and SuperStack3ComSwitch agents

The following example deactivates the StandardSwitch and the SuperStack3ComSwitch discovery agents.

```
insert into disco.agents
(
        m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence
    )
values
(
        'StandardSwitch', 0, 1, 1, 3
```

```
    );

insert into disco.agents
(
        m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence
    )
values
(
        'SuperStack3ComSwitch', 0, 1, 1, 3
    );
```

Sample: changing the number of threads used by the IpRoutingTable discovery agent

The following example sets the number of threads used by the IpRoutingTable discovery agent to 50. Increasing the number of threads used by an agent allows the agent to process more devices at once, and can speed up discovery. However, increasing the number of threads used by an agent also uses more memory.

```
insert into disco.agents
(
        m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence, m_NumThreads
    )
values
(
        'IpRoutingTable', 1, 0, 0, 2, 50
    );
```

Sample: changing the number of threads used by the NMAPScan Perl discovery agent

The following example sets the number of threads used by the NMAPScan Perl discovery agent to 50. To define the number of threads used by a Perl discovery agent, you must first enable multiple threads for that agent in the discovery agent definition file.

```
insert into disco.agents
(
        m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence, m_NumThreads
    )
values
(
        'NMAPScan', 1, 0, 0, 2, 50
    );
```

DiscoAgentReturns.filter configuration file

The DiscoAgentReturns.filter configuration file allows you to apply a topology data filter to data returned by all discovery agents.

Filtering topology data returned by all agents

The $NCHOME/precision/disco/agents/DiscoAgentReturns.filter configuration file filters the same topology data from all of the agent returns tables. The syntax used in this file is the same as the syntax used in topology filters in the discovery agent definition files.

Sample: filtering out subscriber cable-modem interfaces

The following example filters out subscriber cable modem interfaces from topology data:

```
DiscoAgentReturnsFilterList
{
      DiscoReturnsFilter
      {
            "(
             m_LocalNbr->m_IfType = 229
            )"
      }
};
```

DiscoARPHelperSchema.cfg configuration file

The DiscoARPHelperSchema.cfg configuration file performs IP address to MAC address resolution.

Database used

The DiscoARPHelperSchema.cfg configuration file defines inserts into the ARPHelper.configuration database table.

Sample: Configuring the ARP helper

The following example insert configures the ARP helper to use one thread.

```
insert into ARPHelper.configuration
(
        m_NumThreads
)
values
(
        1
);
```

DiscoCollectorFinderSeeds.cfg configuration file

The `DiscoCollectorFinderSeeds.cfg` configuration file defines how topology data is acquired from Element Management System (EMS) collectors during discovery.

Database used

The `DiscoCollectorFinderSeeds.cfg` configuration file defines inserts into the collectorFinder database.

Note that there is another file associated with the collectorFinder database, the DiscoCollectorFinderSchema.cfg file, but you should not need to alter this file.

Sample: configuring a single collector

The following example seeds a single collector running on the local server. The example does not specify values for other fields, such as m_DataSourceId and m_NumRetries, and they automatically take the default values from the configuration table.

```
insert into collectorFinder.collectorRules
        ( m_Port)
values
        ( 8082 );
```

DiscoDBEntryFinderQueries.cfg configuration file

The DiscoDBEntryFinderQueries.cfg file is used to specify the a database query to be run against a specified database in order to retrieve a list of IP addresses of devices to discover on the network.

Database tables used

This configuration file can be used to configure inserts into the following database tables:

- dbEntryFinder.configuration
- dbEntryFinder.dbQueries

Example: configuring the Database finder to use five threads

The following example insert configures the Database finder to use five threads.

```
insert into dbEntryFinder.configuration
   ( m_NumThreads )
```

```
values
   ( 5 );
```

Example: configuring the Database finder to use an external Tivoli Data Warehouse database

The following example configuration instructs the Database finder to retrieve device data from an external Tivoli Data Warehouse database.

```
insert into dbEntryFinder.dbQueries
(
            m_DbId, m_TriggerType, m_Query, m_Parameters, m_Mapping
)
values
(
            "TDW",
            1,
            "select DISTINCT MAC_Address , System_Name ,
                Network_Interface_Name , Interface_Status ,
                Device_Type , Interface_IP_Address
                from ABC_Network where Linux_OS_Config = ?",
            [ 'Redhat 6.5'  ],
            [
               {
                   FromDb = "eval(text,'&System_Name')",
                   ToFinder = 'm_Name'
               },
               {
                   FromDb = "eval(text,'&Interface_IP_Address')",
                   ToFinder = 'm_UniqueAddress'
               },
               {
                   FromDb = 23,
                   ToFinder = 'm_ExtraInfo->m_SourceId'
               },
               {
                   FromDb = "eval(text,'&Device_Type')",
                   ToFinder = 'm_ExtraInfo->m_DeviceType'
               }
            ]
);
```

The foregoing insert specifies that:

- The database that hosts the device details has a database identifier (dbId) of `TDW` in the `DbLogins.`*`DOMAIN`*`.cfg` configuration file.
- The trigger type is `1`. This means that this query will be invoked during a full discovery.
- The query to run against the `TDW` database must retrieve the following data for each device:
  - MAC address
  - Device name
  - Network interface name
  - Interface status
  - Device type
  - IP address
- Using the optional parameter field, specify that the Linux operating system is RedHat 6.5.
- Map device data retrieved by the query to relevant fields in the `finders.returns` table.

DiscoDNSHelperSchema.cfg configuration file

The DiscoDNSHelperSchema.cfg configuration file defines access to DNS, which enables the discovery to do domain name lookups, by configuring the DNS helper.

Database tables used

The DiscoDNSHelperSchema.cfg configuration file can be used to configure inserts into the following database tables:

- DNSHelper.configuration
- DNShelper.methods

Sample: configuring the DNS helper

The following example inserts configure the DNS helper using the information in the DNSHelper.configuration database table and the DNShelper.methods database table. The example shows inserts into the DNShelper.methods database table corresponding to the following method types:

- 0 - System
- 1 - DNS using `m_NameDomain` to specify a domain suffix to append to all discovered device names.
- 1 - DNS using `m_NameDomainList` to specify a list of expected domain suffixes.
- 2 - File

```
insert into DNSHelper.configuration
(
     m_NumThreads, m_MethodList, m_TimeOut
)
values
(
     1, ['HostsFile'] , 5
);

insert into DNSHelper.methods
(
     m_MethodName, m_MethodType
)
values
(
     "HostService", 0
);

insert into DNSHelper.methods
(
     m_MethodName, m_MethodType, m_NameServerAddr, m_TimeOut, m_NameDomain
)
values
(
     "abcIPv6DNS", 1, "2222:15f8:106:203:250:4ff:fee8:6d75", 3,
"tivlab.raleigh.ibm.com"
);

insert into DNSHelper.methods
(
     m_MethodName, m_MethodType, m_TimeOut, m_NameServerAddr, m_NameDomainList
)
values
(
     "defIPv6DNS", 1, 3, "2222:15f8:106:203:250:4ff:fee8:6d75",
     ['uk.eu.org',
     'fra.eu.org',
     'de.eu.org',
     'it.eu.org',
     'sp.eu.org']
);

insert into DNSHelper.methods
(
     m_MethodName, m_MethodType, m_FileName, m_FileOrder
)
values
(
     'HostsFile', 2, 'etc/hosts', 1
);
```

DiscoFileFinderParseRules.cfg configuration file

The DiscoFileFinderParseRules.cfg file can be used to specify the files to be parsed for a list of IP addresses of devices that exist on the network.

Database tables used

This configuration file can be used to configure inserts into the following database tables:

- fileFinder.parseRules
- fileFinder.configuration

Note that there is another configuration file associated with the fileFinder database, the DiscoFileFinderSchema.cfg file, but you should not need to alter this file.

Sample: configuring the File finder to use five threads

The following example insert configures the File finder to use five threads.

```
insert into fileFinder.configuration
   ( m_NumThreads )
values
   ( 5 );
```

Example: configuring the File finder to parse /var/tmp/logged_hosts

The following example configuration instructs the File finder to parse an example text file, logged_hosts, that has been saved in the /var/tmp directory. The contents of the example file are shown below.

```
vi /var/tmp/logged_hosts

172.16.1.21   dharma              04:02:08
172.16.1.201  phoenix             19:07:08
172.16.1.25   lnd-sun-tivoli   15:10:00
172.16.2.33   ranger              19:07:07
~
"/var/tmp/logged_hosts" [Read only] 4 lines, 190 characters
```

The three columns in this example file respectively contain an IP address, the device name, and a time value. The columns are separated by white space, which can be multiple tabs, spaces, or a combination of both. You could configure the File finder to parse this example text file using an insert similar to the example.

```
insert into fileFinder.parseRules
(
            m_FileName, m_Delimiter, m_ColDefs
)
values
(
        "/var/tmp/logged_hosts",
        "[     ]+",
        [
            {
                m_VarName="m_UniqueAddress",
                m_ColNum=1
            },
            {
                m_VarName="m_Name",
                m_ColNum=2
            }
        ]
);
```

The above insert specifies that:

- The full path and name of the file is /var/tmp/logged_hosts.

- The source-file delimiter is white space. The column delimiter is indicated in the insert using a simple regular expression, [*tab space*]+ . You must press the **tab** and **space** keys rather than typing \t to represent the tab character.
- The first column contains IP addresses and must be mapped to the m_UniqueAddress column of the finders.returns table.
- The second column contains host names and must be mapped to the m_Name column of the finders.returns table.

Because the third column in the example text file is not relevant, it has not been mapped to a column of finders.returns and is ignored by the File finder during the discovery.

Example: configuring the File finder to parse the /etc/hosts file

The following insert instructs the File finder to:

- Parse /etc/hosts.
- Treat white space as the data separator.
- Use the following column definitions:
  - m_UniqueAddress for the first column
  - m_Name for the second column

```
insert into fileFinder.parseRules
(
            m_FileName,
            m_Delimiter,
            m_ColDefs
)
values
(
            "/etc/hosts",
            "[     ]",
            [
                {
                    m_VarName="m_UniqueAddress",
                    m_ColNum=1
                },
                {
                    m_VarName="m_Name",
                    m_ColNum=2
                }
            ]
);
```

Sample: configuring the File finder to parse /etc/defaultrouter

The following insert instructs the File finder to:

- Parse /etc/defaultrouter.
- Treat one or more occurrences of white space as the data separator.
- Use m_UniqueAddress as the column definition.

```
insert into fileFinder.parseRules
(
            m_FileName,
            m_Delimiter,
            m_ColDefs
)
values
(
            "/etc/defaultrouter",
            "[     ]+",
            [
                {
                    m_VarName="m_UniqueAddress",
                    m_ColNum=1
                }
```

```
            ]
);
```

DiscoHelperServerSchema.cfg configuration file

The `DiscoHelperServerSchema.cfg` configuration file defines the contents of the several helper databases.

Database tables used

This configuration file can be used to configure inserts into the following database tables.

ARP helper database tables:

- ARPHelper.ARPHelperTable
- ARPHelper.ARPHelperConfig

DNS helper database tables:

- DNSHelper.DNSHelperTable
- DNSHelper.DNSHelperConfig

Ping helper database tables:

- PingHelper.PingHelperTable
- PingHelper.PingHelperConfig

SNMP helper database tables:

- SnmpHelper.SnmpHelperTable
- SnmpHelper.SnmpHelperConfig

Telnet helper database tables:

- TelnetHelper.TelnetHelperTable
- TelnetHelper.TelnetHelperConfig

XMLRPC helper database tables:

- XmlRpcHelper.XmlRpcHelperTable
- XmlRpcHelper.XmlRpcHelperConfig

DiscoPingFinderSeeds.cfg configuration file

The DiscoPingFinderSeeds.cfg configuration file is used for seeding the Ping finder and restricting device detection.

Database tables used

The DiscoPingFinderSeeds.cfg configuration file can be used to configure inserts into the following database tables:

- pingFinder.pingRules
- pingFinder.scope

Note that there is another configuration file associated with the pingFinder database, the DiscoPingFinderSchema.cfg file, but you should not need to alter this file.

Note: If you are seeding an IPv6 discovery, bear in mind that there are potentially billions of devices to be pinged within a single IPv6 subnet. To ensure that discovery completes, you must specify a sufficiently large netmask if you specify an IPv6 subnet as a ping seed.

Sample: seeding the Ping finder with a single device address

The following example insert defines a single seed with IP address of 10.10.2.224. This example does not specify values for m_NumRetries and m_TimeOut because they automatically take the default values from the configuration table.

Restriction: Network Manager does not support the IPv4–mapped IPv6 format and expects all IPv6 addresses to be in standard colon-separated IPv6 format. For example, Network Manager does not support an IPv4–mapped IPv6 address such as `::ffff:192.0.2.128`. Instead enter this address as `::ffff:c000:280` (standard colon-separated IPv6 format).

```
insert into pingFinder.pingRules
        ( m_Address, m_RequestType )
values
        ( "10.10.2.224", 1 );
```

Sample: seeding the Ping finder with a class B subnet address

The following example insert defines a single class B subnet as a seed.

```
insert into pingFinder.pingRules
        ( m_Address, m_RequestType, m_NetMask )
values
        ( "10.10.0.0", 2, "255.255.0.0" );
```

Sample: seeding the Ping finder with class C subnet addresses

The following example insert defines two Class 2 subnets as seeds.

```
insert into pingFinder.pingRules
        ( m_Address, m_RequestType, m_NetMask )
values
        ( "10.10.2.0", 2, "255.255.255.0" );

insert into pingFinder.pingRules
        ( m_Address, m_RequestType, m_NetMask )
values
        ( "10.10.47.0", 2, "255.255.255.0" );
```

Sample: restricting device detection

The following example insert configures the Ping finder to use the scope.zones table and use the discovery scope.

```
insert into pingFinder.scope
        ( m_UseScope, m_UsePingEntries )
values
        ( 1, 1 );
```

Important: Other combinations of m_UseScope and m_UsePingEntries filters are not recommended. Specifying values (0,0) results in an unbounded discovery, while specifying values (0,1) results in devices that you do not want to discover being needlessly pinged.

DiscoPingHelperSchema.cfg configuration file

The DiscoPingHelperSchema.cfg configuration file defines how devices are to be pinged.

Database table used

The DiscoPingHelperSchema.cfg configuration file can be used to configure inserts into the pingHelper.configuration database table.

In this example configuration of the DiscoPingHelperSchema.cfg configuration file, the parameters specify to:

- Use 20 threads of process execution.

- Wait a maximum of 250 ms for a reply from a device.
- Retry unresponsive devices a maximum of five times.
- Wait 50 ms between pinging devices in a subnet.
- Not use broadcast or multicast pinging.

```
insert into pingHelper.configuration
(
        m_NumThreads,
        m_TimeOut,
        m_NumRetries,
        m_InterPingTime,
        m_Broadcast,
        m_Multicast
)
values
(
        20, 250, 5, 50, 0, 0
);
```

DiscoConfig.cfg configuration file

The DiscoConfig.cfg configuration file is used to have the Ping finder automatically check the devices discovered by the File finder, and to enable a context-sensitive discovery.

Database table used

The DiscoConfig.cfg configuration file can be used to configure inserts into the following tables:

- disco.config
- disco.managedProcesses
- disco.NATStatus
- translations.NATAddressSpaceIds
- disco.ipCustomTags
- disco.filterCustomTags
- translations.collectorInfo
- failover.restartPhaseAction
- failover.config
- failover.doNotCache

The following examples illustrate inserts into the disco.config database table.

Sample: pinging File finder devices

The following example command configures the discovery so that the devices discovered by the File finder are automatically checked by the Ping finder.

```
update disco.config set m_CheckFileFinderReturns = 1;
```

Sample: enabling a context-sensitive discovery

Attention: Enabling a context-sensitive discovery automatically enables all the Context agents. Disabling a context-sensitive discovery automatically disables all the Context agents. You should not manually enable or disable Context agents, either through the configuration files or through the Discovery Configuration GUI.

To enable a context-sensitive discovery, append the following insert to the `DiscoConfig.cfg` file:

```
insert into disco.config
(
      m_UseContext
)
```

```
values
(
      1
)
```

Inserting the value `0` disables the context-sensitive discovery.

Enriching topology using custom tags

You can use the disco.ipCustomTags and disco.filterCustomTags tables to enrich the discovered topology by associating one or more name-value pair tags with discovered entities.

DiscoScope.cfg configuration file

The DiscoScope.cfg configuration file can be used to configure the scope of a discovery.

Database tables used

This configuration file can be used to configure inserts into the following database tables:

- scope.zones
- scope.detectionFilter
- scope.instantiateFilter
- scope.multicastGroup
- scope.multicastSource
- scope.special

Restriction: The scope.zones table can only be used to configure scoping for IP-based entities. Non-IP entities such as layer 1 optical devices must be scoped using the scope.detectionFilter table.

Sample: defining an inclusion zone

The following example insert defines the 10.10.2.* subnet as an inclusion zone.

Restriction: Network Manager does not support the IPv4–mapped IPv6 format and expects all IPv6 addresses to be in standard colon-separated IPv6 format. For example, Network Manager does not support an IPv4–mapped IPv6 address such as ::ffff:192.0.2.128. Instead enter this address as ::ffff:c000:280 (standard colon-separated IPv6 format).

```
insert into scope.zones
(
        m_Protocol,
        m_Action,
        m_Zones
)
values
(
        1,
        1,
        [
            {
                m_Subnet="10.10.2.*"
            }
        ]
);
```

Sample: defining multiple inclusion zones

The following example defines three different IP inclusion zones each using a different syntax to define the subnet mask. The following devices are discovered:

- Any device within the 172.16.1.0 subnet (with a subnet mask of 24, that is, 24 bits turned on and 8 bits turned off, which implies a netmask of 255.255.255.0).
- Any device within the 172.16.2.0 subnet with a mask of 255.255.255.0.

- Any device within the 172.16.3.0 subnet with a mask of 255.255.255.0.

```
insert into scope.zones
(
    m_Protocol,
    m_Action,
    m_Zones
)
values
(
        1,
        1,
        [
              {
                    m_Subnet="172.16.1.0",
                    m_NetMask=24
              },
              {
                    m_Subnet="172.16.2.*"
              },
              {
                    m_Subnet="172.16.3.0",
                    m_NetMask="255.255.255.0"
              }
        ]
);
```

Sample: defining an exclusion zone

The following example insert defines a single exclusion zone for the IP protocol, and associates the zone with a subnet.

```
insert into scope.zones
(
        m_Protocol,
        m_Action,
        m_Zones
)
values
(
        1,
        2,
        [
            {
                m_Subnet="172.16.1.0",
                m_NetMask=24
        ]
);
```

Sample: defining an inclusion zone within a NAT domain

The following example defines one inclusion zone. The inclusion zone includes any device with an IP address starting with 172.16.2 that also belongs to the NAT address space NATDomain1. The protocol is set to 1, that is, IP.

```
insert into scope.zones
(
    m_Protocol, m_Action, m_Zones, m_AddressSpace
)
values
(
        1,
        1,
        [
            {
                m_Subnet="172.16.2.*",
            }
        ],
        "NATDomain1"
);
```

Sample: restricting device interrogation based on IP address

The following example shows how to prevent the further interrogation of devices that match a given IP address. Only devices that do not have the IP address 10.10.63.234 are interrogated further. Within the scope.detectionFilter table, specify:

- The filter condition(s). Only devices that pass this filter, that is, for which the filter evaluates true, are further investigated. If no filter is specified, all devices are passed through the detection filter.

```
insert into scope.detectionFilter
(
        m_Filter
)
values
(
        "( ( m_UniqueAddress <> '10.10.63.234' ) )"
);
```

A stitcher tests each discovered device against the filter condition in the scope.detectionFilter table, and the outcome of this test determines whether the device is discovered.

Because the process flow of the discovery is fully configurable, you can configure this stitcher to act at any time during the discovery process. By default, the stitcher performs the conditional test on the device details returned by the Details agent. Your filter must therefore be based on the columns of the Details.returns table.

Although you can configure the filter condition to test any of the columns in the Details.returns table, you might need to use the IP address as the basis for the filter to restrict the detection of a particular device. If the device does not grant SNMP access to the Details agent, the Details agent might not retrieve MIB variables such as the Object ID. However, you are guaranteed the return of at least the IP address when the device is detected.

The following examples show how else you might configure the detection filter.

Sample: restricting interrogation based on Object ID

The following example shows how to prevent the further interrogation of devices that match a given Object ID. The OQL `not like` clause indicates that only devices that pass the filter (that is, devices for which the OID is *not* like 1.3.6.1.4.1.*) are interrogated further.

The backslash must be used in the insert to escape the `.`, which would otherwise be treated as a wildcard. A full explanation of the syntax of OQL can be found in the *IBM Tivoli Network Manager Reference*.

```
insert into scope.detectionFilter
(
        m_Filter
)
values
(
        "(
                ( m_ObjectId not like '1\.3\.6\.1\.4\.1\..*' )
        )"
);
```

Sample: combining multiple filter restrictions

You can combine filter conditions within a single OQL insert. The following example ensures that only devices that do not have the specified OID and do not have the specified IP address are detected:

```
insert into scope.detectionFilter
(
        m_Filter
)
values
(
        "(
                ( m_ObjectId not like '1\.3\.6\.1\.4\.1\..*' )
                AND
```

```
                ( m_UniqueAddress <> '10.10.63.234' )
        )"
);
```

Restricting instantiation: limitation when filtering out interfaces

Note the following limitation when you restrict instantiation of interfaces.

Restriction: To ensure that alerts are not raised for *interfaces* that are excluded by the instantiation filter, you must set the `RaiseAlertsForUnknownInterfaces` variable. To this, perform the following steps:

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

```
update config.properties set RaiseAlertsForUnknownInterfaces = 0;
```

Sample: restricting instantiation based on entity name

To restrict the devices that are instantiated, append an OQL insert into the scope.instantiateFilter table. The instantiateFilter table requires a conditional test. Only devices that pass the filter are sent to ncp_model. If no filter is defined, all discovered devices are passed to ncp_model.

The conditional test must be based on the ncimCache data format.

The following example postdiscovery filter restricts instantiation of a chassis and its contents.

```
insert into scope.instantiateFilter
(
    m_Filter
)
values
(
    "
    (
        BASENAME != 'jane'
    )
    "
);
```

The following example postdiscovery filter restricts instantiation of a chassis and its contents.

```
insert into scope.instantiateFilter
(
    m_Filter
)
values
(
    "
    (
        snmpSystem->SYSDESCR NOT LIKE ' device'
    )
    "
);
```

Using scope.special to restrict device detection

Make entries in the scope.special table for network interfaces that can be accessed through multiple IP addresses. The entries in the scope.special table control which IP addresses Network Manager uses to monitor devices for NCMP and SNMP polling policies.

The following sample shows an INSERT statement to the scope.special table. It defines the IP address 192.168.1.3 as a potential management interface for chassis and interfaces. It provides extra customer

information that is added to the ExtraInfo section of the entity in the model master.entityByName database table if the IP address is discovered.

```
insert into scope.special
(
        m_Zones,
        m_Identifier,
        m_Priority,
        m_NonPingable,
        m_AdminInterface,
        m_ExtraInfo,
        m_Protocol,
        m_IsManagement,
        m_OutOfBand,
        m_IsValidVirtual
)
values
(
        [
                {
                        m_Subnet="192.168.1.3",
                        m_NetMask=32
                }
        ],
        "CustomerFacing",
        99,
        0,
        1,
        {
                m_CustomerName = 'MyCompany',
                m_CustomerType = 'Internal'
        },
        1,
        0,
        1,
        0
);
```

For a device that has 2 IP addresses, 172.20.1.1 and 192.168.1.3, the configuration means that 172.20.1.1 is not chosen as the IP address through which to manage the device. 192.168.1.3 is used instead. The following example shows what the final topology entry in the master.entityByName looks like in this instance. The data inside ExtraInfo that is prefixed with m_ScopeSpecial comes from the scope.zones entry that matched the IP address of 192.168.1.3.

```
{
        EntityName='192.168.1.3';
        Address=['','','192.168.1.3'];
        EntityType=1;
        EntityOID='1.3.6.1.4.1.8072.3.2.10';
        IsActive=1;
        Status=1;
        ExtraInfo={
                m_SysName='SYS1';
                m_DNSName='DNS1';
                m_time=1362486845;
                m_DisplayLabel='DNS1';
                m_AssocAddress=
[{m_IfIndex = 1, m_IpAddress = '172.20.1.1', m_Protocol = 1, m_IfOperStatus = 1 },
{m_IfIndex = 2, m_IpAddress = '192.168.1.3', m_Protocol = 1, m_IfOperStatus = 1 }];
                m_ScopeSpecialIsManagement=1;
                m_ScopeSpecialPriority=99;
                m_ScopeSpecialIdentifier='CustomerFacing';
                m_ScopeSpecialExtraInfo={
                          m_CustomerName = 'MyCompany',
                    m_CustomerType = 'Internal'
                    };
                m_DefinedMgmtIP=1;
                m_IsOutOfBand=1;
                m_BaseName='192.168.1.3';
                m_AddressSpace=NULL;
                m_AccessProtocol=1;
                m_AccessAddress='192.168.1.3';
                };
        LingerTime=3;
        ActionType=0;
        CreateTime=1362486848;
        ChangeTime=1362486848;
```

```
        ClassName='NetworkDevice';
        ClassId=5;
        ObjectId=2272;
}
```

Scope configuration for non-IP devices
Configure scope for non-IP devices by configuring inserts into the scope.detectionFilter database table within the DiscoScope.cfg configuration file.

Note: The detection filter runs against the data returned by the agents, in the `Details.returns` and `CollectorDetails.returns` tables, so scoping devices in this way has minimal impact on the target devices.

Sample: only pass non-IP devices for discovery

The following example insert ensures that only non-IP devices are passed to discovery and interrogated further. This filter would exclude all IP-based devices.

```
insert into scope.detectionFilter
(
        m_Filter,
 )
values
(
       "( m_Protocol = 4 )"
 );
```

Sample: combining multiple filter restrictions for non-IP devices

The following example insert ensures that only non-IP devices are passed to discovery and the non-IP devices that are passed must not include the specified string in their unique Element Management System (EMS) key.

```
insert into scope.detectionFilter
(
        m_Filter,
 )
values
(
       "(
            ( m_Protocol = 4 )"
             AND
            ( m_UniqueAddress NOT LIKE 'LONDON' )"
       )"
 );
```

Devices with out-of-scope interfaces
A network might contain devices that are within the discovery scope but that contain interfaces that are out of scope. Because the device is in scope, the default behavior of the layer 3 discovery agents is to download the interface table of the device and discover all the interfaces of a device, even if the interfaces themselves are out of scope.

If this situation applies to your network, and you want to modify the way in which the discovery process handles devices that are partially in scope, there are several ways to modify the discovery and monitoring process to exclude these interfaces from the discovery.

A possible configuration adjustment is to modify the insert into the scope.instantiateFilter such that the out-of-scope interfaces are not instantiated. This solution means that the out-of-scope interfaces are still discovered, but are not passed to MODEL to be instantiated to an Active Object Class (AOC); therefore the out-of-scope interfaces are not represented on the topology or monitored.

You could also configure an SNMP instance filter to prevent the downloading of SNMP data for certain interfaces.

You could also filter the data returned by the discovery agents using the DiscoAgentReturns.filter configuration file.

DiscoSnmpHelperFilters.cfg configuration file

The DiscoSnmpHelperFilters.cfg configuration file defines SNMP interface filters for the SNMP Helper. SNMP interface filters define the interfaces for which you want the SNMP Helper to retrieve information.

Database table used

The DiscoSnmpHelperFilters.cfg configuration file can be used to configure inserts into the snmpHelper.instanceFilter database table.

Example simple interface filter

The following example retrieves information for interfaces with a name like "Gi0" on a specific kind of device.

```
insert into snmpHelper.instanceFilter
(
    m_FilterName,
    m_DeviceFilter,
    m_InstanceFilter
)
values
(
    "TESTFILTER",
    "sysObjectID = '1.3.6.1.4.1.4874.1.1.1.1.3' OR sysDescr LIKE 'ERX-1440'",
    "ifName like 'Gi0'"
);
```

DiscoSnmpHelperSchema.cfg configuration file

The DiscoSnmpHelperSchema.cfg configuration file defines the operation of the SNMP Helper, which specifies the general rules of SNMP information retrieval.

Database table used

The DiscoSnmpHelperSchema.cfg configuration file can be used to configure inserts into the snmpHelper.configuration database table.

Sample: Configuring timeouts and threads

The following example configuration causes the SNMP helper to behave as follows:

- 120 threads of program execution are started to process incoming requests for SNMP data from the Helper Server. The SNMP helper processes a maximum of 120 such requests simultaneously.
- A three-second timeout period is specified for a device to respond to an SNMP query issued by the SNMP helper. If the device has not responded after this time, the helper issues the request again, one time.

```
insert into snmpHelper.configuration
(
        m_NumThreads,
        m_TimeOut,
        m_NumRetries,
)
values
(
    120, 3000, 1
);
```

DiscoTelnetHelperSchema.cfg configuration file

The DiscoTelnetHelperSchema.cfg configuration file defines the operation of the Telnet helper, which returns the results of a Telnet operation into a specified device.

Database tables used

The DiscoTelnetHelperSchema.cfg configuration file can be used to configure inserts into the following database tables:

- telnetHelper.configuration
- telnetHelper.deviceConfig

You can configure the Telnet helper to use the Secure Shell (SSH) program. SSH enables authentication and provides more secure communications over the network.

Sample: configuring the Telnet helper

The following insert can be appended to the DiscoTelnetHelperSchema.cfg configuration file to configure the operation of the Telnet helper. The insert configures the Telnet helper to:

- Use 20 threads of process execution
- Wait a maximum of 5000 ms for a reply from a device
- Try the request up to three times

```
insert into telnetHelper.configuration
(
          m_NumThreads,
          m_TimeOut,
          m_Retries
)
values
(
          20,
          5000,
          3
);
```

Configuring device-specific settings

The Telnet helper also accepts multiple inserts into the telnetHelper.deviceConfig table within the DiscoTelnetHelperSchema.cfg configuration file that define the interaction of the Telnet operation.

The following examples show how to configure Telnet device-specific settings. You can configure device settings based on the sysObjectID MIB variable or based on IP or subnet. The most effective way to set these options is based on the sysObjectID MIB variable. This variable identifies the device vendor. Device-specific configuration options typically vary with the device vendor. You can configure values for all Cisco devices, for example, regardless of where these devices are in the network.

Sample: configuring settings for devices from a specific vendor

The following typical configuration shows how to configure settings for all devices from a specific vendor. The insert specifies:

- 1.3.6.1.4.1.9.1. as the sysObjectID MIB variable to match for this configuration entry. All devices with object IDs of the form 1.3.6.1.4.1.9.1.* are matched. In general, these are Cisco IOS devices, although there are exceptions.
- terminal length is the command that sets the output page length for Cisco devices.

  Note: This command varies with devices of different vendor types.
- No paging
- Prompt from remote device

- The response to send to the remote device for it to continue paged output.

```
insert into telnetHelper.deviceConfig
(
          m_SysObjectId,
          m_PageLengthCmd,
          m_PageLength,
          m_ContinueMsg,
          m_ContinueCmd
)
values
(
          "1.3.6.1.4.1.9.1.", "terminal length", 0, ".*[Mm]ore.*", " "
);
```

The DiscoTelnetHelperSchema.cfg configuration file contains inserts with default device-specific configuration settings for the following vendor types:

- Cisco IOS devices
- Cisco Cat OS devices
- Juniper JUNOS devices
- Juniper ERX devices
- Huawei devices
- Dasan devices

Sample: configuring device response settings based on IP address

If the output of the telnet command is longer than one page, the device sends a message asking whether to display the next page. Configure the messages to be expected and the responses to be given by the Telnet helper in the DiscoTelnetHelperSchema.cfg configuration file.

Commands beginning m_Continue (such as m_ContinueMsg) and m_PageLength (such as m_PageLengthCmd) are mutually exclusive: you must use one or the other. If these settings are not configured correctly for your devices, data might be lost.

The following example shows how to configure settings for devices based on the IP address. The insert specifies:

- 192.168.112.0 as the IP address
- The prompt from the remote device is a regular expression containing "`wish to continue`"
- The response to send to the remote device for it to continue paged output is "y"

```
insert into telnetHelper.deviceConfig
(
          m_IpOrSubNet,
          m_NetMaskBits,
          m_Protocol,
          m_ContinueMsg,
          m_ContinueCmd
)
values
(
          192.168.112.0,
          24,
          1,
          ".*wish to continue.*",
          "y"
);
```

DiscoXmlRpcHelperSchema.cfg configuration file

The DiscoXmlRpcHelperSchema.cfg configuration file can be used to configure the XML-RPC helper, which enables Network Manager to communicate with EMS collectors using the XML-RPC interface.

Database table used

The DiscoXmlRpcHelperSchema.cfg configuration file can be used to configure inserts into the xmlRpcHelper.configuration database table.

This example insert configures the XML-RPC helper to:

- Use one thread of process execution.
- Allow a maximum size of 1048576 bytes for an XML-RPC response.

```
insert into xmlRpcHelper.configuration
(
    m_NumThreads,
    m_MaxResponseSize
)
values
(
    1, 1048576
);
```

Note: The default maximum response size might be too small when running a Collector-based discovery against Collectors that result in very large responses. In such cases, increase the maximum response size. To increase the maximum response size, set the **m_MaxResponseSize** parameter to a higher value. Make sure you set the same value for **m_MaxResponseSize** in both of the following files:

- `NCHOME/etc/precision/DiscoCollectorFinderSchema.cfg`
- `NCHOME/etc/precision/DiscoXmlRpcHelperSchema.cfg`

SnmpStackSecurityInfo.cfg configuration file

The SnmpStackSecurityInfo.cfg configuration file defines the community strings, versioning, and other properties used by any process that needs to interrogate devices using SNMP, for example, the SNMP helper. Community strings can be configured on a per-device or per-subnet basis, to allow the SNMP Helper to retrieve MIB variables from devices.

Database tables used

This configuration file can be used to configure inserts into the following database tables:

- snmpStack.configuration
- snmpStack.verSecurityTable
- snmpStack.accessParameters

Note that there is another configuration file associated with the snmpStack database, the SnmpStackSchema.cfg file, but you should not need to alter this file.

Sample: Configuring SNMP versions

If auto-versioning is turned on, the following configuration adjustment specifies that a community string of '`public`' is used for devices that support SNMP version 1, and specific configuration is used for devices that support SNMP version 3. Since no value has been specified for `m_SnmpPort`, this value defaults to the standard SNMP 161 port.

```
insert into snmpStack.verSecurityTable
(
        m_SNMPVersion,
        m_Password,
        m_SNMPVer3Level,
        m_SNMPVer3Details,
        m_SecurityName,
)
```

```
values
(
        0,
        'public',
        2,
        {
            m_AuthPswd="authpassword",
            m_PrivPswd="privpassword"
        },
        'authPriv'
);
```

Sample: Defining community strings

The following inserts define the community strings `public` and `crims0n` for use to access SNMP devices.

You can append as many inserts as there are passwords to the `SnmpStackSecurityInfo.cfg` configuration file. All password and subnet configurations are tried until a match is found.

Note: Only one SNMP community string, the `public` community string, is set up by default.

```
insert into snmpStack.verSecurityTable
(
        m_SNMPVersion,
        m_Password,
        m_SNMPVer3Level,
        m_SNMPVer3Details,
        m_SecurityName
)
values
(
        0,
        'public',
        2,
        {
            m_AuthPswd="authpassword",
            m_PrivPswd="privpassword"
        },
        'authPriv'
);

insert into snmpStack.verSecurityTable
(        m_IpOrSubNetVer,
        m_NetMaskBitsVer,
        m_SNMPVersion,
        m_Password,
        m_SNMPVer3Level,
        m_SNMPVer3Details,
        m_SecurityName
)
values
(
        "10.10.2.0",
        24,
        0,
        'crims0n',
        2,
        {
            m_AuthPswd="authpassword",
            m_PrivPswd="privpassword"
        },
        'authPriv'
);
```

Sample: Specifying an SNMP port

This example configures the same SNMP settings as in the previous example on all devices within the subnet 192.168.64.0 and specifies the SNMP port as 6161 on all devices within this subnet.

```
insert into snmpStack.verSecurityTable
(
        m_IpOrSubNetVer,
        m_NetMaskBitsVer,
        m_SNMPVersion,
        m_Password,
        m_SNMPVer3Level,
```

```
        m_SNMPVer3Details,
        m_SecurityName,
        m_SnmpPort,
)
values
(
        192.168.64.0,
        24,
        0,
        'public',
        2,
        {
            m_AuthPswd="authpassword",
            m_PrivPswd="privpassword"
        },
        'authPriv'
        6161
);
```

TelnetStackPasswords.cfg configuration file

The TelnetStackPasswords.cfg configuration file defines access credentials for Telnet access to devices.

You can use the TelnetStackPasswords.cfg configuration file to specify a Secure Shell (SSH) connection when configuring Telnet device access. SSH enables password encryption when performing Telnet access. SSH versions 1 and 2 are supported (restrictions apply in FIPS mode).

Important: SSH within Network Manager currently supports password-based authentication or no authentication. It does not support RSA signature authentication.

Database table used

The TelnetStackPasswords.cfg configuration file can be used to configure inserts into the telnetStack.passwords database table.

Note that there is another configuration file associated with the telnetStack database, the TelnetStackSchema.cfg file, but you should not need to alter this file.

Sample: Configuring Telnet access parameters for a subnet

The following example insert configures the Telnet access parameters for a subnet. The insert specifies:

- A subnet address of `192.168.200.0` with a netmask of 25.
- The password and username to use to access the device.
- The password, login and console prompts to expect from the device.
- The devices on this subnet support SSH.

```
insert into telnetStack.passwords
(
        m_IpOrSubNet,
        m_NetMaskBits,
        m_Password,
        m_Username,
        m_PwdPrompt,
        m_LogPrompt,
        m_ConPrompt,
        m_SSHSupport
)
values
(
        '192.168.200.0',
        25,
        '3v3rt0n',
        'user',
        '.*assword:.*',
        '.*ogin.*',
        '.*onsole>.*',
        1
);
```

Sample: Configuring Telnet access parameters for a device

The following example insert shows how you can configure the access parameters for a single IP address. The insert specifies:

- A single IP address of `172.16.1.21`. The address is identified as a single address by the fact that `m_NetMaskBits=32`.
- The password and username to use to access the device.
- The password, login and console prompts to expect from the device.
- This device does not support SSH.

```
insert into telnetStack.passwords
(
        m_IpOrSubNet,
        m_NetMaskBits,
        m_Password,
        m_Username,
        m_PwdPrompt,
        m_LogPrompt,
        m_ConPrompt,
        m_SSHSupport
)
values
(
        '172.16.1.21',
        32,
        '',
        '',
        '.*assword.*',
        '.*sername.*',
        '.*Morr.*',
        0
);
```

Sample: Configuring Telnet device-access for a subnet

The following example insert configures the Telnet access parameters for a subnet. The insert specifies:

- A subnet address of `192.168.200.0` with a netmask of 25.
- The password and username to use to access the device.
- The password, login and console prompts to expect from the device.
- The devices on this subnet support SSH.

```
insert into telnetStack.passwords
(
        m_IpOrSubNet,
        m_NetMaskBits,
        m_Password,
        m_Username,
        m_PwdPrompt,
        m_LogPrompt,
        m_ConPrompt,
        m_SSHSupport
)
values
(
        '192.168.200.0',
        25,
        '3v3rt0n',
        'user',
        '.*assword:.*',
        '.*ogin.*',
        '.*onsole>.*',
        1
);
```

Sample: Configuring Telnet device-access for a single IP address

The following example insert shows how you can configure the access parameters for a single IP address. The insert specifies:

- A single IP address of `172.16.1.21`. The address is identified as a single address by the fact that `m_NetMaskBits=32`.
- The password and username to use to access the device.
- The password, login and console prompts to expect from the device.
- This device does not support SSH.

```
insert into telnetStack.passwords
(
        m_IpOrSubNet,
        m_NetMaskBits,
        m_Password,
        m_Username,
        m_PwdPrompt,
        m_LogPrompt,
        m_ConPrompt,
        m_SSHSupport
)
values
(
        '172.16.1.21',
        32,
        '',
        '',
        '.*assword.*',
        '.*sername.*',
        '.*Morr.*',
        0
);
```

Retrieving extra information

You can configure the discovery agents to retrieve extra information from devices and store this information in the ExtraInfo column of the topology database.

About this task

To specify that extra information be retrieved by a given discovery agent, modify the definition file of the agent ($NCHOME/precision/disco/agents/*.agnt). All discovery agents have a definition file in the agents directory, regardless of whether the agent is text-based or precompiled.

The changes that you must make to the agent definition are described in the following topics.

Changing the agent type

You can change the agent type in the agent definition file.

About this task

At the start of the discovery agent definition file, one of the following types of agent is identified:

Procedure

- `DiscoCompiledAgent{}`: Denotes a compiled discovery agent (with a corresponding shared library in the `$NCHOME/precision/lib` directory).
- `DiscoDefinedAgent{}`: Denotes a text-based discovery agent (with no corresponding shared library).
- `DiscoCombinedAgent{}`: Denotes a discovery agent that is a combination of text-based and precompiled, where extra processing (such as the retrieval of extra information from devices) is defined in the discovery agent definition file.

Results

To retrieve extra information from devices, the agent type must either be `DiscoDefinedAgent{}` or `DiscoCombinedAgent{}`. Therefore, if you are modifying an existing compiled agent to retrieve extra information, the first step is to change the type of agent from `DiscoCompiledAgent{}` to `DiscoCombinedAgent{}`.

Mediation and processing layers

The retrieval of extra information from devices and the addition of the information to the entity records is conducted in two layers: the mediation and processing layers. In the mediation layer, the actual SNMP requests to retrieve the variables are carried out. In the processing layer, the retrieved variables are added to the appropriate entity records. There is also an optional filter on the mediation layer.

The following code segment is an overview of the structure of the mediation and processing sections of the discovery agent definition file.

```
DiscoAgentMediationFilter
            {
                    // Optional section containing filters for the mediation layer.
            }

DiscoAgentMediationLayer
            {
                    // Contains the SNMP Get and GetNext requests to be performed.
                    // In addition, an ICMP trace can be performed and SNMP access
                    // parameters can be retrieved in the mediation layer.
            }

DiscoAgentProcessingLayer
            {
                    // Adds the retrieved variables to the appropriate entity
                    // record(s).
            }
```

The mediation layer

The mediation layer is where the SNMP and ICMP requests are performed.

In the following code, the `DiscoSnmpGetResponse();` rule performs an SNMP Get request, and the `DiscoSnmpGetNextResponse();` rule performs an SNMP Get Next request. You can include as many of each type of request as necessary.

You can also include the `DiscoSnmpGetAccessParameters();` rule, which retrieves the SNMP access details for the device, and the `DiscoICMPGetTrace();` rule, which retrieves all the IP addresses in the path to the device.

```
DiscoAgentMediationLayer
  {
        DiscoSnmpRequests
                {
                        DiscoSnmpGetResponse( ARGUMENT, VARIABLE );
                        DiscoSnmpGetNextResponse( ARGUMENT, VARIABLE, );
                        DiscoSnmpGetAccessParameters( VARIABLE );
                }
        DiscoICMPRequests
                {
                        DiscoICMPGetTrace( VARIABLE );
                }
  }
```

DiscoSnmpGetResponse();

DiscoSnmpGetResponse(); performs an SNMP Get request. The simple form of this rule takes two arguments, separated by a comma. The first argument is the key to assign to the response. This key is used in the processing layer. The second argument is the OID (Object ID) to retrieve from the device.

The following example retrieves sysUpTime, assigning the key m_SysUpTime to the value that is returned.

```
DiscoSnmpGetResponse( "m_SysUpTime", sysUpTime );
```

A more complex form of DiscoSnmpGetResponse(); takes a third argument, the OID index. The following example retrieves ifDescr, assigns the key m_IfDescr to the value returned, and uses the OID index 1.

```
DiscoSnmpGetResponse( "m_IfDescr", ifDescr, "1" );
```

DiscoSnmpGetNextResponse();
DiscoSnmpGetNextResponse(); performs an SNMP GetNext request. This rule takes the same arguments as DiscoSnmpGetResponse();.

The following example retrieves ipRouteIfIndex and assigns the key m_IpRouteIfIndex to the value returned.

```
DiscoSnmpGetNextResponse( "m_IpRouteIfIndex", ipRouteIfIndex );
```

DiscoSnmpGetAccessParameters();
`DiscoSnmpGetAccessParameters();` retrieves the SNMP access parameters for the device.

If you configure the discovery agent to retrieve the access parameters in the mediation layer, you must also configure the agent to add the information to the database record in the processing layer.

```
DiscoSnmpGetAccessParameters( "m_AccessParam" );
```

DiscoICMPGetTrace();
`DiscoICMPGetTrace();` retrieves the IP addresses in the path to the device.

If you configure the discovery agent to retrieve the path to the device in the mediation layer, you must also configure the agent to add the information to the database record in the processing layer.

```
DiscoICMPGetTrace( "m_Trace" );
```

Mediation layer filter
The mediation layer filter is an optional filter that restricts the SNMP requests for extra information to specific devices. You can include a condition within the DiscoMediationSnmpGetFilter{} section within the DiscoAgentMediationFilter{}, that only devices passing the filter are processed by the agent.

The following example ensures that only devices with an ipForwarding value of 1 are processed.

```
DiscoAgentMediationFilter
            {
                    DiscoMediationSnmpGetFilter
                    {
                            "ipForwarding" = 1 ;
                    }
            }
```

The processing layer
The processing layer is where the retrieved information is added to the entity records. Both the DiscoAgentProcLayerAddTags{} and DiscoAgentProcLayerAddLocalTags{} sections are optional. However, if both sections are omitted, no extra information is stored in the database records.

The structure of the processing layer is shown below.

```
DiscoAgentProcessingLayer
  {
      DiscoAgentProcLayerAddTags
              {
                   DiscoAddTagSnmpGet( KEY );
                   DiscoAddTagSnmpGetNext( KEY );
                   DiscoAddTagSnmpGetAccessParameters( "m_AccessParam" );
                   DiscoAddTagTrace( "m_Trace" );
              }
      DiscoAgentProcLayerAddLocalTags
              {
                   DiscoAddTagSnmpGet(
                                TAG FROM KEY WHERE CONDITION );
                   DiscoAddTagSnmpGetNext(
```

```
                                    TAG FROM KEY WHERE CONDITION );
                }
    }
```

DiscoAgentProcLayerAddTags{}
Within the `DiscoAgentProcLayerAddTags{}` section, you can include as many `DiscoAddTagSnmpGet();` or `DiscoAddTagSnmpGetNext();` rules as necessary. These rules add the retrieved variable to the database record for the discovered entity.

Each rule within the `DiscoAgentProcLayerAddTags{}` section takes a single argument, which is the key assigned to the retrieved variable in the mediation layer. The following example adds the value of `m_SysUpTime`, retrieved in the mediation layer, to the entity record.

```
DiscoAddTagSnmpGet( "m_SysUpTime" );
```

If you configured the discovery agent to retrieve either the SNMP access parameters or the path to the device during the mediation layer, you must include either the `DiscoAddTagSnmpGetAccessParameters();` or the `DiscoAddTagTrace();` rule in the `DiscoAgentProcLayerAddTags{}` section to ensure that the retrieved information is added to the MODEL database.

DiscoAgentProcLayerAddLocalTags{}
Within the `DiscoAgentProcLayerAddLocalTags{}` section, you can include as many `DiscoAddTagSnmpGet();` or `DiscoAddTagSnmpGetNext();` rules as necessary. These rules add the retrieved variable to the database record for a local neighbor.

The structure of the rules is:

```
DiscoAddTagSnmpGet( TAG FROM KEY WHERE CONDITION );
DiscoAddTagSnmpGetNext( TAG FROM KEY WHERE CONDITION );
```

The arguments that determine the local neighbor to which the tag is added are:

- *TAG* specifies the field name of the tag to be added.
- *KEY* indicates the key assigned to the value returned in the mediation layer.
- *CONDITION* indicates a condition that determines whether or not the tag is added.

The following example adds a field called `m_IfDescr` to the local neighbor object (using the value returned in the mediation layer that was assigned the key `m_IfDescr`) where `m_IfIndex=1`.

```
DiscoAddTagSnmpGet( "m_IfDescr" FROM "m_IfDescr"
                                    WHERE ( "m_IfIndex" = "1" )
                            );
```

The following example adds a field called `m_IfType` to the local neighbor object using the list of values returned by the GetNext request performed in the mediation layer and assigned the key `m_IfType`. The `WHERE` clause indicates the particular value required from the list of data. The value is retrieved by looking for the entry where the value of the `m_IfIndex` field in the local neighbor object is equal to `SNMPINDEX(0)`, that is, the first value of the SNMP table entry.

```
DiscoAddTagSnmpGetNext( "m_IfType" FROM "m_IfType"
                                    WHERE ( "m_IfIndex" = SNMPINDEX(0) )
                                );
```

Configuring trap forwarding

The SNMP trap multiplexer, the **ncp_trapmux** process, listens to a single port and forwards all the traps it receives to a set of host/socket pairs.

Restriction: The SNMP trap multiplexer does not forward SNMPv3 Inform messages.

About trap management

Trap management enables you to ensure that traps received from network devices are forwarded to ports where they can be handled by Network Manager and other network management systems.

In most networks, traps arrive on a single default port (usually port 162). This can cause problems if you have Network Manager and another network management system installed on the same server. Both of these systems might need to listen for traps; however, only one process can bind to one port at a time.

The SNMP Trap Multiplexer is a Network Manager process that resolves this problem: it listens to a single port and forwards all the traps it receives to a set of host/socket pairs.

By default, the SNMP Trap Multiplexer listens for traps on port 162, but you can change this by inserting an alternative port number into the trapMux.config database table.

The ncp_trapmux process can also store trap events in a binary format file (containing trap and timing information) that can be used to recreate the trap events in the order they occurred at a later date. This is useful mainly for debugging purposes.

Starting the SNMP trap multiplexer

Although it is good practice to ensure that the **ncp_ctrl** process is configured to launch and manage the SNMP Trap Multiplexer, you can also start it manually.

About this task

To start the **ncp_trapmux** process, use the following command:

```
ncp_trapmux -domain DOMAIN_NAME
```

Forwarding traps

Using the SNMP Trap Multiplexer, you can forward traps to one or more servers.

About this task

To configure the SNMP Trap Multiplexer to forward traps to network management systems running on host1 and host2:

Procedure

1. Edit the schema file, $NCHOME/etc/precision/TrapMuxSchema.cfg, to contain a set of host/socket pairs. For example, append the following lines to the file:

   ```
   insert into trapMux.sinkHosts (host, port) values ("host1", 5999);
   insert into trapMux.sinkHosts (host, port) values ("host2", 5999);
   ```

2. Start the SNMP Trap Multiplexer using the following commands:

   ```
   ncp_trapmux -domain DOMAIN1
   ncp_trapmux -domain DOMAIN2
   ```

Results

In the above example, when a trap is sent to the server that is running the **ncp_trapmux** process, it is forwarded to test-host1, port 5999 and test-host2, port 5999.

Starting trap capture

You can start capturing traps by inserting commands into the SNMP Trap Multiplexer database.

About this task

To instruct the SNMP Trap Multiplexer to start capturing traps:

Procedure

1. Log into the TrapMux service using the OQL Service Provider or the Management Database Access page.
2. Issue the following commands:

```
insert into trapMux.command
(command) values( "capture_start" );
go
```

Stopping trap capture
You can stop capturing traps by inserting commands into the SNMP Trap Multiplexer database.

About this task

To instruct the SNMP Trap Multiplexer to stop capturing traps:

Procedure

1. Log into the TrapMux service using the OQL Service Provider or the Management Database Access page.
2. Issue the following commands:

```
insert into trapMux.command
(command) values( "capture_stop" );
go
```

Printing traps to a file
You can print traps to a file by inserting commands into the SNMP Trap Multiplexer database.

About this task

To instruct **ncp_trapmux** to print traps:

Procedure

1. Log into the TrapMux service using the OQL Service Provider or the Management Database Access page.
2. Issue the following commands:

```
insert into trapMux.command
(command, fileName) values( "print", FILENAME );
go
```

Results

Where *FILENAME* specifies the file to which the output is written. If the file is not specified, `$NCHOME/etc/precision/trapmux.out` is used.

Replaying traps from a file
If you have created a text-readable file for traps, you can use the **ncp_trapmux** process to recreate the trap events in the order specified in this file.

About this task

The **ncp_trapmux** process can replay traps using a binary file or a human-readable file, however, the **ncp_trapmux** process can only generate binary files.

To instruct **ncp_trapmux** to replay traps from a file:

Procedure

1. Log into the TrapMux service using the OQL Service Provider or the Management Database Access page.
2. Issue the following commands:

```
insert into trapMux.command
(command, fileName) values( "replay", "trapmux.out" );
go
```

SNMP trap multiplexer commands

You can issue commands to the SNMP trap multiplexer, the ncp_trapmux process, to control its operation.

The commands used to control the ncp_trapmux process are described in the following table:

Table 41. Commands used to control the ncp_trapmux process

Command	Function and Default Filename
`capture_start`	Begin logging traps to memory. The default filename is NULL (not required).
`capture_stop`	Stop logging traps to memory. The default filename is NULL (not required).
`capture_continue`	Continue logging traps to memory. The default filename is NULL (not required).
`capture_empty`	Clear memory of all currently logged traps. The default filename is NULL (not required).
`rehash`	Shut down the ncp_trapmux process and clear all memory. The daemon then rereads the configuration file and starts up again. The default filename is NULL (not required).
`restart`	Set the daemon to normal mode. The default filename is NULL (not required).
`replay`	Either read the traps in memory or read the raw trap packet information in the specified file and replay the traps with a small delay between each. The default filename is NULL (play from memory).
`replay timed`	Either read the traps in memory or read the raw trap packet information in the specified file and replay the traps in the order of the time they were received with the same delays between traps. The default filename is NULL (play from memory).
`print`	Print the current traps in memory in a non-readable format to the specified file. Time information is encoded with the trap. The default filename is `$NCHOME/etc/precision/trapmux.out`.

Types of traps

Traps are administrative messages sent from network devices such as routers that indicate that the device or its connections have started or stopped.

About this task

The Trap finder discovers devices by listening for SNMP traps and extracting IP addresses from those traps. The different types of traps are described in Table 42 on page 240.

Table 42. Types of traps		
Number	**Name**	**Description**
0	coldStart trap	A coldStart trap signifies that the sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation may be altered.
1	warmStart trap	A warmStart trap signifies that the sending protocol entity is reinitializing itself such that neither the agent configuration nor the protocol entity implementation is altered.
2	linkDown trap	A linkDown trap is generated by the failure of a recognized communication link.
3	linkUp trap	A linkUp trap is generated when a communication link that was formerly down comes alive.
4	authenticationFailure trap	An authenticationFailure trap is generated by a protocol message that has not been authenticated by the recipient, for example, an incorrect password.
5	egpNeighborloss trap	An egpNeighborLoss trap signifies that an Exterior Gateway Protocol (EGP) neighbor for which the sending protocol entity was an EGP peer has been marked down and the peer relationship is no longer valid.
6	enterprise-specific trap	An enterprise-specific trap signifies that the sending protocol entity recognizes that an enterprise-specific event has occurred.

Filtering SNMP information from devices

You can filter out SNMP information when devices are queried by the SNMP helper by configuring inserts into the SNMP helper database.

SNMP interface filtering

You can filter the SNMP data retrieved from devices by the discovery process by configuring SNMP interface filters. You can configure SNMP interface filters only on the command line.

Why use SNMP interface filtering

Sometimes a device or a class of devices returns too much MIB data. For example, if virtual devices have a large number of interfaces, discovering them can take a long time. In order to speed up discovery of such devices, you can use SNMP interface filters to reduce the number of interfaces that are retrieved by the SNMP helper.

How SNMP interface filtering works

When discovery agents, Perl scripts, or the **SNMP MIB Browser** request SNMP information, the SNMP helper retrieves the information from network devices. SNMP interface filters define rows in MIB tables that the SNMP helper retrieves. The SNMP helper retrieves a subset of the information that would have been returned without a filter, and sends it to the process that requested the SNMP information. SNMP interface filters can also define entire tables that are not to be retrieved by the SNMP Helper.

SNMP interface filters are only applied to requests for full walks of MIB tables. SNMP Get or GetNext requests for specific interfaces within a MIB table are not filtered.

The devices that should have the filter applied to them are defined in the *device filter*. If a device filter is defined, any request for SNMP information for a device is first checked against the device filter. Only devices which pass the filter are then checked for interface filtering.

The filter can filter on multiple rows of a table. The first time that a filtered table is accessed, one or more columns in the table are walked. All subsequent requests for SNMP walks of that table return only interfaces that match the filter.

Including multiple tables with dependent filters

You can also define *dependent filters*. If an SNMP interface filter *Filter 1* is defined on Table A, then a second, dependent filter *Filter 2* can be defined on Table B. SNMP information in Table B that relates to the same interface is also retrieved.

In order to define a dependent filter, in addition to a filter being defined on Table A, one or more of the following conditions must be true:

- Table A and Table B have equivalent indexes.
- The index of Table A is a value in Table B.

If Table A and Table B have exactly the same index, then you do not need to define a dependent filter. Information from Table B is automatically retrieved according to the filter defined on Table A.

When you can use SNMP interface filtering

You can use SNMP interface filtering on any SNMP MIB table that is keyed on ifIndex. For example, you filtering on ifTable or ifXTable allows filtering on values such as ifType and ifDescr.

Restriction: Filtering on any SNMP MIB variable other than interfaces is not supported. You can, however, block access to any table using the m_InstanceFilterTable option.

The following example extract shows the definition for the ipAddrTable from the `NCHOME/precision/mibs/RFC1213.mib` file:

```
          -- the IP address table

          -- The IP address table contains this entity's IP addressing
          -- information.

          ipAddrTable OBJECT-TYPE
              SYNTAX  SEQUENCE OF IpAddrEntry
              ACCESS  not-accessible
              STATUS  mandatory
              DESCRIPTION
                      "The table of addressing information relevant to
                      this entity's IP addresses."
              ::= { ip 20 }
```

The syntax "SEQUENCE OF" defines this as a table. You can find out what tables are defined in the MIB by searching for this string. The following example shows the output of a search run on UNIX:

```
% grep 'SEQUENCE OF'  RFC1213.mib
              SYNTAX  SEQUENCE OF IfEntry
              SYNTAX  SEQUENCE OF AtEntry
              SYNTAX  SEQUENCE OF IpAddrEntry
              SYNTAX  SEQUENCE OF IpRouteEntry
              SYNTAX  SEQUENCE OF IpNetToMediaEntry
              SYNTAX  SEQUENCE OF TcpConnEntry
              SYNTAX  SEQUENCE OF UdpEntry
              SYNTAX  SEQUENCE OF EgpNeighEntry
```

Configuring SNMP interface filters

You can configure one or more interface filters per device type.

About this task

To configure one or more interface filters, complete the following steps:

Procedure

1. Ensure that the Entity agent is enabled. You can enable the Entity agent in the **Discovery Configuration GUI**.
2. Back up and edit the following file:

   NCHOME/etc/precision/DiscoSnmpHelperFilters.*DOMAIN_NAME*.cfg
3. Specify an insert in the snmpHelper.instanceFilter table. You can use the examples given later in this procedure or copy another example insert from this documentation.
4. Give this filter a name. Locate the value of m_FilterName and give the filter a descriptive name, enclosed in double quotes.
5. Configure which devices should have this interface filter applied to them by defining a device filter. Locate the value of m_DeviceFilter and define a filter, enclosed in double quotes. You can use any Object Identifiers (OIDs) to construct the filter. Use Object Query Language (OQL) syntax.

   The filter must be in the following form:

   ```
   mibVariableName expression values
   [ optional_Boolean_operator expression optional_Boolean_operator .. ]
   ```

   For example, the following are all valid filters:

   ```
   // Apply the interface filter to only a specific type of device
   sysObjectID = '1.3.6.1.4.1.4874.1.1.1.1.3'


   // More complex example of the above
   sysObjectID = '1.3.6.1.4.1.4874.1.1.1.1.3' OR sysDescr LIKE 'ERX-1440'

   // Apply the interface filter only to devices in certain locations
   sysLocation in ( 'location1', 'location2' )

   //Apply the interface filter to all types of device.
   sysObjectID != ''
   ```
6. Define the SNMP interface filter expression to be applied to the MIB tables. Only rows that match this filter, that is, for which this expression evaluates true, are retrieved. Locate the value of m_InstanceFilter and define a filter, enclosed in double quotes. You can use any Object Identifiers (OIDs) to construct the filter. Use Object Query Language (OQL) syntax. You can define more than one interface filter per named filter.

   The filter must be in the following form:

   ```
   mibVariableName expression values
   [ optional_Boolean_operator expression optional_Boolean_operator .. ]
   ```

   For example, the following are all valid filters:

   ```
   // Only interfaces with names like this are returned
   ifName like 'Gi0'

   // This filter is against 2 distinct tables (ifTable and ifXTable)
   with the requirement that these share a common index (ifIndex)
   ifName like 'Gi0' or ifDescr like 'FastEthernet'

   // Filter out interfaces of these types
   ifType not in ( 1, 53, 166 )
   ```
7. Optional: If you want to filter out information that relates to the same interface in related MIB tables that are not explicitly defined with the same index, specify which tables to include using a *dependent filter*. Copy and edit a default insert into the snmpFilter.dependentInstances table from the NCHOME/etc/precision/DiscoSnmpHelperFilters.*DOMAIN_NAME*.cfg file or copy an example insert from this documentation.

   The syntax of the filter is

   ```
   MIB_variable_name in
   (eval(list_type,'&MIB_table.MIB_entry'))
   ```

where *MIB_variable_name* must exist in *MIB_table*, and a filter on *MIB_table* has been defined in the snmpHelper.instanceFilter table.

8. Run the following command to instruct the SNMP helper, ncp_dh_snmp, to re-read its configuration files:

```
kill -HUP PID
```

9. You can configure more than one SNMP interface filter in the `NCHOME/etc/precision/DiscoSnmpHelperFilters.DOMAIN_NAME.cfg` file. If more than one interface filter is configured, a table row that matches any filter is retrieved.
10. If one or more interface filters are configured, ensure that the the `RaiseAlertsForUnknownInterfaces` property in the `NCHOME/etc/precision/NcPollerSchema.cfg` file is set to `0`. This setting ensures that alerts are not raised against interfaces that were not discovered, for example, because they were filtered out of the discovery by interface filters.

Example simple interface filter

The following example retrieves information for interfaces with a name like "Gi0" on a specific kind of device.

```
insert into snmpHelper.instanceFilter
(
    m_FilterName,
    m_DeviceFilter,
    m_InstanceFilter
)
values
(
    "TESTFILTER",
    "sysObjectID = '1.3.6.1.4.1.4874.1.1.1.1.3' OR sysDescr LIKE 'ERX-1440'",
    "ifName like 'Gi0'"
);
```

Example dependent SNMP interface filter

You can create dependent filters that use the results of previous filters to filter additional tables.

In the following example, the ipNetToMediaTable is indexed on ipNetToMediaIfIndex, which is equivalent to ifIndex. The ifTable table is also indexed on ifIndex. This relationship is written into the MIB definition, but cannot be programmatically determined. Entering the relationship as a dependent filter ensures that matching entries in the ipNetToMediaTable are also included.

```
insert into snmpHelper.dependentInstanceFilter
(
    m_InstanceFilter
)
values
(
    "ipNetToMediaIfIndex in ( eval(list type int, '&ifTable.ifEntry') )"
)
;
```

In the following example, the ipAddrTable is indexed on IP address, but each row contains ipAdEntIfIndex, whose value is equivalent to ifIndex. The ifTable table is also indexed on ifIndex. This relationship is written into the MIB definition, but cannot be programmatically determined. Entering the relationship as a dependent filter ensures that matching entries in the ipAddrTable are also included.

```
insert into snmpHelper.dependentInstanceFilter
(
    m_InstanceFilter
)
values
(
    "ipAdEntIfIndex in ( eval(list type int, '&ifTable.ifEntry') )"
)
;
```

Both these example filters return rows that match the filter. If a row in the MIB table does not have a valid value, it is not retrieved. For example, if a row in the ipAddrTable table does not contain a valid value for ipAdEntIfIndex, it is not retrieved by the SNMP helper.

Both these example filters rely on an interface filter already being defined on ifIndex. The matching results from the ipAddrTable and ipNetToMediaTable tables are cached, so that the MIB is not queried again.

Both of the above filters are configured by default.

Configuring backups of the ncp_store cache

You can back up and restore the `ncp_store` cache, so that you have a copy of the topology.

About this task

When backups are configured, the `RunCacheCopy.stch` stitcher is called as part of the discovery. This stitcher runs the `RunCacheCopy.pl` script, which backs up the cache files. You do not need to modify the stitcher or the script.

Cache files are copied from `$NCHOME/var/precision/` to `$NCHOME/var/precision/backup/StoreCacheBackup/`. Within this directory, a subdirectory is created called `Full_`*`domain_timestamp`*.

Procedure

1. Backups are off by default. To enable and configure backups, edit the `EventGatewaySchema.cfg` file.
   a) Back up and edit the `$NCHOME/etc/precision/EventGatewaySchema.cfg` file.
   b) Locate the insert into the `config.defaults` database table and edit the values.
   c) Set `backupDiscoveryCaches` to `1` to enable the backups, or `0` to disable them.
   d) Set `numberOfBackupsToKeep` to the number of backups to keep. After this number of backups have been made, older ones are deleted.
2. Restart the core components using the `itnm_stop ncp` and `itnm_start ncp` commands.
3. To restore the backups:
   a) Shut down the core components using the `itnm_stop ncp` command.
   b) Copy the cache files from `$NCHOME/var/precision/backup/StoreCacheBackup/` back to `$NCHOME/var/precision/`.
   c) Restart the core components using the `itnm_start ncp` command.

Configuring specialized discoveries

You can configure the system to perform more complex discoveries, such as MPLS and NAT discovery.

About this task

Specialized discoveries include the following:

EMS discoveries
Collects topology data from Element Management Systems and integrates this data into the discovered topology.

MPLS discoveries
Discovers layer 3 VPNs and enhanced layer 2 VPNs running across MPLS core networks.

NAT discoveries
Discovers NAT gateway devices thereby enables you to retrieve data on devices in private address spaces.

Configuring a context-sensitive discovery

If you need to discover devices that support multiple contexts, you must run a context-sensitive discovery. For example, a device that uses separate SNMP contexts to provide access to its virtual routers. Context-sensitive discovery ensures correct representation of context-accessible virtual devicess. Always check that your particular device type is supported for discovery.

About this task

In a context-sensitive discovery, information about a device is passed from the `returns` table of the Details agent to the `despatch` table of the relevant Context agent.

The Context agents use the filters in the files with an extension of .agnt to determine which devices to process. This is true for all discovery agents. If the device is not of a type which supports context-accessible virtual devices, that is, does not need context-sensitive processing, it is passed directly to the Associated Address agent.



Attention: Enabling a context-sensitive discovery automatically enables all the Context agents. Disabling a context-sensitive discovery automatically disables all the Context agents. Do not manually enable or disable Context agents, either through the configuration files or through the Discovery Configuration GUI.

To enable a context-sensitive discovery, append the following insert to the `DiscoConfig.cfg` file:

```
insert into disco.config
(
      m_UseContext
)
values
(
      1
)
```

Inserting the value `0` disables the context-sensitive discovery.

Configuring EMS discoveries

You can configure Network Manager to collect topology data from Element Management Systems (EMS) and integrate this data into the discovered topology.

About this task

The following topics describe how to configure an EMS discovery.

For an overview of how Network Manager collects topology data from Element Management Systems (EMSs) and integrates this data into the discovered topology, see the *IBM Tivoli Network Manager User Guide*.

About EMS integration

The Network Manager EMS integration enables Network Manager to collect topology data from Element Management Systems.

About collectors

A collector is a software module that retrieves topology data from a data source, such as an Element Management System (EMS) or a comma-separated value (CSV) file, and makes this data available to the discovery process as a set of XML data. Network Manager can then stitch this data into the discovered topology.

A collector translates the topology data from the format in which it is held in the proprietary EMS into a standard XML structure that can be processed by Network Manager. This means that a different collector must be developed for each different EMS vendor and model. The predefined collectors provided with Network Manager are written in either Java or Perl. However, collectors may be written in any language, as long as that language can provide an XML-RPC server which the **ncp_disco** process can query.

Network Manager ships with Java and Perl modules to support the development of collectors in those languages.

Collectors can run on the same host as the Network Manager. Collectors can also run on a separate host.

All interaction between Network Manager and the collectors is conducted using XML, and this interaction occurs over an XML-RPC interface.

Predefined collectors

Network Manager ships with a set of default collectors that retrieve device data from a variety of EMSs, including EMSs that manage radio access network (RAN) and Optical devices.

About this task

Scripts and a plain-text configuration file for each collector are held in separate directories named after the collector, within the `$NCHOME/precision/collectors/` directory, as follows:

- Collectors written in Java are stored in the `$NCHOME/precision/collectors/javaCollectors/` directory.
- Collectors written in Perl are stored in the `$NCHOME/precision/collectors/perlCollectors/` directory.

For example, the CiscoLMS collector is stored in the directory `javaCollectors/CiscoLMS/`, and the AlcatelNR8PLIooIsn collector is stored in the `perlCollectors/ AlcatelNR8PLIooIsn/` directory.

Each collector supplied with Network Manager downloads data from an EMS using a Northbound Interface (NBI) protocol or uses data downloaded from the EMS by the user. Each EMS manages devices that support certain technologies.

The default collectors are listed in the tables that follow:

Table 43. Default Java collectors

Collector	Description	NBI Protocols	Technologies
Alcatel5620Sam	Collector for the Alcatel 5620 SAM EMS. This Java collector retrieves the same information as the Perl Alcatel5620SamSoapFindToFile collector, and additionally retrieves Link Aggregation Group (LAG) information. The collector retrieves LAG containment data, which is modelled and presented in the Structure Browser. The collector retrieves VLAN interface information. The collector retrieves address and subnet information for ENodeB elements, creates VLAN information based on that information, and mdoels Ethernet ports and control boards. Starting from V4.2 FixPack 1, LTE-related discovery data for the SAM 5620 EMS is no longer supported for Network Manager partial discovery. Fix Pack 6 The collector retrieves information for cloud-based, or virtual MMEs (Multimedia Extensions). Fix Pack 6 The collector retrieves information for Nokia SAE gateways, which have LTE PDNGateway and ServingGateway functionality.	SOAP	Long Term Evolution (LTE), LAG, OSI Layer 2, OSI Layer 3, Interface, Inventory, Physical Entity, VPN Layer 3 and VPN Layer 2
Cisco APIC REST	Collector for the Cisco Application Policy Infrastructure Controller (APIC) EMS.	JSON and XML over REST WebSocket protocol	Interface Inventory, Physical Entity
CiscoLMS	Collector for the CiscoWorks LMS EMS. This collector uses the Cisco Open Database Schema and Cisco Data Extraction Engine to communicate with the EMS.	JDBC and REST Web services	Physical Entity, Interface Inventory
csv	Generic CSV-based collector that reads `.csv` files for input data. There is a GenericCsv collector written in Java, and a second GenericCsv collector written in Perl.	N/A	Various

Table 43. Default Java collectors (continued)

Collector	Description	NBI Protocols	Technologies
Huawei CORBA TMF 814	Collector for the Huawei CORBA TMF 814 EMS. This collector collects information from EMS systems that use the CORBA TMF 814 interface, such as Huawei T2000 and U2000. The collector retrieves SNC data to supplement CTP data. Visualization of SNC data is not supported.	CORBA TMF814	OSI Layer 1, Interface Inventory, Physical Entity, SONET, SDH
Huawei M2000	Collector for the Huawei M2000 EMS. This collector processes IMS and STP devices, and LTE 3G, 2G, PSCore, and CSCore data by using the configuration management XML files generated from a Huawei M2000 EMS. Place the XML files in a local directory for access by the collector. The Huawei M2000 EMS can produce many different kinds of XML files. This collector specifically supports the following XML files: • LNBI_XML_RT_*.xml • SRANNBIExport_ XML_LTE_RT_*.xml • CMExport_*.xml • GNBIExport_XML_RT_*.xml • UNBIExport_XML_RT_*.xml The collector also processes any available IM*.xml, AIM_*.xml and W_OMC_*.xml files to supplement the data from the XML files above.	N/A	eNodeB, BSC, BTS, NodeB, RNC, MGW, UGW, USN, HIR, EIR, HSS, CGNE, MSS, ATS, CCF, CSCF, MRFP, SE2900, SPG, SG7000, USCDB
MTOSISoap	Generic collector that discovers Element Management Systems that support the MTOSI Soap NBI, for example, the Huawei U2000 iManager EMS. Supported technologies: Layer 2 discovery, Layer 3 discovery, and Physical Inventory discovery.	MTOSI SOAP	Physical Inventory, Layer 2, Layer 3

Table 43. Default Java collectors (continued)

Collector	Description	NBI Protocols	Technologies
NetActCMDump	Processes 2G, 3G and LTE RAN data by using the configuration management XML file for the NetAct EMS collector. The collector needs to connect to the NetAct EMS collector in order to retrieve this XML file, and it does this using either FTP or SFTP. In order to use data from the NetAct XML Interface for Configuration Management collector in a network discovery, you must configure the collector properties file with the FTP and SFTP connection details between the NetAct EMS and Network Manager.	FTP or SFTP	LTE, RAN, SGSN, eNodeB, MME, PGW, BTS, RNC, SGW, BSC, NodeB, MGW, TRX, MSC, PCU
NetViewer	Collector for the Nokia Solutions and Networks (NSN) NetViewer EMS.	CORBA TMF814	OSI Layer 1, Interface Inventory, Physical Entity, SONET, SDH
Nokia5529Idm	Collector for the Nokia5529Idm EMS. This Java collector retrieves the same information as the Perl Nokia5529Idm collector, and it additionally supports the Bulk Network Export feature and supports the Java Message Service (JMS). The collector listens for JMS notifications about entities that have been added or removed from the EMS. The collector pauses and then rediscovers the EMS. During the next discovery or partial discovery, the NCIM topology database is updated with the latest data from the collector.	SOAP	Interface Inventory, Physical Entity
NokiaOMS1350	Collector for the Nokia OMS 1350 EMS. This collector discovers Nokia 1830 PSS-based optical network devices.	REST	OSI Layer 1
Tellabs INM8000	Collector for the Tellabs INM8000 EMS.	Java API	OSI Layer 3, OSI Layer 2, Interface Inventory, Physical Entity

Table 44. Default Perl collectors

Collector	Description	NBI Protocols	Technologies
Nokia5529Idm	Collector for the Nokia5529Idm EMS.	SOAP	Interface Inventory, Physical Entity

Table 44. Default Perl collectors (continued)

Collector	Description	NBI Protocols	Technologies
Alcatel5620SamSoap	Collector for the Alcatel 5620 SAM EMS.	SOAP	OSI Layer 2, OSI Layer 3, Interface Inventory, Physical Entity, VPN Layer 3, VPN Layer 2
Alcatel5620SamSoap FindToFile	Collector for the Alcatel 5620 SAM EMS. The collector retrieves the same data as the Alcatel5620SamSoap FindToFile collector. The collector stores the data from the EMS in XML files with the same name as the objects queried. The collector transfers the XML files to the Network Manager using FTP. You must configure the FTP connection details before running the collector.	SOAP, FTP	LTE, OSI Layer 2, OSI Layer 3, Interface Inventory, PCRF, SGW, PGW, MME, eNodeB, Physical Entity, VPN Layer 3, VPN Layer 2
Alcatel5620SamCsv	Collector for the Alcatel 5620 SAM EMS. This collector retrieves EMS topology data from a CSV dump of the Alcatel 5620 SAM EMS.	N/A	Interface Inventory, Physical Entity
AlcatelNR8PLIooIsn	Collector for the Alcatel Lucent 1353 NM and Alcatel Lucent 1354 RM components within the AlcatelNR8PL EMS.	IOO (for the Alcatel Lucent 1353 component), ISN (for the Alcatel Lucent 1354 RM component)	1353 NM, 1354 RM
GenericCsv collector	Generic CSV-based collector that reads `.csv` files for input data. There is a GenericCsv collector written in Java, and a second GenericCsv collector written in Perl.	N/A	Inventory
HuaweiU2000ImanagerTL1	Collector for the HuaweiU2000Imanager EMS.	TL1	Physical Entity, Interface Inventory
HuaweiU2000iManager TL1DumpExport	Collector for the HuaweiU2000Imanager EMS. This collector uses XML files from the EMS.	N/A	Physical Entity, Interface Inventory

Table 44. Default Perl collectors (continued)			
Collector	**Description**	**NBI Protocols**	**Technologies**
OpticalBlackboxXml	Blackbox collector that enables you to add passive layer 1 entities to the discovered network topology.	N/A	Physical Entity, Interface, Layer 1

Other components of the EMS integration

In addition to collectors, EMS integration is composed of several components that assist in the collection of topology data.

The components of the EMS integration are described in Table 45 on page 251.

Table 45. Components of EMS integration	
Component	**Description**
Collector finder `ncp_df_collector`	The Collector finder reads the collector host seeds from a seed table in the `collectorFinder` database. It then queries the collectors specified in this table to get a list of devices managed by the EMS associated with each collector.
Collector agents	Retrieve basic and detailed information about the devices on the collector. Each agent makes use of the Collector helper to retrieve this information.
CollectorDetails agent	Retrieves basic information about the devices on the collector, including sysObjectId, sysDescr, and naming data.
CollectorInventory agent	Retrieves local neighbor, entity and associated address data for each of the devices on the collector.
CollectorLayer1 agent	Retrieves layer 1 and microwave connectivity information for the devices on the collector.
CollectorLayer2 agent	Retrieves layer 2 connectivity information for the devices on the collector.
CollectorLayer3 agent	Retrieves layer 3 connectivity information for the devices on the collector.
CollectorLTE agent	Retrieves LTE-specific entity information for the devices on the collector.
CollectorRan agent	Retrieves radio access network (RAN) information for the devices on the collector.
CollectorVpn agent	Retrieves layer 2 and layer 3 VPN data for the devices on the collector.
Collector helper `ncp_dh_xmlrpc`	Enables Network Manager to communicate with the collectors using the XML-RPC interface.

EMS discovery data flow

Use this information to understand the steps in an EMS discovery.

Standard EMS discovery data flow

Use this information to understand how Network Manager collects topology data from an EMS as part of a discovery or partial discovery.

The table below shows the steps involved in collecting topology data from EMS as part of a discovery or partial discovery. After this data has been collected, Network Manager stitches it together with the topology.

Table 46. Collecting topology data from EMS during discovery

Step	Data Flow
1	Using the Collector finder, the discovery system queries the collector to obtain a list of devices managed by the EMS. In the case of a partial discovery, the discovery might query for a single device or subnet only.
2	The Collector queries the EMS for the list of devices.
3	The EMS responds with list of managed devices.
4	The Collector responds by providing the list of devices.
5	Using a number of specialized collector discovery agents at different times during the discovery, the discovery system queries the collector for basic and detailed information about each of the devices in the list. Detailed information requested includes the following: • Inventory information. • Layer 1 and microwave connection details. • Layer 2 connection details. • Layer 3 connection details. • LTE information. • Radio access network (RAN) information. • VPN information.
6	The Collector responds by providing basic and detailed information as this is requested.

Synchronization for multiple data sources

If you run a discovery that includes collector and SNMP data sources, or multiple collectors, you must synchronize the "Interrogating Devices" phase (phase 1) of the discovery across all of the data sources. Synchronization of phase 1 reduces the likelihood of excessive discovery cycling and ensures that the maximum amount of data is available during topology building. Synchronization capability is provided by the `m_WaitForManagedProcs` configuration setting. `m_WaitForManagedProcs` is set to 0 (off) by default, which is more suited to SNMP-only discoveries.

During phase 1 of an IP discovery, discovery waits by default up to 90 seconds between the discovery of one device and the next. If no further devices are found after 90 seconds, then the discovery moves into phase 2. This time delay is controlled by the `m_NothngFndPeriod` field in the `disco.config` discovery database table, and is ideal for SNMP discoveries in which entities do not all respond at the same time.

However, the behavior of collectors is different than SNMP. Collectors return all of their devices in one go, potentially after a comparatively long initial delay, during which the EMS is queried. Using only `m_NothngFndPeriod` when multiple collectors or collector and SNMP discoveries are run under the same domain can lead to undesired multiple discovery cycles. This effect happens because the interrogation phase delay between each collector can be far in excess of 90 seconds. This delay causes the discovery to continue to the next phase if the SNMP ping phase completed.

To ensure that data from all collectors in a multiple data source discovery is collected and stitched together in the same discovery cycle, you must configure the following fields in the `disco.config` discovery database table:

`m_WaitForManagedProcs`
Set this field to 1. This setting ensures that the discovery process remains in phase 1 until all the collectors finish processing data on their respective EMSs.

By default this field is set to 0. This setting means that when the first collector completes data processing and the number of seconds defined in the `m_NothngFndPeriod` field pass, then the discovery process moves to phase 2 without waiting for the other collectors to start processing.

`m_ManagedWaitTimeOut`
Applicable when `m_WaitForManagedProcs` is set to 1. This field defines the maximum time to wait for all collectors to complete retrieving data from their EMSs. This setting is effectively a fail-safe mechanism to cover the situation where one of the collectors never completes processing. Set this setting to the maximum value in seconds to wait for all collectors to complete data processing. When this timeout elapses, and the number of seconds defined in the `m_NothngFndPeriod` field pass, then the discovery process moves to phase 2.

By default, this value is set to 0, which means wait indefinitely.

Configuring an EMS discovery

Configure an EMS discovery to collect topology data from Element Management Systems and integrate this data into the discovered topology.

About this task

You configure an EMS discovery in the same way that you configure the discovery of any other type of network. In addition to the standard discovery configuration activities you must perform some EMS-specific discovery configuration activities.

To configure an EMS discovery, do the following activities in addition to standard discovery configuration activities:

- Configure and start the EMS collectors
- Seed the EMS discovery by seeding the Collector finder

Note: Collector agents are enabled by default, so they will automatically run when you configure the collector-based discovery.

These EMS-specific discovery configuration activities are described in the following topics.

Configuring collectors

The purpose of a collector is to gather and standardise EMS data based on requests by the main Network Manager discovery process, for consumption by the discovery process. Collector configuration governs how the collector interrogates the EMS or data source in order to service requests from the Network Manager discovery process, and how the collector listens for requests from the Network Manager discovery process.

About this task

Collector configuration is made up of three areas:

- Collector to EMS or data source configuration: these settings define how the collector interrogates the EMS or data source in order to service requests from the Network Manager discovery process.
- Collector to Network Manager configuration: these settings define how the collector listens for requests from the Network Manager discovery process.
- Miscellaneous configuration: these settings are optional and include settings such as debug mode.

These configuration settings are typically held in a single file on a per-collector basis inside the relevant collector directory.

Scripts and a plain-text configuration file for each collector are held in a separate directories within the $NCHOME/precision/collectors/ directory, as follows:

- Collectors written in Java are stored in the $NCHOME/precision/collectors/javaCollectors/ directory.
- Collectors written in Perl are stored in the $NCHOME/precision/collectors/perlCollectors/ directory.

Note: To allow the collectors to be run in isolation, that is, on another machine in a Network Manager installation, the collector directory can be moved to that machine and the collectors can be run there provided that the following hold:

- For the Perl collector directory, Perl must be installed on the target machine.
- For the Java collector directory, a suitable Java Virtual Machine must be available on the target machine.

Experienced users can develop new collectors to enable Network Manager to interact with other EMSs. Configuration and executable files for each new collector must be placed in an appropriately named directory within one of the directories listed above, depending on whether the collector is written in Java or Perl. For more information, see the *EMS Collector Developer Guide*.

Configuring Java collectors
You can configure generic settings for all Java collectors. You can also configure specific settings for each Java collector.

Configuring generic settings for Java collectors
You can configure generic settings for all Java collectors in the generic collector.properties file .

About this task

You can configure generic settings for all Java collectors by editing the collector.properties file, located at: $NCHOME/precision/collectors/javaCollectors/framework/. The file uses standard Java key value pair notation, that is, key = value. Using this file you can configure the following parameters:

- Port on which to run the Java collectors.
- Logging details for the Java collectors.

Sample collector.properties file

The following code fragment presents sample settings from a collector.properties file.

```
# Port on which to run the embedded collector server
port = 8080

# Log directory relative to the bin directory
log.directory = ../log/

# Name of the collector log file
log.filename = collector.log

# Level of logging for the collector framework
log.level = INFO

# Name of the collector log file
trace.filename = collector-trace.log

# Level of logging for the collector framework
trace.level = FINEST
```

Generic properties file reference for Java collectors
Use this information to understand how the generic properties file for Java collectors is constructed.

The generic properties file for Java collectors contains the properties listed in the following table.

Table 47. Generic properties file contents	
Property	**Description**
`port`	Port on which to run the collector.
`log.directory`	Base directory where log files should be stored.
`log.filename`	Filename for the log file. Can also specify a pattern for the log file name using a set of system-defined elements.
`log.level`	One of the following: • NONE • FINEST • FINER • FINE • CONFIG • INFO • WARNING • SEVERE • ALL
`log.maxsize`	Maximum size of the log file. Once this maximum is reached, the file is renamed and is kept as a backup.
`log.count`	Number of backup log files to keep.
`log.messageprefix`	Log message prefix.
`trace.filename`	Filename for the trace file. Can also specify a pattern for the trace file name using a set of system-defined elements.
`trace.level`	One of the following: • NONE • FINEST • FINER • FINE • CONFIG • INFO • WARNING • SEVERE • ALL
`trace.maxsize`	Maximum size of the trace file. Once this maximum is reached, the file is renamed and is kept as a backup.
`trace.count`	Number of backup trace files to keep.

Configuring individual Java collectors
You can configure specific settings for each Java collector by editing the `.properties` file for that collector.

Configuring the Nokia5529Idm Java collector
This collector uses the Bulk Network Export feature to retrieve data from the Alcatel Lucent 5529 IDM EMS. Before running the Nokia5529Idm Java collector in a network discovery, you must copy certain required files and configure the connection details between the EMS and Network Manager.

About this task

You can also configure additional information to be collected from the EMS.

Restriction: The Nokia5529Idm Java collector is only supported on the 5529 IDM EMS running on releases 9.4, 9.6.07, and 9.6.08.

To configure the collector, complete the following steps:

Procedure

1. Before running the collector, copy the required .jar files from the EMS to the server on which the collector is installed:
   a) Log into the Schema page of the IDM using the credentials of an IDM NBI user.
      The URL of the IDM Schema page is https://*hostname*:8443/idm/schemadoc/html/index.html, where *hostname* is the name or IP address of the IDM server.
   b) Click on the Client sample URL and download the `idm-oss-client.tar.gz` file to the server on which the collector is installed.
   c) Uncompress the file.
   d) If the EMS version is 9.4 or below, then copy the following files to the `$NCHOME/precision/collectors/javaCollectors/lib` directory:

```
axs-mobject-remote-api-9.4-268573.jar
hornetq-core-client-2.3.1.Final-ALU-1
hornetq-jms-client-2.3.1.Final-ALU-1.jar
jaxen-1.1.jar
jboss-as-security-7.2.0.Final-ALU-8.jar
jboss-client.jar
jboss-logging-3.1.2.GA-ALU-1.jar
jboss-remoting-3.2.17.GA.jar
jdom-1.1.2-ALU-2.jar
jms-1.1.jar
log4j-1.2.13.jar
netty-3.6.2.Final-ALU-1.jar
trove-2.1.1.jar
```

   e) If the EMS version is 9.6.07 or 9.6.08, then copy the following files to the `$NCHOME/precision/collectors/javaCollectors/lib` directory:

```
axs-encription-app-9.6.07-399857.jar
jboss-logging-3.3.0.Final.jar
slf4j-simple-1.7.21.jar
axs-mobject-api-9.6.07-399857.jar
log4j-1.2.14.jar
wildfly-client-all.jar
axs-mobject-remote-api-9.6.07-399857.jar
picketbox-4.9.6.Final.jar
xbean-2.6.0.jar
idm-oss-client-1_9.6.07-399857.jar
picketbox-infinispan-4.9.6.Final.jar
```

2. Change to the collector directory:

```
cd $NCHOME/precision/collectors/javaCollectors/Alcatel5529Idm/
```

3. Within this directory, find the sample configuration file for the Nokia5529Idm Java collector and copy it to the working configuration file using a command similar to the following example:

```
cp Alcatel5529IdmCollector.properties.sample Alcatel5529IdmCollector.properties
```

4. The configuration file consists of the following sections:

   Collector properties
   General configuration parameters for the collector, such as port number and log and trace details.

   Data Source properties
   Details of the EMS that the collector is connecting to. Data from these fields is used by Network Manager to model the EMS.

   Data Acquisition properties
   Parameters that specify which data to collect from the EMS.

   SOAP properties
   SOAP-specific parameters.

   FTP properties
   FTP-specific parameters.

   JMS properties
   Properties relating to the Java Messaging System.

   Fix Pack 9 **JMS HTTPS properties**
   Properties relating to the JMS HTTPS configuration.

   CSV properties
   Properties relating to comma-separated value format files.

5. Configure the collector port and log and trace parameters:

   port
   The port on which to run the collector. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8080.

   log.filename
   Filename for the collector log file. You can also specify a pattern for the log file name using a set of system-defined elements. The default value is Alcatel5529IdmCollector.log.

   log.level
   Takes one of the following values:

   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO
   - WARNING
   - SEVERE
   - ALL

   trace.filename
   Filename for the collector trace file. You can also specify a pattern for the trace file name using a set of system-defined elements. The default value is Alcatel5529Collector-trace.log.

   trace.level
   Takes one of the following values:

   - NONE
   - FINEST
   - FINER

- FINE
- CONFIGINFO
- WARNING
- SEVERE
- ALL

6. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

   DataSource.id
   Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

   DataSource.descr
   Description of the EMS.

   DataSource.emsName
   Name of the EMS.

   DataSource.emsPort
   Port of the EMS.

   DataSource.emsVersion
   Version of the EMS.

   DataSource.emsIdentifier
   Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

   DataSource.emsRole
   Role of the EMS. This parameter can take one of the following values:

   - unknown
   - primary
   - backup
   - other

   DataSource.emsStatus
   Status of the EMS. This parameter can take one of the following values:

   - unknown
   - up
   - down
   - other

7. In the Data Acquisition properties section, configure data acquisition parameters:

   collectData
   The default value is `true`.

   True
   Enables the collector. The collector collects data from the EMS.

   False
   Disables the collector. The collector does not collect data from the EMS.

   DataAcquisition.fullDiscoTimeout
   The number of seconds that the collector waits for the Bulk Network Export operation to be completed by the EMS. The default value is 30.

8. In the SOAP properties section, configure the SOAP properties:

   soap.username
   Username for the SOAP service.

soap.password
Password for the SOAP service.

soap.port
Port for the SOAP service.

soap.secure
This property can be `true` or `false`. The default value is false. If it is set to true, a secure HTTPS connection is used to connect to the EMS. If it is false, the collector uses an HTTP connection to the EMS.

jsse.httpsImpl
The HTTPS implementation class name. The default class name is `com.ibm.net.ssl.www2.protocol`. This property takes effect only when the `soap.secure` property is set to `true`.

jsse.trustStore
Complete path to the trust-store file. This property takes effect only when the `soap.secure` property is set to `true`.

jsse.trustPass
Password for the trust-store file. This property takes effect only when the `soap.secure` property is set to `true`.

Fix Pack 9 **TLSVersion**
The protocol version of the SSL TLS protocol. The default version is `TLSv1.1`. If you specify the value `TLSv1.2` the collector uses TLS v1.2 instead during the SSL handshake.

SSLDebugInfo
If `true`, the SSL handshake trace info is logged during the SSL handshake. If `false`, the debug information is not printed. The default is `false`.

9. In the FTP properties section, configure the FTP properties.

ftp.host
The IP address or hostname of the server on which the collector is running.

ftp.username
The FTP session username. This user must be able to log in to the server and have write permissions for the FTP directory.

ftp.password
The password for the username specified for the FTP session.

ftp.directory
The full path to the destination directory for files transferred from the EMS. This directory is on the server on which the collector is running. The default directory is `/tmp`.

ftp.compressed
The default is `false`.

True
Files are transferred in a compressed archive.

False
Files are not compressed.

10. In the JMS properties section, configure the properties that relate to the Java Messaging Service:

jms.username
Username for the JMS service.

jms.password
Password for the JMS service.

jms.topic
The Topic name to listen for JMS messages.

jms.filter
Message filter to process specific messages. Leave blank for no filtering.

jms.durable
Set as `true` to enable durable subscription. Default is `false`.

jms.unsubscribe
Set as `true` to unsubscribe from the Topic when durable subscription is enabled. Default is `false`.

jms.clientId
JMS client identifier to be set for durable subscriptions.

jndi.properties.file
Complete path of the `jndi.properties` file that contains all the JNDI related properties. If this property is not found or not configured, or the file is incorrect, then the collector will use the default JNDI configuration. If you configure this property, set the following properties in the file:

java.naming.factory.initial
Specifies the initial context factory to use. The value of the property must be the fully qualified class name of the factory class that will create an initial context. The default value is `org.jboss.naming.remote.client.InitialContextFactory`.

java.naming.provider.url
Specifies the location of the JBoss JNDI service provider that the client will use. The NamingContextFactory class uses this information to know which JBossNS server to connect to. The value of the property must be a URL string. The format of this value is `remote://`*EMS host*`:`*port*. The default value is `remote://127.0.0.1:4447`. Change this value to the EMS host and port.

java.naming.factory.url.pkgs
Specifies the list of package prefixes to use when loading in URL context factories. The value of the property must be a colon-separated list of package prefixes for the class name of the factory class that will create a URL context factory. Default is `org.jboss.ejb.client.naming`.

java.naming.security.principal
The principal to authenticate. This must be a string representing the name of a principal. The default is `admin`.

java.naming.security.credentials
The credentials to authenticate the principal, for example, the password. The default is `admin`.

jnp.maxRetries
An integer that controls the number of connection retry attempts that will be made on the initial connection to the naming server. This parameter only applies to ConnectException failures. A value less than or equal to 1 means that only one attempt will be made. The default value is 5.

jnp.timeout
The connection timeout in milliseconds. The default value is `30000`. If this value is set to `0`(zero), the connection will block until the VM TCP/IP layer times out.

jnp.partitionName
This property is applicable only for a Cluster JNDI setup. If you are not using a Cluster JNDI setup, remove this property. Specifies the cluster partition name that discovery is to be restricted to. If you are running the product in an environment with multiple clusters, you might want to restrict the naming discovery to a particular cluster. There is no default value, which means that any cluster response will be accepted.

jnp.discoveryGroup
This property is applicable only for a Cluster JNDI setup. If you are not using a Cluster JNDI setup, remove this property. The multicast IP address to which the discovery query is sent. The default is `228.1.2.5`.

jboss.naming.client.connect.options.org.xnio.Options.SSL_STARTTLS

Note: This property is applicable only to AMS/IDM version 9.6 or above. If you are not using AMS/IDM version 9.6 or above, remove this property from the file.

Boolean parameter.

jboss.naming.client.connect.options.org.xnio.Options.SASL_POLICY_NOPLAINTEXT

Note: This property is applicable only to AMS/IDM version 9.6 or above. If you are not using AMS/IDM version 9.6 or above, remove this property from the file.

Boolean parameter.

jboss.naming.client.remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED

Note: This property is applicable only to AMS/IDM version 9.6 or above. If you are not using AMS/IDM version 9.6 or above, remove this property from the file.

Boolean parameter.

jboss.naming.client.connect.options.org.xnio.Options.SSL_PROTOCOL=

Note: This property is applicable only to AMS/IDM version 9.6 or above. If you are not using AMS/IDM version 9.6 or above, remove this property from the file.

Specifies the protocol to be used. Allowed values are `TLSv1.2`.

11. Configure the CSV Properties section.

    The data from the EMS is downloaded as a file in the Comma-Separated Values (CSV) format. You can configure which values from the EMS for the NE and NE System are exported to the CSV file. The first three values are always mapped to the NE name (or NE System name), managed object type, and object identifier. Because the exact order of the remaining values in the CSV file can vary in different systems, you must configure the mapping so that the values are mapped correctly by the collector.

    Restriction: The values that you use for the following variables must match exactly the attribute names exported by the EMS.

    csv.NE.1

    Specifies the mapping for the first unmapped variable for a line in the CSV file that describes an NE, that is, for the fourth variable in the line. For example:

    ```
    csv.NE.1=IP Address
    ```

    csv.NE.n

    Specifies the mapping for the next unmapped variable, where n is an integer that increases by 1 with each subsequent variable.

    csv.NE_System.1

    Specifies the mapping for the first unmapped variable for a line in the CSV file that describes an NE System, that is, for the fourth variable in the line. For example:

    ```
    csv.NE_System.1=Contact
    ```

    csv.NE_System.n

    Specifies the mapping for the next unmapped variable, where n is an integer that increases by 1 with each subsequent variable.

    In the following example data for an NE, the NE name is ISAM123, the managed object type is NE, and the object identifier is ISAM123.

    ```
    ISAM123,NE,ISAM123,192.168.242.175
    ```

    The value 192.168.242.175 is the IP address. So you must define the following property:

    ```
    csv.NE.1=IP Address
    ```

In the following example data for an NE System, the NE System name is ISAM123, the managed object type is NE System, and the object identifier is ISAM123.

```
ISAM123,NE System,ISAM123,Mike,Main UK ISAM system,R4.5.02,,00:80:C0:52:D4:8F,
"Administrators
 office"/,3FE478262BNAA_E3.2.0.9,EX1234,LEUK,
10/9/14 3:11:26 AM
```

In order to correctly map the remaining values, you would define the following properties:

```
csv.NE_System.1=Contact
csv.NE_System.2=Description
csv.NE_System.3=Ethernet DSL
csv.NE_System.4=ETSI Version
csv.NE_System.5=HUB MAC Address
csv.NE_System.6=Location
csv.NE_System.7=MIB Version
csv.NE_System.8=System ID
csv.NE_System.9=Type
csv.NE_System.10=Up Time
```

12. In the JMS HTTPS properties section, configure the properties that relate to the secure Java Messaging Service:

    jms.secure
    If `true`, the HTTPS secure JMS connection is established. If `false`, the connection is insecure. The default is `true`.

    jms.keystore
    The complete path to the trust-store file. This property takes effect only when the `soap.secure` property is set to `true`.

    jms.keypass
    Password for the trust-store file. This property takes effect only when the `soap.secure` property is set to `true`.

    jndi.isClusterSetup
    Set to `true` if the JNDI server is running in cluster mode. Set to `false` if the JNDI server is not running in cluster mode. If the EMS version is 9.6 or above, set the value to `false`. The default is `true`.

13. Save the collector configuration file.

Configuring the Alcatel5620Sam Java collector
To use data from the Alcatel5620Sam Java collector in a network discovery, you must copy certain files from the EMS to the server where the collector is installed, and configure the connection details between the EMS and Network Manager.

About this task

You can also configure additional information to be collected from the EMS. To configure the Alcatel5620Sam Java collector, complete the following steps:

Procedure

1. Before running the collector for the first time, copy the file `/opt/5620sam/server/nms/integration/samOss.jar` from the EMS server to the `/opt/IBM/netcool/core/precision/collectors/javaCollectors/lib/` directory. You must use the `samOss.jar` file from the same version and release as the Aclatel 5620 system.
2. Install the Oracle JRE on the Alcatel 5620 server.
3. Create a new script to run the collector.
   a) Create a copy of the `/opt/IBM/tivoli/netcool/precision/collectors/javaCollectors/bin/collector.sh` script in the same directory.
   This script is used to start only the SAM 5620 collector.

For example, name the script `collectorSAM5620Java.sh`

b) Edit the new script and change the value for `COLL_JAVA` to point to the newly installed JRE. For example, change `COLL_JAVA=$JAVA` to `COLL_JAVA=/usr/local/java/jre1.7.0_72/bin/java`

4. Change to the directory that contains the collector files.

```
cd $NCHOME/precision/collectors/javaCollectors/Alcatel5620Sam/
```

5. Within this directory, find the sample configuration file for the collector and copy it to the working configuration file.

```
cp Alcatel5620Sam.
properties.sample
Alcatel5620Sam.
properties
```

6. Edit the collector configuration file:

```
$NCHOME/precision/collectors/javaCollectors/Alcatel5620Sam/Alcatel5620Sam.
properties.
```

This file includes the following configuration sections:

- Collector properties. General configuration parameters for the collector, such as port number and log and trace details.
- EMS connection properties. Details of the EMS that the collector is connecting to. Data from these fields is used by Network Manager to model the EMS.
- HTTPS properties. Properties for configuring the HTTPS communication with the SAM 5620 EMS.
- FTP properties. Defines the file on the Network Manager server to which the SAM 5620 EMS writes the generated SOAP XML response. You must ensure that the FTP directory has public read/write permissions.
- Data Acquisition properties. Parameters that specify what data to collect from the EMS.

The following steps list the configurable parameters. All other properties in this file are collector system-based configurations and must not be changed.

7. Configure the collector port and log and trace parameters:

port
The port on which to run the collector. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file.

log.filename
Filename for the collector log file. You can also specify a pattern for the log file name using a set of system-defined elements.

log.level
Takes one of the following values:

- NONE
- FINEST
- FINER
- FINE
- CONFIGINFO
- WARNING
- SEVERE
- ALL

log.maxsize

Maximum size of the collector log file in MB. The default is 50.

log.maxcount

Maximum number of generated collector log files. The default is 100.

trace.maxsize

Maximum size of the collector trace file in MB. The default is 50.

log.maxcount

Maximum number of generated collector trace files in MB. The default is 100.

trace.filename
Filename for the collector trace file. You can also specify a pattern for the trace file name using a set of system-defined elements.

trace.level
Takes one of the following values:

- NONE
- FINEST
- FINER
- FINE
- CONFIGINFO
- WARNING
- SEVERE
- ALL

8. In the EMS Connection Configuration section, configure these parameters:

EmsHost
SAM 5620 host IP address.

EmsPort
SAM 5620 port accepting SOAP requests.

Username
Username of a SAM 5620 OSS user with OSS Management privileges.

Password
The SAM 5620 password, in plain text. This property only takes effect if the `md5Password` property is not used.

Md5Password
The SAM 5620 password as an MD5 hash. If both `Password` and `Md5Password` are specified, `Md5Password` is used.

Secure
If set to `true`, the HTTPS protocol is used to communicate with the EMS.

9. In the FTP Configurations section, configure the following parameters:

IsRemoteFTPServer
Indicates whether the FTP is Remote FTP server. To use remote FTP server, set the value to true. Default is false. If this value is set to true, the EMS inventory files are first copied to Remote FTP server and then downloaded to local ITNM server for processing.

FtpUsername
The FTP username to use to connect to the EMS.

FtpPassword
The FTP password to use to connect to the EMS.

FtpHost
The IP address of the Network Manager host.

LocalFtpDirectory
The absolute path to the local directory where the file is saved.

SecureFTP
If true, SFTP is used.

RemoteFTPDirectory
The absolute path in the Remote FTP server, to which EMS transfers the inventory files.

Note: Valid only when FTP server is not same as ITNM server and configured with remote server. For example, `isRemoteFTPServer` is set to `true`.

RemoteFTPTimeout
The timeout interval for the FTP process (in seconds). Valid only when `isRemoteFTPServer` is set to `true`.

10. In the HTTPS Configuration section, configure the following parameters:

HTTPSSecure
If `true`, HTTPS (instead of HTTP) is used to connect to the EMS.

TrustStore
The full path of SAM 5620 truststore file.

TrustStorePassword
The TrustStore Password.

HTTPSPort
HTTPS port for the SAM 5620. The default is 8443.

HTTPSProtocolHandler
Protocol handler for HTTPS connection. Because the SAM 5620 does not support the IBM HTTPS protocol handler package, the collector uses Oracle's HTTPS handler package by default.

Fix Pack 8 **TLSVersion**
The protocol version of the SSL TLS protocol. The default version is `TLSv1.1`. If you specify the value `TLSv1.2` the collector uses TLS v1.2 instead during the SSL handshake.

Fix Pack 8 **SSLDebugInfo**
If `true`, the SSL debug info is printed during the SSL handshake. If `false` , the debug information is not printed. The default is `false`.

11. In the Data Acquisition configuration section, configure the following parameters:

GetEntities

1: Enable
Enables download of physical entity data from the EMS.

0: Disable
Disables download of physical entity data from the EMS.

GetVplsVpns

1: Enable
Enables download of VPLS VPN data from the EMS.

0: Disable
Disables download of VPLS VPN data from the EMS.

GetVllVpns

1: Enable
Enables download of VLL VPN data from the EMS.

0: Disable
Disables download of VLL VPN data from the EMS.

GetLayer3Vpns

1: Enable
Enables download of Layer 3 VPN data from the EMS.

0: Disable
Disables download of Layer 3 VPN data from the EMS.

GetMplsInterfaces

1: Enable
Enables download of MPLS interface data from the EMS.

0: Disable
Disables download of MPLS interface data from the EMS.

GetLayer2Connections

1: Enable
Enables download of Layer 2 connectivity data from the EMS.

0: Disable
Disables download of Layer 2 connectivity data from the EMS.

GetLteData

1: Enable
Enables download of LTE data from the EMS.

0: Disable
Disables download of LTE data from the EMS.

Fix Pack 8 **loadFirstRun**
If `true`, the collector queries the EMS when the collector starts up. If `false`, the collector queries the EMS when you start the network discovery.

Fix Pack 8 **EntityExtendedNameRequired**

If `true`, the `ifName` and `entityDescription` for Physical Interfaces contain extra information, as shown in the following examples.

```
ifName: (displayedName att value or PortName) - Mode (mode att value)
 : - Speed (speed att value) : -Encap (encapType att value) : - State
(compositeEquipmentState att value)
```

```
EntityDescription : displayedName ( bit att value) ,
(compositeEquipmentState att value) , (redundantStatus att value)
```

If `false`, the `ifName` and `entityDescription` for Physical Interfaces do not contain extra information. The default is `false`.

12. Save the collector configuration file.

What to do next

Use the script to start the collector. For example, use a command line similar to the following:

```
./collectorSAM5620Java.sh -Xms512m -Xmx1024m
-jar Alcatel5620Sam/Alcatel5620SamCollector.jar
-propsFile Alcatel5620Sam/Alcatel5620Sam.properties
```

Configuring the Cisco APIC REST collector
To use data from the Cisco APIC REST collector in a network discovery, you must configure the connection details between the Cisco Application Policy Infrastructure Controller (APIC) and Network Manager.

About this task

You can also configure additional information to be collected from the EMS. To configure the Cisco APIC REST collector, complete the following steps:

Procedure

1. Change to the Cisco APIC REST collector directory.

```
cd $NCHOME/precision/collectors/javaCollectors/CiscoAPICREST/
```

2. Within this directory, find the sample configuration file for the Cisco APIC REST collector and copy it to the working configuration file.

```
cp CiscoApicRestCollector.properties.sample CiscoApicRestCollector.properties
```

3. Edit the collector configuration file:
   `$NCHOME/precision/collectors/javaCollectors/CiscoAPICREST/CiscoApicRestCollector.properties`.
   This file includes the following configuration sections:
   - Collector specific properties
   - Data Acquisition properties
   - Data Source properties
   - REST Connection properties

   The following steps list the configurable parameters. The remaining properties in this file are collector system-based configurations and not meant to be changed.
4. Configure the collector port and log and trace parameters:

   port
   The port on which to run the collector. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8080.

   log.filename

   Filename for the collector log file. You can also specify a pattern for the log file name using a set of system-defined elements.The default value is `CiscoApicRestCollector.log`.

   log.level
   Takes one of the following values:
   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO
   - WARNING
   - SEVERE
   - ALL

   trace.filename
   Filename for the collector trace file. You can also specify a pattern for the trace file name using a set of system-defined elements. The default value is `CiscoApicRestCollector-trace.log`

   trace.level
   Takes one of the following values:
   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO

- WARNING
- SEVERE
- ALL

5. Configure data acquisition parameters for the collector:

collectData
Takes one of the following values. The default value is `True`.

True
Enables the collector. The collector collects data from the EMS.

False
Disables the collector. The collector does not collect data from the EMS.

DataAcquisition.GetEntities
Takes one of the following values. The default value is 1.

1: Enable
Enables download of physical entity data from the EMS.

0: Disable
Disables download of physical entity data from the EMS.

DataAcquisition.GetLayer1Connections
Takes one of the following values. The default value is 1.

1: Enable
Enables download of layer 1 connectivity data from the EMS.

0: Disable
Disables download of layer 1 connectivity data from the EMS.

DataAcquisition.GetLayer2Connections
Takes one of the following values. The default value is 1.

1: Enable
Enables download of layer 2 connectivity data from the EMS.

0: Disable
Disables download of layer 2 connectivity data from the EMS.

DataAcquisition.GetLayer3Connections
Takes one of the following values. The default value is 1.

1: Enable
Enables download of layer 3 connectivity data from the EMS.

0: Disable
Disables download of layer 3 connectivity data from the EMS.

DataAcquisition.localDataDirectory
The location to store the output files generated from the Cisco APIC. A relative or full path to the directory location is required. You can not use $NCHOME. For example, `/opt/IBM/netcool/core/precision/collectors/javaCollectors/CiscoAPICREST/data/`.

6. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

DataSource.id
Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

DataSource.descr
Description of the Cisco APIC data source.

DataSource.emsHost
IP address or hostname of the EMS.

DataSource.emsPort
Port of the EMS.

DataSource.emsUserName
The username used to connect to the Cisco APIC.

DataSource.emsPassword
The password for the user specified by the DataSource.emsUserName property.

DataSource.emsName
Name of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver. For the Cisco APIC REST collector, this identifier must be set to `ciscoapicrest`.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary
- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

7. Configure Web Services and REST connection properties for the collector.

enableSSL
Enable or disable SSL connectivity between the collector and the EMS server. This property takes the following values: `true` or `false`. The default is `false`.

Fix Pack 6 **TLSVersion**
The version of the TLS protocol. Allowed values are: SSL, SSLv2, SSLv3, TLS, TLSv1, TLSv1.1

Fix Pack 6 **MaxBufferSize**
The maximum size of REST response that the collector will process, measured in MB. The default is 1024.

keyStoreFileName

Specify the name of the keystore file that contains the SSL client certificate and trusted authority certificate.

The keystore file must be placed in the directory specified in the `pathToKeyStoreFile` parameter.

keyStorePassword

Specify the password required to access the certificate specified by the `keyStoreFileName` property.

pathToKeyStoreFile
The full path to the `keyStoreFileName` directory. You must specifiy the relative or full path to the directory location. You can not use $NCHOME. For example, `/opt/IBM/netcool/core/precision/collectors/javaCollectors/CiscoApicRest/`.

setResponseTimeout

Specify how long (in seconds) the collector waits for a response from the EMS before timing out. The default is 300.

setHttpVersion

Specify the version of the HTTP protocol that the target system supports. For Cisco APIC, this property must be set to `1.0`.

setRefreshInterval

Specify the interval (in seconds) that the collector waits between successive login refresh requests. The Cisco APIC session timeout period is 300 seconds (or 5 minutes), so this value must be less than 300. The default is 180.

Tip: If your network has performance or stability issues, set a lower value than 180.

8. Save the collector configuration file.
9. Optional: Set up SSL between the collector and Network Manager.
   a) Obtain the required SSL certificates and the Trusted Authority certificate from the Cisco APIC server administrator.
   b) Add the certificates to a local Java keystore so that they can be referenced by the KeyStore property.
   c) If you have a key and a certificate from the server in separate files, you must combine them into a single PKCS12 format file to load into a new keystore. To convert the server certificate into PKCS12 format, use the following OpenSSL toolkit command:

```
openssl pkcs12 -export -inkey key_file-in cert_file-out cert_pkcs12
```

   Where `key_file` is the key file retrieved from the server, `cert_file` is the certificate retrieved from the server, and `cert_pkcs12` is the combined file in PKCS12 format for loading into the keystore.
10. To create a Java keystore using the Keytool utility, follow these steps:
    a) Generate a keystore and self-signed certificate using the following command:

```
keytool -genkey -keyalg RSA -alias alias_name -keystore keystore_file
 -storepass keystore_password -validity 360 -keysize 2048
```

    b) Import the SSL certificate from Cisco APIC into the newly created Java keystore file using the following command:

```
keytool -import -trustcacerts -alias alias_name -file cert_file
 -keystore keystore_file
```

    c) Verify that the certificates are in a Java keystore using the following command:

```
keytool -list -v -keystore keystore_file
```

    d) Set the `keyStoreFileName` and `keyStorePassword` properties in the collector property file.
    e) Set the `enableSSL` property in the collector property file to `true`.
    f) Fix Pack 6
    If required, configure the `TLSVersion` property in the collector property file.
    g) Ensure that the `DataSource.emsPort` property in the collector property file is set to the HTTPS port.
    h) Copy the generated keystore file into the directory specified in the `pathToKeyStoreFile` property.

Results

The collector stores retrieved data as `.xml` and `.json` files in the `$NCHOME/precision/collectors/javaCollectors/CiscoAPICREST/data/` directory. This directory must have the appropriate permissions for files to be created in it.

Configuring the CiscoWorks LMS collector

To use data from the CiscoWorks LMS collector in a network discovery, you must configure the connection details between the CiscoWorks LMS EMS and Network Manager.

Before you begin

In order to enable the Network Manager CiscoWorks LMS collector integration with the CiscoWorks LMS EMS, you must first copy the jdbc driver. To do this, and subject to your verification that you have the necessary permission or authorization to do so, manually copy the jconn2.jar library from the CiscoLMS server into the Network Manager Java collector library directory at `$NCHOME$NCHOME/precision/collectors/javaCollectors/lib`.

Note: By default, the location of the jconn2.jar on the Cisco LMS EMS server is `/opt/CSCOpx/lib/classpath`.

About this task

You can also configure additional information to be collected from the EMS. To configure the CiscoWorks LMS collector, complete the following steps:

To use data from the CiscoWorks LMS collector in a network discovery, you must configure connection parameters for Cisco Open Database Schema and Cisco Data Extraction Engine used to communicate between the CiscoWorks LMS EMS and Network Manager.

Procedure

1. Change to the CiscoWorks LMS collector directory.

   ```
   cd $NCHOME/precision/collectors/javaCollectors/CiscoLMS/
   ```

2. Within this directory, find the sample configuration file for the CiscoWorks LMS collector and copy it to the working configuration file.

   ```
   cp CiscoLMSCollector.properties.sample CiscoLMSCollector.properties
   ```

3. Edit the collector configuration file:

   `$NCHOME/precision/collectors/javaCollectors/CiscoLMS/CiscoLMSCollector.properties`.

   This file includes the following configuration sections:

   - A section containing parameters for interfacing to the Cisco Open Database Schema, the Cisco Data Extraction Engine. This section also contains data acquisition flag settings.
   - A system configuration section.

   The following steps list the configurable parameters. The remaining properties in this file are collector system-based configurations and not meant to be changed.

4. Configure the collector port and log and trace parameters:

   port
   The port on which to run the collector. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file.

   log.filename
   Filename for the CiscoWorks LMS collector log file. You can also specify a pattern for the log file name using a set of system-defined elements.

log.level
Takes one of the following values:

- NONE
- FINEST
- FINER
- FINE
- CONFIGINFO
- WARNING
- SEVERE
- ALL

trace.filename
Filename for the CiscoWorks LMS collector trace file. You can also specify a pattern for the trace file name using a set of system-defined elements.

trace.level
Takes one of the following values:

- NONE
- FINEST
- FINER
- FINE
- CONFIGINFO
- WARNING
- SEVERE
- ALL

5. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

DataSource.id
Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

DataSource.descr
Description of the EMS.

DataSource.emsName
Name of the EMS.

DataSource.emsPort
Port of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary
- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

6. Configure parameters for interfacing to the Cisco LMS database:

DB.Host
IP address or hostname of the Cisco LMS database.

DB.Port
Port number of the database. The default value is 43455.

DB.DbName
The schema name of the Cisco LMS database (database schema). The default value is rmengdb.

DB.UserName
The username used to connect to the database. The default value is lmsdatafeed.

DB.Password
The password used to connect to the database. The default value is lmsdatafeed.

7. Configure parameters for interfacing to the Cisco Data Extraction Engine:

webservice.host
IP address or hostname of the Cisco Data Extraction Engine.

webservice.port
Port of the Cisco Data Extraction Engine. The default value is 1741.

webservice.servicename
Web service for the Cisco Data Extraction Engine. The default value is `/campus/servlet/CMExportServlet`.

webservice.username
User name for the Cisco Data Extraction Engine. The default value is admin.

webservice.password
User password for the Cisco Data Extraction Engine. The default value is admin.

8. Configure data processing parameters for the Cisco Data Extraction Engine.

The Cisco Data Extraction Engine can sometimes generate huge XML files for the layer 2 topology. In order to manage XML processing, the CiscoWorks LMS collector has the ability to read the XML envelope from the Cisco Data Extraction Engine and process it right away, or spool it to a flat file for later processing. By default, the Collector reads the XML envelope from the Cisco Data Extraction Engine and processes it right away.

Cisco Data Extraction Engine processing parameters are as follows:

L2TopologyXML.inputStreamEnable
Takes one of the following values:

1: Enable
Collector reads from the input stream and consumes the XML envelope.

0: Disable
Collector reads from the input stream, spools the XML envelope to a file and consumes the XML envelope.

L2TopologyXML.outputdirectory
Output directory for the spooled XML envelope if `L2TopologyXML.inputStreamEnable = 1`. The default value is `../CiscoLMS/tmp`.

L2TopologyXML.outputfilename
Output file for the spooled XML envelope if `L2TopologyXML.inputStreamEnable = 1`. The default value is `output.xml`.

9. Configure data acquisition parameters for the collector:

   collectData
   Takes one of the following values. The default value is `True`.

   True
   Enables the collector. The collector collects data from the CiscoWorks LMS EMS.

   False
   Disables the collector. The collector does not collect data from the CiscoWorks LMS EMS.

   DataAcquisition.GetEntities
   Takes one of the following values. The default value is 1.

   1: Enable
   Enables download of physical entity data from the CiscoWorks LMS EMS.

   0: Disable
   Disables download of physical entity data from the CiscoWorks LMS EMS.

   DataAcquisition.GetLayer2Connections
   Takes one of the following values. The default value is 1.

   1: Enable
   Enables download of layer 2 connectivity data from the CiscoWorks LMS EMS.

   0: Disable
   Disables download of layer 2 connectivity data from the CiscoWorks LMS EMS.

10. Save the collector configuration file.

Configuring the Huawei M2000 collector
To use data from the Huawei M2000 collector in a network discovery, you must configure the collector to process the XML files.

About this task

You can also configure additional information to be collected from the EMS. To configure the Huawei M2000 collector, complete the following steps:

Procedure

1. Change to the Huawei M2000 collector directory.

   ```
   cd $NCHOME/precision/collectors/javaCollectors/HuaweiM2K/
   ```

2. Within this directory, find the sample configuration file for the collector and copy it to the working configuration file.

   ```
   cp HuaweiM2KCollector.properties.sample HuaweiM2KCollector.properties
   ```

3. Edit the collector configuration file:
   `$NCHOME/precision/collectors/javaCollectors/HuaweiM2K/HuaweiM2KCollector.properties`.

   The file includes the following sections:

   Collector properties
   General configuration parameters for the collector, such as port number and log and trace details.

   Data Source properties
   Details of the EMS that the collector is connecting to. Data from these fields is used by Network Manager to model the EMS.

   Data Acquisition properties
   Parameters that specify which data to collect from the EMS.

File processing properties
Parameters that specify the options for file processing.

4. In the Collector properties section, configure the general collector properties:

port
Port on which to run the embedded collector server. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8080.

log.filename
Filename for the Huawei M2000 collector log file. The default value is `HuaweiM2KCollector.log`.

log.level
Level of logging for the collector framework. Logging level takes one of the following values:

- NONE
- FINEST
- FINER
- FINE
- CONFIG
- INFO
- WARNING
- SEVERE
- ALL

The default value is INFO.

trace.filename
Filename for the Huawei M2000 collector trace file. The default value is `HuaweiM2KCollector-trace.log`.

trace.level
Level of trace for the collector framework. The default value is INFO.

5. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

DataSource.id
Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

DataSource.descr
Description of the EMS.

DataSource.emsName
Name of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary
- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

6. In the Data Acquisition properties section, configure data acquisition parameters:

collectData
Set to `true` to acquire data or `false` to not acquire data.

DataAcquisition.GetEntities
Takes one of the following values. The default value is `0`.

1: Enable
Enables discovery of physical entity data from the XML file.

0: Disable
Disables discovery of physical entity data from the XML file.

7. In the File processing properties section, configure file processing parameters:

file.localDirectory
The local directory where the files to be processed by the collector are located. Specify the full path to the directory location.

8. Save the collector configuration file.

Configuring the Huawei CORBA TMF 814 collector
To use data from the Huawei CORBA TMF 814 collector in a network discovery, you must configure the connection details between the Huawei CORBA TMF EMS and Network Manager.

About this task

You can also configure additional information to be collected from the EMS. To configure the Huawei CORBA TMF 814 collector, complete the following steps:

Procedure

1. Change to the Huawei CORBA TMF 814 collector directory.

```
cd $NCHOME/precision/collectors/javaCollectors/HuaweiCorbaTMF814/
```

2. Within this directory, find the sample configuration file for the Huawei CORBA TMF 814 collector and copy it to the working configuration file.

```
cp HuaweiCorbaTMF814Collector.properties.sample
HuaweiCorbaTMF814Collector.properties
```

3. Edit the collector configuration file:
`$NCHOME/precision/collectors/javaCollectors/HuaweiCorbaTMF814/HuaweiCorbaTMF814Collector.properties`.

The file is made up of the following sections:

Collector properties
General configuration parameters for the collector, such as port number and log and trace details.

Data Source properties
Details of the EMS that the collector is connecting to. Data from these fields is used by Network Manager to model the EMS.

Data Acquisition properties
Parameters that specify which data to collect from the EMS.

ORB Initialization properties
CORBA ORB-specific parameters.

NameService properties
Naming service parameters.

EmsSessionFactory properties
Parameters that specify how the collector should obtain the EMSSessionFactory object reference.

Manager properties
Parameters that specify the names of the manager interfaces.

Iterator properties
Parameters that define the behavior of the CORBA iterator.

Filtering properties
Parameters that define which devices should be excluded from processing.

4. In the Collector properties section, configure the general collector properties:

port
Port on which to run the embedded collector server. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8080.

log.filename
Filename for the Huawei CORBA TMF 814 collector log file. The default value is `HuaweiCorbaTMF814Collector.log`.

log.level
Level of logging for the collector framework. The default value is INFO.

trace.filename
Filename for the Huawei CORBA TMF 814 collector trace file. The default value is `HuaweiCorbaTMF814Collector-trace.log`.

trace.level
Level of trace for the collector framework. The default value is INFO.

5. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

DataSource.id
Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

DataSource.descr
Description of the EMS.

DataSource.emsHost
Hostname of the EMS.

DataSource.emsName
Name of the EMS.

DataSource.emsPort
Port of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary

- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

DataSource.emsUserName
Username of the EMS.

DataSource.emsPassword
Password of the EMS.

Note: The DataSource.emsHost and DataSource.emsPort fields are compulsory if the `nameService.useNameService` variable is TRUE.

6. In the Data Acquisition properties section, configure data acquisition parameters:

collectData
Takes one of the following values. The default value is `True`.

True
Enables the collector. The collector collects data from the EMS.

False
Disables the collector. The collector does not collect data from the EMS.

DataAcquisition.GetEntities
Takes one of the following values. The default value is `0`.

1: Enable
Enables download of physical entity data from the EMS.

0: Disable
Disables download of physical entity data from the EMS.

Fix Pack 2 **DataAcquisition.GetEquipmentAdditionalInfo**
Takes one of the following values. The default value is `0`.

1: Enable
Enables download of equipment additional information from the EMS.

0: Disable
Disables download of equipment additional information from the EMS.

DataAcquisition.GetCTPs
Takes one of the following values. The default value is `0`.

1: Enable
Enables download of ConnectionTP data from the EMS.

0: Disable
Disables download of ConnectionTP data from the EMS.

DataAcquisition.GetTopoLinks
Takes one of the following values. The default value is `0`.

1: Enable
Enables download of layer 1 topological link connectivity from the EMS.

0: Disable
Disables download of layer 1 topological link connectivity from the EMS.

DataAcquisition.GetSNCs
Takes one of the following values. The default value is `0`.

1: Enable
Enables discovery of Layer 1 subnetwork connections connectivity.

0: Disable
Disables discovery of Layer 1 subnetwork connections connectivity.

7. In the ORB Initialization properties section, do not modify the following ORB initialization properties unless directed by IBM Support:

orbProp.1.name=org.omg.CORBA.ORBClass
orbProp.1.value=com.ibm.CORBA.iiop.ORB
Specifies the IBM ORB implementation to be activated.

orbProp.2.name=org.omg.CORBA.ORBSingletonClass
orbProp.2.value=com.ibm.rmi.corba.ORBSingleton
Specifies the IBM ORB implementation to be activated.

orbProp.3.name=com.ibm.CORBA.Debug.Output
orbProp.3.value=../log/orbtrc.log
Specifies the log file to which to write ORB error messages when the collector fails to connect to the CORBA service on the EMS.

8. In the NameService properties section, configure the naming service properties:

nameService.useNameService
Takes one of the following values. Default value is `True`.

True
Collector obtains the root naming context reference by accessing the naming service using a corbaloc URL

False
Collector obtains the root naming context reference from the specified IOR file and bypasses the naming service.

nameService.iorFile
The full filepath to the IOR file containing the NameService reference. Supports local filepath and remote filepath; for example, HTTP, and FTP.

nameService.nameServiceName
Service name for the naming service.

9. In the EmsSessionFactory properties section, configure the EmsSessionFactory properties:

emsSessionFactory.useNameService
Takes one of the following values. Default value is `True`.

True
Collector obtains the EmsSessionFactory_I reference by using the naming service and by resolving the specified naming context properties.

False
Collector obtains the EmsSessionFactory_I reference from the specified IOR file and bypasses the naming service.

emsSessionFactory.iorFile
Full filepath to the IOR file containing the EmsSessionFactory reference. Supports local filepath and remote filepath; for example, HTTP, FTP.

emsSessionFactory.namingContext.*
These naming context properties represent the naming context bindings for resolving the EmsSessionFactory reference from the naming service. All entries must be in *.id and *.kind pairs and specified in the correct sequence in order for the EmsSessionFactory reference to be correctly resolved.

10. In the Manager properties section, configure the manager properties:

manager.emsMgr
Name for the EMSMgr interface. The default value is `EMS`.

manager.equipmentInventoryMgr
Name for the EquipmentInventoryMgr interface. The default value is `EquipmentInventory`.

manager.managedElementMgr
Name for the ManagedElementMgr interface. The default value is `ManagedElement`.

manager.multiLayerSubnetworkMgr
Name for the MultiLayerSubnetworkMgr interface. The default value is `MultiLayerSubnetwork`.

11. In the Iterator properties section, configure the CORBA iterator properties:

iterator.resultSize
Specifies the maximum number of results returned for each set by the CORBA Iterator. The default value is 20. Increase this value if you want to decrease the number of iterations and increase the size of the data response from the EMS.

iterator.destroyIterator
Set to true if you want the CORBA Iterator to destroy the corresponding iterator object on the EMS side. The default value is `false`. Enable this property only if the EMS does not automatically perform memory garbage collection.

12. In the filtering properties section, configure the device filtering properties:

filter.enabled
Takes one of the following values:

- True: filter is enabled; the collector will exclude devices with the specified filter.productName property.
- False: filter is disabled. This is the default value.

filter.productName
Specify here the model or type of device to excluded from processing. This property has no default value, and may be left blank. For multiple entries, separate the product names using commas. For example;

```
filter.productName=Virtual NE,OptiX OSN 7500,OptiX DWDM OADM
```

During processing, the values specified in the filter.productName property are compared with the contents of the entityData.Description field.

13. Save the collector configuration file.

Configuring the MTOSISoap collector
To use data from the MTOSISoap collector in a network discovery, you must configure the connection details between the EMS and Network Manager.

About this task

You can also configure additional information to be collected from the EMS. To configure the MTOSISoap collector, complete the following steps:

Procedure

1. Change to the MTOSISoap collector directory.

```
cd $NCHOME/precision/collectors/javaCollectors/MTOSISoap/
```

2. Within this directory, find the sample configuration file for the MTOSISoap collector and copy it to the working configuration file.

```
cp MTOSISoapCollector.properties.sample
MTOSISoapCollector.properties
```

3. Edit the collector configuration file:

`$NCHOME/precision/collectors/javaCollectors/MTOSISoap/MTOSISoapCollector.properties`.

The file includes the following sections:

Collector properties
General configuration parameters for the collector, such as port number and log and trace details.

Data Acquisition properties
Parameters that specify which data to collect from the EMS.

Data Source properties
Details of the EMS that the collector is connecting to. Data from these fields is used by Network Manager to model the EMS.

4. In the Collector properties section, configure the general collector properties:

port
Port on which to run the embedded collector server. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8080.

log.filename
Filename for the MTOSISoap collector log file. The default value is `MTOSISoapCollector.log`.

log.level
Level of logging for the collector framework. Logging level takes one of the following values:

- NONE
- FINEST
- FINER
- FINE
- CONFIG
- INFO
- WARNING
- SEVERE
- ALL

The default value is INFO.

trace.filename
Filename for the MTOSISoap collector trace file. The default value is `MTOSISoapCollector-trace.log`.

trace.level
Level of trace for the collector framework. The default value is INFO.

5. In the Data Acquisition properties section, configure data acquisition parameters:

collectData
Takes one of the following values. The default value is `True`.

True
Enables the collector. The collector collects data from the EMS.

False
Disables the collector. The collector does not collect data from the EMS.

DataAcquisition.MSLN
Value of the MultiLayer Subnetwork of the EMS. Default value is 1.

DataAcquisition.ManagementDomain
Name of the management domain of the EMS. Default value is `Huawei/U2000`.

DataAcquisition.ManagedElementsBatchSize
Defines the maximum number of MTOSI objects that can be included in a SOAP EMS response for a ManagedElementsRequest. The default value is 4500 objects.

Note: You must set the `DataAcquisition.ManagedElementsBatchSize` and `DataAcquisition.TopologicalLinksBatchSize` parameters to values that are less than the number of elements on the EMS. Otherwise, discovery fails and you see this error in the discovery logs:

```
Register Iterator faild.The number of Active Iterators is more than allowed!
```

DataAcquisition.TopologicalLinksBatchSize
Defines the maximum number of MTOSI objects that can be included in a SOAP EMS response for a TopologicalLinkRequest. The default value is 6000 objects.

Note: You must set the `DataAcquisition.ManagedElementsBatchSize` and `DataAcquisition.TopologicalLinksBatchSize` parameters to values that are less than the number of elements on the EMS. Otherwise, discovery fails and you see this error in the discovery logs:

```
Register Iterator faild.The number of Active Iterators is more than allowed!
```

DataAcquisition.GetEntities
Takes one of the following values. The default value is `0`.

1: Enable
Enables download of physical entity data from the EMS.

0: Disable
Disables download of physical entity data from the EMS.

DataAcquisition.GetLayer2Connections
Takes one of the following values. The default value is `0`.

1: Enable
Enables download of layer 2 connectivity data from the EMS.

0: Disable
Disables download of layer 2 connectivity data from the EMS.

DataAcquisition.GetLayer3Connections
Takes one of the following values. The default value is `0`.

1: Enable
Enables download of layer 3 connectivity data from the EMS.

0: Disable
Disables download of layer 3 connectivity data from the EMS.

Fix Pack 4 **DataAcquisition.receiveTimeout**
The maximum time in seconds to wait for a response from the EMS. The default is 300 seconds.

6. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

DataSource.id
Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

DataSource.descr
Description of the EMS.

DataSource.emsHost
Hostname of the EMS.

DataSource.emsName
Name of the EMS.

DataSource.emsPort
Port of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary
- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

DataSource.emsUserName
Username of the EMS.

DataSource.emsPassword
Password of the EMS.

7. Save the collector configuration file.

Configuring the Nokia Solutions and Networks NetAct XML Interface for Configuration Management collector

The Nokia Solutions and Networks (NSN) NetAct XML Interface for Configuration Management collector processes 2G, 3G and LTE RAN data by utilizing the configuration management XML file for the NSN NetAct EMS. This XML file contains the NetAct Configurator network configuration data, in RAML/CM2 format.

About this task

The collector can retrieve the configuration management XML file for the NSN NetAct EMS in one of the following ways:

- By connecting to the NSN NetAct EMS. In this case the collector uses either FTP or SFTP. Therefore if you want to use this method, you must configure the collector properties file with the FTP and SFTP connection details between the NSN NetAct EMS and Network Manager.
- By accessing the XML file in a designated local directory. If you want to use this method, you must specify the local directory that will contain XML files retrieved from the EMS.

In either case, you must configure the relevant data acquisition properties within the collector configuration file, as specified in the procedure that follows.

Note: The collector stores the operating system timestamp of each processed XML file. If, during a discovery, the collector finds that the last modified timestamp of the current XML file is the same as the timestamp recorded during the last discovery, the file is not processed. This ensures that only files containing device data updated since the last discovery are processed.

To configure the NSN NetAct XML Interface for Configuration Management collector, complete the following steps:

Procedure

1. Change to the NSN NetAct XML Interface for Configuration Management collector directory.

   ```
   cd $NCHOME/precision/collectors/javaCollectors/NetActCMDump/
   ```

2. Within this directory, find the sample configuration file for the collector and copy it to the working configuration file.

   ```
   cp NetActCMDumpCollector.properties.sample NetActCMDumpCollector.properties
   ```

3. Edit the collector configuration file:
   `$NCHOME/precision/collectors/javaCollectors/NetActCMDump/NetActCMDumpCollector.properties`.
   This file includes the following configuration sections:
   - Collector configuration properties
   - Data acquisition properties
   - Data source properties

   Note: The following steps list the configurable parameters. The remaining properties in this file are collector system-based configuration values and are not meant to be changed.

4. Configure the following collector properties:

   port
   Port on which to run the collector. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8080.

   log.filename
   Filename for the collector log file. You can also specify a pattern for the log file name using a set of system-defined elements.

   log.level
   Level of logging for the collector framework. Takes one of the following values:
   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO
   - WARNING
   - SEVERE
   - ALL

   The default value is INFO.

   trace.filename
   Filename for the collector trace file. You can also specify a pattern for the trace file name using a set of system-defined elements.

   trace.level
   Level of trace for the collector framework. Takes one of the following values:
   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO

- WARNING
- SEVERE
- ALL

The default value is INFO.

5. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

DataSource.id
Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

DataSource.descr
Description of the EMS.

DataSource.emsHost
Hostname of the EMS.

DataSource.emsName
Name of the EMS.

DataSource.emsPort
Port of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary
- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

DataSource.emsUserName
Username of the EMS.

DataSource.emsPassword
Password of the EMS.

6. Configure data acquisition parameters for the collector:

collectData
Takes one of the following values. The default value is `True`.

True
Enables the collector. The collector collects data from the EMS.

False
Disables the collector. The collector does not collect data from the EMS.

DataAcquisition.GetEntities
Takes one of the following values. The default value is 0.

1: Enable
Enables download of physical entity data from the EMS.

0: Disable
Disables download of physical entity data from the EMS.

DataAcquisition.getRanTopology
Takes one of the following values. The default value is 1.

1: Enable
Enables download of RAN connectivity data for the Net Act EMS.

0: Disable
Disables download of RAN connectivity data for the Net Act EMS.

DataAcquisition.localDataDirectory
Specifies the location of XML files retrieved from the EMS.

DataAcquisition.postProcessFlag
Defines how the collector handles the data file after processing. The default valus is 0.

0 : Delete
Deletes the xml file after processing.

1 : Move
Moves the xml file after processing. Used in conjunction with parameter DataAcquisition.moveLocation.

2 : Leave
Leaves the xml file in the data directory after processing. Used in conjunction with parameter DataAcquisition.localDataDirectory.

DataAcquisition.moveLocation
Specifies the location for the files to be moved post processing. This parameter only takes effect if the variable DataAcquisition.postProcessFlag has the value of 1.

DataAcquisition.loadFirstRun
Defines whether to process XML files in the XML local data directory upon start of the collector, without waiting for the XML-RPC call from the GUI.

0
Do not process data directory upon collector start. Other values will process the data directory immediately.

1
Process data directory upon collector start.

DataAcquisition.downloadFile
Defines whether to download the XML file using FTP or SFTP (secure FTP) from the EMS upon discovery.

0
Do not download.

1
Download.

The default value is 1.

DataAcquisition.maskCredentials
Defines whether to mask the user credentials in the log when issuing the FTP or SFTP command to transfer the XML file from the EMS.

0
Do not mask credentials.

1
Mask credentials.

DataAcquisition.remoteFtpHost
Hostname or IP of the EMS where the XML data is generated.

DataAcquisition.remoteDir
Remote directory containing the file or files to be processed.

DataAcquisition.remoteFile
Name of the XML file that contains the CM data. If retrieving files using FTP or SFTP, then set this property to a wildcard value, such as `*.xml` or `*.xml.gz`.

Note: If the remote directory configured using the DataAcquisition.remoteDir property contains more than the current file, and you have set DataAcquisition.remoteFile as a wildcard value, then all files that match the wildcard will be transferred from the remote directory and processed. To prevent the collector retrieving all previously processed files the remote directory must only contain the most recent XML file or files to be processed.

DataAcquisition.remoteFtpUser
Username to be used for the FTP or SFTP connection.

DataAcquisition.remoteFtpPassword
Password to be used for the FTP or SFTP connection.

DataAcquisition.remoteFtpPort
Port to be used for the FTP or SFTP connection.

DataAcquisition.secureConnection
Indicates whether to use secure connection (SFTP) when retrieving the cmdump files.

1
Use SFTP to connect to the EMS.

0
Use FTP to connect to the EMS.

Default is 0.

DataAcquisition.ftpTimeout
Timeout interval for the FTP process.

DataAcquisition.technologyType
Mobile technology that will be handled by the collector. Available options are as follows:

- 2G3G
- LTE
- ALL

Default is 2G3G.

7. Save the collector configuration file.

Fix Pack 6 *Configuring the NokiaOMS1350 Java collector*

This collector retrieves data from the Nokia OMS 1350 EMS. Before running the collector in a network discovery, you must copy certain required files and configure the connection details between the EMS and Network Manager.

About this task

You can also configure additional information to be collected from the EMS.

To configure the collector, complete the following steps:

Procedure

1. Change to the collector directory:

```
cd $NCHOME/precision/collectors/javaCollectors/NokiaOMS1350/
```

2. Within this directory, find the sample configuration file for the collector and copy it to the working configuration file using a command similar to the following example:

```
cp NokiaOMS1350RestCollector.properties.sample NokiaOMS1350RestCollector.properties
```

3. The configuration file consists of the following sections:

   Collector properties
   General configuration parameters for the collector, such as port number and log and trace details.

   Data Acquisition properties
   Parameters that specify which data to collect from the EMS.

   Data Source properties
   Details of the EMS that the collector is connecting to. Data from these fields is used by Network Manager to model the EMS.

   REST connection properties
   Properties relating to the REST connection to the EMS.

   Debug properties
   Properties for debugging.

4. Configure the collector port and log and trace parameters:

   port
   The port on which to run the collector. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8443.

   log.filename
   Filename for the collector log file. You can also specify a pattern for the log file name using a set of system-defined elements.

   log.level
   Takes one of the following values:

   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO
   - WARNING
   - SEVERE
   - ALL

   trace.filename
   Filename for the collector trace file. You can also specify a pattern for the trace file name using a set of system-defined elements.

   trace.level
   Takes one of the following values:

   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO
   - WARNING
   - SEVERE
   - ALL

5. In the Data Acquisition properties section, configure data acquisition parameters:

   collectData
   The default value is `true`.

   True
   Enables the collector. The collector collects data from the EMS.

   False
   Disables the collector. The collector does not collect data from the EMS.

   DataAcquisition.GetEntities
   Takes one of the following values. The default value is 1.

   1: Enable
   Enables download of physical entity data.

   0: Disable
   Disables download of physical entity data.

   DataAcquisition.GetLayer1Connections
   Takes one of the following values. The default value is 1.

   1: Enable
   Enables download of layer 1 connectivity data.

   0: Disable
   Disables download of layer 1 connectivity data.

   DataAcquisition.GetLayer2Connections
   Takes one of the following values. The default value is 1.

   1: Enable
   Enables download of layer 2 connectivity data.

   0: Disable
   Disables download of layer 2 connectivity data.

   DataAcquisition.GetLayer3Connections
   Takes one of the following values. The default value is 1.

   1: Enable
   Enables download of layer 3 connectivity data.

   0: Disable
   Disables download of layer 3 connectivity data.

   DataAcquisition.localDataDirectory
   Specifies the location of the output files generated from the EMS. Use a relative or absolute path to the directory location. Do not use directory variables such as $NCHOME.

6. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

   DataSource.id
   Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

   DataSource.descr
   Description of the EMS.

   DataSource.emsHost
   The host name of the EMS.

   DataSource.emsPort
   Port of the EMS.

   DataSource.emsUserName
   The user name to use when logging into the EMS.

DataSource.emsPassword
The password to use when logging into the EMS.

DataSource.emsServiceName
The EMS service name.

DataSource.emsPresHost
The hostname of the presentation server.

DataSource.emsPresUsername
The username for logging into the presentation server.

DataSource.emsPresPassWord
The password for logging into the presentation server.

DataSource.emsName
Name of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary
- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

DataSource.emsNumNEElements
The number of network elements that will be processed to the DataStore.

7. Configure REST connection properties for the collector.

enableSSL
Enable or disable SSL connectivity between the collector and the EMS server. This property takes the following values: `true` or `false`. The default is `false`.

pathToKeyStoreFile
The full path to the `keyStoreFileName` directory. You must specify the relative or full path to the directory location. You can not use $NCHOME. For example, `/opt/IBM/netcool/core/precision/collectors/javaCollectors/CiscoApicRest/`.

keyStoreFileName

Specify the name of the keystore file that contains the SSL client certificate and trusted authority certificate.

The keystore file must be placed in the directory specified in the `pathToKeyStoreFile` parameter.

keyStorePassword

Specify the password required to access the certificate specified by the `keyStoreFileName` property.

setResponseTimeout

Specify how long (in seconds) the collector waits for a response from the EMS before timing out. The default is 300.

setHttpVersion

Specify the version of the HTTP protocol that the target system supports. The default is 1.1.

setRefreshInterval

Specify the interval (in seconds) that the collector waits between successive login refresh requests. The default is 600.

TLSVersion
The version of the TLS protocol. Allowed values are: SSL, SSLv2, SSLv3, TLS, TLSv1, TLSv1.1

8. In the Debug properties section, configure the following properties.

```
#   emsSelectElements    - Debug the Selected EMS Items
#   emsNENode.1    - Select NE Node name 1 for debug
#   emsNENode.2    - Select NE Node name 2 for debug
#   enable_manual_service_ticket    - Enable or Disable the Manual service Ticket
#   service_ticket- Set the Service Ticket Value for Override and validating with manual
way.
#---------------------------------------

=ST-1083-Gz5plCWlPlq9gBkrLfj4-cas01.example.org
```

Debug.emsSelectElements
Set to `true` to enable debugging for the specified EMS items. Set to `false` to turn off debugging.

Debug.emsNENode.1
The name of the first node that you want to debug.

Debug.emsNENode.2
The name of the second node that you want to debug.

Debug.enable_manual_service_ticket
Set to `true` to enable the manual service ticket. Set to `false` to turn off the manual service ticket.

Debug.service_ticket
The service ticket value that you want to override and validate manually.

9. Save the collector configuration file.

Configuring the Nokia Solutions and Networks NetViewer collector
To use data from the Nokia Solutions and Networks (NSN) NetViewer collector in a network discovery, you must configure the connection details between the NSN NetViewer EMS and Network Manager.

About this task

You can also configure additional information to be collected from the EMS. To configure the NSN NetViewer collector, complete the following steps:

Procedure

1. Change to the NSN NetViewer collector directory.

```
cd $NCHOME/precision/collectors/javaCollectors/NetViewer/
```

2. Within this directory, find the sample configuration file for the NSN NetViewer collector and copy it to the working configuration file.

```
cp NetViewerCollector.properties.sample NetViewerCollector.properties
```

3. Edit the collector configuration file:

`$NCHOME/precision/collectors/javaCollectors/NetViewer/NetViewerCollector.properties`.

4. Configure the following collector properties:

   port
   Port on which to run the embedded collector server. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8080.

   log.filename
   Filename for the NSN NetViewer collector log file. The default value is `NetViewerCollector.log`.

   log.level
   Level of logging for the collector framework. The default value is INFO.

   trace.filename
   Filename for the NSN NetViewer collector trace file. The default value is `NetViewerCollector-trace.log`.

   trace.level
   Level of trace for the collector framework. The default value is INFO.

5. Configure data acquisition parameters for the collector:

   collectData
   Takes one of the following values. The default value is `True`.

   True
   Enables the collector. The collector collects data from the EMS.

   False
   Disables the collector. The collector does not collect data from the EMS.

   DataAcquisition.GetEntities
   Takes one of the following values. The default value is `0`.

   1: Enable
   Enables download of physical entity data from the EMS.

   0: Disable
   Disables download of physical entity data from the EMS.

   DataAcquisition.GetLayer2Connections
   Takes one of the following values. The default value is `0`.

   1: Enable
   Enables download of layer 2 connectivity data from the EMS.

   0: Disable
   Disables download of layer 2 connectivity data from the EMS.

6. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

   DataSource.id
   Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

   DataSource.descr
   Description of the EMS.

   DataSource.emsHost
   Hostname of the EMS.

   DataSource.emsName
   Name of the EMS.

   DataSource.emsPort
   Port of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary
- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

DataSource.emsUserName
Username of the EMS.

DataSource.emsPassword
Password of the EMS.

7. Do not modify the following ORB initialization properties unless directed by IBM Support:

   These properties are codified as name-value pair parameters.

   orbProp.1.name=org.omg.CORBA.ORBClass
   orbProp.1.value=com.ibm.CORBA.iiop.ORB
   Specifies the IBM ORB implementation to be activated.

   orbProp.2.name=org.omg.CORBA.ORBSingletonClass
   orbProp.2.value=com.ibm.rmi.corba.ORBSingleton
   Specifies the IBM ORB implementation to be activated.

   orbProp.3.name=com.ibm.CORBA.Debug.Output
   orbProp.3.value=../log/orbtrc.log
   Specifies the log file to which to write ORB error messages when the collector fails to connect to the CORBA service on the EMS.

   orbProp.4.name=com.ibm.CORBA.ORBWCharDefault
   orbProp.4.value=UTF16
   Indicates that character encoding set UTF16 is to be used for integration with the NSN NetViewer EMS.

   orbProp.5.name=com.ibm.CORBA.ListenerPort
   orbProp.5.value=10005
   Integration with the NSN NetViewer EMS uses a Corba Interoperable Object Reference (IOR) file. Interfacing using an IOR file requires bidirectional communication. To facilitate this, the ListenerPort parameter is set to port 10005, and this is the port to be used for communication from the NSN NetViewer EMS to the collector. This parameter setting is especially important if you want to integrate the collector to the NSN NetViewer EMS in a NAT or firewall environment.

8. Configure the naming service properties:

   nameService.useNameService
   Takes one of the following values. Default value is `True`.

True
Collector obtains the root naming context reference by accessing the naming service using a corbaloc URL

False
Collector obtains the root naming context reference from the specified IOR file and bypasses the naming service.

nameService.iorFile
The full filepath to the IOR file containing the NameService reference. Supports local filepath and remote filepath; for example, HTTP, and FTP.

nameService.nameServiceName
Service name for the naming service.

9. Configure the EmsSessionFactory properties:

emsSessionFactory.useNameService
Takes one of the following values. Default value is `True`.

True
Collector obtains the EmsSessionFactory_I reference by using the naming service and by resolving the specified naming context properties.

False
Collector obtains the EmsSessionFactory_I reference from the specified IOR file and bypasses the naming service.

emsSessionFactory.iorFile
Full filepath to the IOR file containing the EmsSessionFactory reference. Supports local filepath and remote filepath; for example, HTTP, FTP.

10. Configure the manager properties:

manager.emsMgr
Name for the EMSMgr interface. The default value is `EMS`.

manager.equipmentInventoryMgr
Name for the EquipmentInventoryMgr interface. The default value is `EquipmentInventory`.

manager.managedElementMgr
Name for the ManagedElementMgr interface. The default value is `ManagedElement`.

manager.multiLayerSubnetworkMgr
Name for the MultiLayerSubnetworkMgr interface. The default value is `MultiLayerSubnetwork`.

11. In the Iterator properties section, configure the CORBA iterator properties:

iterator.resultSize
Specifies the maximum number of results returned for each set by the CORBA Iterator. The default value is `20`. Increase this value if you want to decrease the number of iterations and increase the size of the data response from the EMS.

iterator.destroyIterator
Set to true if you want the CORBA Iterator to destroy the corresponding iterator object on the EMS side. The default value is `false`. Enable this property only if the EMS does not automatically perform memory garbage collection.

12. Configure the extraInfo mappings properties:

These properties map the values of the provided fields into the respective tables in the NCIM database. Multiple values can be specified, but must be differentiated by a numeral in the properties; for example, chassisData.2.extraInfo , shelfData.3.extraInfo , and so on.

- chassisData.1.extraInfo
- shelfData.1.extraInfo
- slotData.1.extraInfo
- moduleData.1.extraInfo

- ptpData.1.extraInfo
- ctpData.1.extraInfo
- topologyData.1.extraInfo

13. Save the collector configuration file.

Configuring the Tellabs INM8000 collector
To use data from the Tellabs INM8000 collector in a network discovery, you must configure the connection details between the Tellabs INM8000 EMS and Network Manager.

Before you begin

In order to enable the Network Manager Tellabs INM8000 collector integration with the Tellabs INM8000 EMS, you must first copy the following two files to the Network Manager Tellabs INM8000 collector:

- `tol.jar` library file
- `NbifAttributeValues.ini` lookup file

To do this, and subject to your verification that you have the necessary permission or authorization to do so, perform the following manual copy operations:

- Copy the `tol.jar` library file from the Tellabs INM8000 server into the Network Manager Java collector library directory at `$NCHOME/precision/collectors/javaCollectors/lib`.
- Copy the `NbifAttributeValues.ini` lookup file from the Tellabs INM8000 server into the Network Manager Java Tellabs INM8000 collector dat directory at `$NCHOME/precision/collectors/javaCollectors/TellabsINM8000/dat`.

Note: You can obtain the `tol.jar` library file and the `NbifAttributeValues.ini` lookup file from the Tellabs INM8000 EMS server, preferably from a server that is running version INM SR5.0-SP2. These files are also available on the CD-ROM that came with the Tellabs INM8000 installation and labelled: `Tellabs 8000 Intelligent Network Manager SRxx-SPxx Inventory and PePerformance adapter, JavaAPI and NBITestClient for Northbound Interface`.

About this task

You can also configure additional information to be collected from the EMS. To configure the Tellabs INM8000 collector, complete the following steps:

Procedure

1. Change to the Tellabs INM8000 collector directory.

   ```
   cd $NCHOME/precision/collectors/javaCollectors/TellabsINM8000/
   ```

2. Within this directory, find the sample configuration file for the Tellabs INM8000 collector and copy it to the working configuration file.

   ```
   cp TellabsINM8000Collector.properties.sample TellabsINM8000Collector.properties
   ```

3. Edit the collector configuration file:
   `$NCHOME/precision/collectors/javaCollectors/TellabsINM8000/TellabsINM8000Collector.properties`.
   This file includes the following configuration sections:
   - Collector configuration properties
   - Data acquisition properties
   - Data source properties
   - Mapping properties

The following steps list the configurable parameters. The remaining properties in this file are collector system-based configurations and not meant to be changed.

4. Configure the collector port and log and trace parameters:

   port
   The port on which to run the collector. The port must match the port configured in the insert into the `collectorFinder.collectorRules` table in the `DiscoCollectorFinderSeeds.cfg` file. The default value is 8080.

   log.filename
   Filename for the collector log file. You can also specify a pattern for the log file name using a set of system-defined elements. The default value is `TellabsINM8000Collector.log`.

   log.level
   Takes one of the following values:

   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO
   - WARNING
   - SEVERE
   - ALL

   trace.filename
   Filename for the collector trace file. You can also specify a pattern for the trace file name using a set of system-defined elements. The default value is `TellabsINM8000Collector-trace.log`.

   trace.level
   Takes one of the following values:

   - NONE
   - FINEST
   - FINER
   - FINE
   - CONFIGINFO
   - WARNING
   - SEVERE
   - ALL

5. Configure parameters for the Tellabs-specific region identifier.

   regionId.enable
   If the Tellabs INM8000 EMS supports region identifiers then user should be able to collect data based on the region identifier configured in the EMS. This parameter takes one of the following values:

   - True
   - False

   The default value is True.

   regionId.set
   Defines the region identifier (regid) number to filter. This parameter can take the form of a single region identifier, or a combination of multiple region identifiers. For example:

   - Single region identifier: `regionId.set=5`
   - Multiple region identifier: `regionId.set=5,10,11`

6. Configure parameters for Tellabs-specific batch processing.

   By using batch processing, Network Manager is able to discover all nodes from the EMS. If the parameter `regionId.enable` is set to False, then the series of settings batch.*entity*.size applies, where entity can be any of the following:

   - subrack
   - unit
   - module
   - interface
   - trunk

   Define the batch size for each of the entities:

   - batch.subrack.size
   - batch.unit.size
   - batch.module.size
   - batch.interface.size
   - batch.trunk.size

   For example, defined each of these parameters for the `getFilteredAllData` API usage. The batch size refers to number of nodes (nid) to be queried and processed by Network Manager at any one time. Each batch size value must be greater than 1. Refer to the supported entities in the Tellabs NBI information model. Some typical values are as follows:

   - batch.subrack.size=100
   - batch.unit.size=50
   - batch.module.size=50
   - batch.interface.size=20
   - batch.trunk.size=20
7. Configure data acquisition parameters for the collector:

   collectData
   Takes one of the following values. The default value is `True`.

   True
   Enables the collector. The collector collects data from the EMS.

   False
   Disables the collector. The collector does not collect data from the EMS.

   DataAcquisition.GetEntities
   Takes one of the following values. The default value is 1.

   1: Enable
   Enables download of physical entity data from the EMS.

   0: Disable
   Disables download of physical entity data from the EMS.

   DataAcquisition.GetLayer1Connections
   Takes one of the following values. The default value is 1.

   1: Enable
   Enables download of layer 1 connectivity data from the EMS.

   0: Disable
   Disables download of layer 1 connectivity data from the EMS.

   DataAcquisition.GetLayer2Connections
   Takes one of the following values. The default value is 1.

1: Enable
Enables download of layer 2 connectivity data from the EMS.

0: Disable
Disables download of layer 2 connectivity data from the EMS.

DataAcquisition.GetLayer3Connections
Takes one of the following values. The default value is 1.

1: Enable
Enables download of layer 3 connectivity data from the EMS.

0: Disable
Disables download of layer 3 connectivity data from the EMS.

8. You can optionally configure details about the source element management system (EMS) in the Data Source properties section by configuring the following generic fields. Data from these fields is used by Network Manager to model the EMS.

DataSource.id
Unique identifier for the datasource, in the form of an integer. This field takes the value 1, indicating that this is the primary data source.

DataSource.descr
Description of the EMS.

DataSource.emsHost
Hostname of the EMS.

DataSource.emsName
Name of the EMS.

DataSource.emsPort
Port of the EMS.

DataSource.emsVersion
Version of the EMS.

DataSource.emsIdentifier
Identifier for the EMS and key to integrate the Network Manager collector with the Netcool Configuration Manager driver.

DataSource.emsRole
Role of the EMS. This parameter can take one of the following values:

- unknown
- primary
- backup
- other

DataSource.emsStatus
Status of the EMS. This parameter can take one of the following values:

- unknown
- up
- down
- other

DataSource.emsUserName
Username of the EMS.

DataSource.emsPassword
Password of the EMS.

Note: The default port for Tellabs NIMA is 2462. The EMS identifier for Tellabs INM8000 must be set to `tellabsinm8000`. The default EMS username is `nbif` and the default EMS password is `tellabs8000`.

9. Configure the mapping properties to reflect the true state of the Tellabs interface state.

   The following parameters represent the mapping of the Network Manager SNMP variable object `ifOperStatus` to the Tellabs interface attribute `ifState`.

   - map.ifoperstatus.undefined
   - map.ifoperstatus.planned
   - map.ifoperstatus.installed
   - map.ifoperstatus.inuse

   The descriptions of attribute values for the Tellabs entity interface, Interface.ifstate, are as follows:

   - 0: Undefined
   - 1: Planned
   - 2: Installed
   - 3: In use

   The mapping for ifOperStatus is as follows:

   - 1: up
   - 2: down
   - 3: testing
   - 4: unknown
   - 5: dormant
   - 6: notPresent
   - 7: lowerLayerDown

   Default values for the mapping are as follows:

   - map.ifoperstatus.undefined=6
   - map.ifoperstatus.planned=3
   - map.ifoperstatus.installed=5
   - map.ifoperstatus.inuse=1

10. Fix Pack 2

    Configure other settings.

    use.name.alias
    Set to `true` if you want to use the name alias instead of the hardware type description. By default, this property is set to `false`.

    use.interface.ifmsg
    Set to true if you want to use the `IFMSG` attribute of the interface instead of the `IFHWTYPE` attribute. By default, this property is set to `false`.

    use.filter.node.discovery
    Set to true if you want to filter the nodes that are discovered by nodeId. You might want to use this option for testing whether two nodes and the connection between them are discovered. If you set this property to `true`, you must set two `node.id` properties. By default, this property is set to `false`

    node.id.n

    If you set `use.filter.node.discovery` to true, you must define two `node.id.n` properties. Only nodes with a nodeId matching one of these properties are discovered.

    For example, the following code extract defines two nodeIds to discover:

    ```
    use.filter.node.discovery=true
    node.id.1=874
    node.id.2=8856
    ```

11. Save the collector configuration file.

Java CSV collector properties file reference
Use this information to understand how the Java CSV collector properties file is constructed.

The location of the Java CSV collector properties file within `$NCHOME/precision/collectors/javaCollectors/` is listed in Table 48 on page 300. Using this `.properties` file you can map the network data to one or more files.

Note: Properties specified in the collector-specific properties file override any related properties specified in the generic Java collector properties file, `$NCHOME/precision/collectors/javaCollectors/framework/collector.properties`.

Table 48. Java CSV Collector properties file

Collector	Location of properties file within
Java CSV collector	`csv/csvcollector.properties`

Sample .properties file for the Java CSV collector

The following code fragment presents sample settings from a the Java CSV collector `.properties` file. This code fragment defines the directory where the CSV data files are located, and then proceeds to define formatting for CSV data in one of those files, the `devices.csv` file, which defines main entity data.

```
# Directory containing CSV data
CSVDir = ../csv/exampleCsvData/

# Main entity data (device data) is in devices.csv
MainEntityData.file = devices.csv

# The delimiter for the file is a |
MainEntityData.delimiter = \\|

# There are 6 columns of data in the file
MainEntityData.numCols = 6

# Only read lines that start with 10.1.1.
MainEntityData.lineMatch = 10\\.1\\.1\\..+

# Map the first data column to the device management IP address (<ip>)
MainEntityData.1.name = ManagementIpAddress
MainEntityData.1.mapsTo = DEVICE_MANAGEMENT_IP_ADDRESS

# Arbitrary mapping of extra information
# Map column 6 to <extraInfo><systemInfo>…</systemInfo></extraInfo>
MainEntityData.6.name = AdditionalSystemInfo
MainEntityData.6.mapsTo = EXTRA_INFO.systemInfo
```

Properties and mappings

The Java CSV collector properties files includes the following properties:

Base directory
For the Java CSV collector, the base directory is defined in the `CSVDir` property.

Data file properties
For the Java CSV collector, a set of properties are defined for the different CSV data files used as input.

The following table lists the properties held in CSV data files. For each property, *data_type* is one of the supported data types, listed in Table 51 on page 301.

Table 49. Data file properties

Property	Descriptions
data_type`.file`	File name containing the CSV data.

Table 49. Data file properties (continued)	
Property	**Descriptions**
data_type.delimiter	Delimiter for the data in the CSV file. The default delimiter is a comma `,`.
data_type.lineMatch	Regular expression syntax pattern. This instructs the collector to read only those lines that start with the pattern.
data_type.numCols	Number of columns of data in the file.
data_type.useCols	If not all columns are needed, this expression uses a comma separated list, for example, `1,3`, to indicate which columns of data to use from the file.

The following table how data in CSV data files is mapped to attributes.

Table 50. Data mappings	
Property	**Descriptions**
data_type.column_number.name	Human readable name for the column data.
data_type.column_number.description	Human readable description for the column data.
data_type.column_number.mapsTo	Mapping to an attribute supported for the data type.

Data types

The following table lists the supported data types.

Table 51. Data types	
Property	**Descriptions**
MainEntityData	Main entity (device) data.
InterfaceData	Interface data.
EntityData	Entity data in ENTITY-MIB format.
GenericEntityData	Entity data in non-ENTITY-MIB format.
L1ConnectivityData	Layer 1 connectivity data.
L2ConnectivityData	Layer 2 connectivity data.
L3ConnectivityData	Layer 3 connectivity data.
L2VpnData	Layer 2 VPN data.
L3VpnData	Layer 3 VPN data.
L3VpnInterfaceData	Layer 3 VPN interface data.
L3VpnRTData	Layer 3 VPN route target data.
LabelSwitchPathData	Label switch path data.
MplsInterfaceData	MPLS interface data.

Data source configuration

You can optionally configure details about the originating element management system (EMS) by configuring the fields listed in the following table. If you configure this EMS information , then data from these fields will be used by Network Manager to model the EMS.

Table 52. Data source configuration

Property	Descriptions
`DataSource.id`	This field takes the value 1, indicating that this is the primary data source.
`DataSource.descr`	Description of the EMS.
`DataSource.emsHost`	Hostname of the EMS.
`DataSource.emsName`	Name of the EMS.
`DataSource.emsVersion`	Version of the EMS
`DataSource.emsIdentifier`	Identifier for the EMS
`DataSource.emsRole`	Role of the EMS. Takes one of the following values: • unknown • primary • backup • other
`DataSource.emsStatus`	Role of the EMS. Takes one of the following values: • unknown • up • down • other

Configuring Perl collectors
Use this information to configure settings for collectors written in Perl.

Configuring the Nokia5529Idm collector
To use data from the Nokia5529Idm collector in a network discovery, you must configure the connection details between the EMS and Network Manager.

Procedure

1. Edit the collector configuration file:
   `$NCHOME/precision/collectors/perlCollectors/Alcatel5529IdmSoap/Alcatel5529IdmSoap Collector.cfg`
2. Edit the `General` section of the configuration file. Configure the following properties:

   Debug
   The collector debug mode. Set the property to `0` to turn off debug. Set the property to `1` to turn debug on. The collector prints debug to the display (stdout).

   Listen

   The port that the collector listens on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is `8081`. The port must match the port you have configured in the insert into the

collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.

Timeout
Timeout for the communication from the collector to Network Manager. The timeout is measured in seconds. The default value is 15 seconds.

The following example shows the default values of these properties:

```
General =>
{
        Debug => 0,
        Listen => 8081,
        Timeout => 15
},
```

3. Edit the `DataSource` section of the configuration file. Configure the following properties:

Host
The hostname of the EMS.

Port
The port to connect to the EMS.

Username
The username to connect to the EMS.

Password
The password to use to connect to the EMS.

Timeout
The timeout for the SOAP communication between the collector and the EMS.

Domain
The domain of the AMS system on which the Inventory Data Manager is running.

GetEntities
A flag for discovery of physical entities such as racks, cards, and ports. Set to 1 to discover physical entities. If this flag is set to 0, only chassis information is discovered. The default value is 1.

GetOnt
A flag to configure whether the collector retrieves Optical Network Terminal (ONT) information. Set to 1 to enable ONT module data retrieval. Retrieval of ONT information relies on physical entity information. Ensure that GetEntities is set to 1 if you want to set GetOnt to 1.

The following example shows example and default values of these properties:

```
DataSource =>
{
        Host => 192.168.1.2,
        Port => 8080

        Username => 'oss',
        Password => 'myPa55w0rd'

        Timeout => 30

             Domain => 'AMS'

            GetEntities => 1

             GetOnt => 0
,
```

4. Ensure that the `Batchsize` parameter is set to 500, unless otherwise advised by IBM Support. This parameter controls the size of each SOAP/XML response.
5. Save the collector configuration file.

Configuring the Alcatel5620Csv collector
To use data from the Alcatel5620Csv collector in a network discovery, you must configure the connection details between the EMS and Network Manager.

About this task

Procedure

1. Edit the collector configuration file:
   `$NCHOME/precision/collectors/perlCollectors/Alcatel5620SamCsv/Alcatel5620SamCsv Collector.cfg`
2. Edit the `General` section of the configuration file. Configure the following properties:

   Debug
   The collector debug mode. Set the property to `0` to turn off debug. Set the property to `1` to turn debug on. The collector prints debug to the display (stdout).

   Listen

   The port that the collector listens on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is `8081`. The port must match the port you have configured in the insert into the collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.

   Timeout
   Timeout for the communication from the collector to Network Manager. The timeout is measured in seconds. The default value is 15 seconds.

   The following example shows the default values of these properties:

```
General =>
{
        Debug => 0,
        Listen => 8081,
        Timeout => 15
},
```

3. Edit the `DataSource` section of the configuration file and specify the filename of the CSV loader module configuration file, as shown in the following example.

   The CSV loader module configuration file in turn specifies the CSV files and the `.drv` driver files that detail how each CSV file is parsed

```
DataSource =>
{
        CsvCfg => 'exampleCsv.cfg',
…
…
…
},
```

4. You can optionally configure details about the originating EMS by adding a `SourceInfo` subsection to the `DataSource` section of the configuration file. If you configure this EMS information , then data from these fields will be used by Network Manager to model the EMS.

   To configure details about the EMS, set values for the fields shown in the following code snippet:

```
        SourceInfo =>
        {
            Id => 1,
            Descr => 'Primary Data Source',
            # EmsHost => '',
            # EmsName => '',
            # EmsVersion => '',
            # EmsIdentifier => '',
            # EmsRole => '',
```

```
            # EmsStatus => '',
        },
```

5. Save the collector configuration file.

Configuring the Alcatel5620SamSoap collector
To use data from the Alcatel5620SamSoap collector in a network discovery, you must configure the connection details between the EMS and Network Manager.

About this task

You can also configure additional information to be collected from the EMS. To configure the Alcatel5620SamSoap collector, complete the following steps:

Procedure

1. Edit the collector configuration file:
   `$NCHOME/precision/collectors/perlCollectors/Alcatel5620SamSoap/Alcatel5620SamSoap Collector.cfg`
2. Edit the `General` section of the configuration file. Configure the following properties:

   Debug
   The collector debug mode. Set the property to `0` to turn off debug. Set the property to `1` to turn debug on. The collector prints debug to the display (stdout).

   Listen

   The port that the collector listens on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is `8081`. The port must match the port you have configured in the insert into the collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.

   Timeout
   Timeout for the communication from the collector to Network Manager. The timeout is measured in seconds. The default value is 15 seconds.

   The following example shows the default values of these properties:

```
General =>
{
        Debug => 0,
        Listen => 8081,
        Timeout => 15
},
```

3. Edit the `DataSource` section of the configuration file. Configure the following properties:

   Host
   The hostname of the EMS.

   Port
   The port to connect to the EMS.

   Username
   The username to connect to the EMS.

   Password
   The password to use to connect to the EMS.

   Timeout
   The timeout for the SOAP communication between the collector and the EMS.

The following example shows example and default values of these properties:

```
DataSource =>
{
        Host => 192.168.1.2,
        Port => 8080

        Username => 'oss',
        Password => 'myPa55w0rd',

        Timeout => 30,
…
…
…
},
```

4. Edit the `DataAcquisition` section of the configuration file and configure the following properties:

   GetEntities
   A flag for discovery of physical entities such as racks, cards, and ports. Set to 1 to discover physical entities. If this flag is set to 0, only the following information is discovered: chassis, logical entities, and data for the other enabled flags in the `DataAcquisition` section. The default value is 1.

   GetVplsVpns
   A flag for discovery of VPLS-based Layer 2 VPN data. Set to 1 to enable discovery of this data. The default value is 1.

   GetVllVpns
   A flag for discovery of VLL-based Layer 2 VPN data for epipes only. Set to 1 to enable discovery of this data. The default value is 1.

   GetLayer3Vpns
   A flag for discovery of Layer 3 VPN data. Set to 1 to enable discovery of this data. The default value is 1.

   GetMplsInterfaces
   A flag for discovery of MPLS interface data. Set to 1 to enable discovery of this data. The default value is 1.

   GetLayer2Connections
   A flag for discovery of physical link data. Set to 1 to enable discovery of this data. The default value is 1.

   The following example shows the default values of these properties:

```
DataAcquisition =>
{
        GetEntities => 1
        GetVplsVpns => 1,
        GetVllVpns => 1,
        GetLayer3Vpns => 1,
        GetMplsInterfaces => 1,
        GetLayer2Connections => 1
},
```

5. 5. Edit the `DataProcessing` section of the configuration file. Configure the ContainmentMethod property.

   The ContainmentMethod property controls how the entity data is processed in cases where containment is ambiguous due to missing or duplicate index data, which can occur with module(card)/slot data.

   The possible values of the ContainmentMethod property are:

   0
   Ignore duplicate indexes and prefer slots. Slot entities are stored, but module (card) entities might be lost if they share the same data as the slot.

   1
   Ignore duplicate indexes and prefer cards. Module entities are stored, but slot entities might be lost if they share the same data as the module.

2
Keep card and slot entities. Generates a false index if duplicates are encountered.

The default value is 2.

6. Optional: If you want to retrieve custom data from the EMS in addition to the data retrieved by default, complete the following steps.
   a) Create a new configuration file in the collector directory, or edit the default file $NCHOME/precision/collectors/perlCollectors/Alcatel5620SamSoap/extraInfo.cfg.
   b) Specify the data to be retrieved, as in the following example:

```
            Device =>
            {
                extraFields => [ { srcField => 'version', destField =>
'm_Version', typeField => 'string' }]
            },
```

   Where srcField is the name of the attribute in the SAM object, destField is the name of the field to which the data will be mapped within the extraInfo field, and typeField is an optional type descriptor.

   The attribute you want to retrieve must be part of one of the objects already retrieved by the collector. The objects queried by the collector are:

   - netw.NetworkElement
   - equipment.PhysicalPort
   - lag.Interface
   - equipment.MediaAdaptor
   - equipment.PhysicalPort
   - equipment.DaughterCard
   - equipment.Equipment
   - equipment.Shelf
   - vpls.L2AccessInterface
   - vll.L2AccessInterface
   - l3fwd.ServiceSite
   - vprn.L3AccessInterface
   - netw.PhysicalLink
   - lldp.RemotePeer.

   Valid types are int and string.
   c) Save and close the configuration file.
   d) Edit the CustomData section of the $NCHOME/precision/collectors/perlCollectors/Alcatel5620SamSoap/Alcatel5620SamSoap Collector.cfg collector configuration file. Specify the name and location of the configuration file that defines the extra information to be collected, as in the following example:

```
        CustomData =>
            {
            ExtraInfoCfg => 'extraInfo.cfg'
        },
```

7. Save the collector configuration file.

Configuring the Alcatel5620SamSoapFindToFile collector

To use data from the Alcatel5620SamSoapFindToFile collector in a network discovery, you must configure the connection details between the EMS and Network Manager, and the FTP details using which the XML files can be sent to the Network Manager server.

About this task

You can also configure additional information to be collected from the EMS. To configure the Alcatel5620SamSoapFindToFile collector, complete the following steps:

Procedure

1. Edit the collector configuration file:

   `$NCHOME/precision/collectors/perlCollectors/Alcatel5620SamSoap FindToFile/Alcatel5620SamSoap FindToFileCollector.cfg`

2. Edit the `General` section of the configuration file. Configure the following properties:

   Debug
   The collector debug mode. Set the property to `0` to turn off debug. Set the property to `1` to turn debug on. The collector prints debug to the display (stdout).

   Listen

   The port that the collector listens on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is `8081`. The port must match the port you have configured in the insert into the collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.

   Timeout
   Timeout for the communication from the collector to Network Manager. The timeout is measured in seconds. The default value is 15 seconds.

   The following example shows the default values of these properties:

```
General =>
{
        Debug => 0,
        Listen => 8081,
        Timeout => 15
},
```

3. Edit the `DataSource` section of the configuration file.

   a) Specify the hostname and port of the EMS, and the username and password to connect to the EMS, as shown in the following example:

```
DataSource =>
{
        Host => 192.168.1.2,
        Port => 8080

        Username => 'oss',
        Password => 'myPa55w0rd',
        Timeout => 30,
…
…
…
},
```

   b) Configure the following FTP parameters:

   UseSFTP
   Boolean flag to determine whether to use SSH FTP (SFTP) for transfer of XML files.

   - UseSFTP = 1: instructs the system to use SFTP.

- UseSFTP = 0: instructs the system to use FTP.

FtpUsername
The FTP username on the Network Manager server.

FtpPassword
The FTP password on the Network Manager server.

FtpHost
The IP address of the Network Manager server.

FtpDefaultDirectory
The default directory of the FTP service on the Network Manager server.

FtpDirectory
A user-defined directory for the FTP service on the Network Manager server. Leave this value blank if it is not used.

Tip: After a successful discovery, copy the generated XML files in the specified FTP directory to another location before performing a new discovery so that the XML files do not get overwritten.

4. Edit the `DataAcquisition` section of the configuration file and configure the following properties:

GetEntities
A flag for discovery of physical entities such as racks, cards, and ports. Set to 1 to discover physical entities. If this flag is set to 0, only the following information is discovered: chassis, logical entities, and data for the other enabled flags in the `DataAcquisition` section. The default value is 1.

GetVplsVpns
A flag for discovery of VPLS-based Layer 2 VPN data. Set to 1 to enable discovery of this data. The default value is 1.

GetVllVpns
A flag for discovery of VLL-based Layer 2 VPN data for epipes only. Set to 1 to enable discovery of this data. The default value is 1.

GetLayer3Vpns
A flag for discovery of Layer 3 VPN data. Set to 1 to enable discovery of this data. The default value is 1.

GetMplsInterfaces
A flag for discovery of MPLS interface data. Set to 1 to enable discovery of this data. The default value is 1.

GetLayer2Connections
A flag for discovery of physical link data. Set to 1 to enable discovery of this data. The default value is 1.

GetLteData
A flag for discovery of LTE data. Set to 1 to enable discovery of this data. The default value is 1.

The following example shows the default values of these properties:

```
DataAcquisition =>
{
        GetEntities => 1
        GetVplsVpns => 1,
        GetVllVpns => 1,
        GetLayer3Vpns => 1,
        GetMplsInterfaces => 1,
        GetLayer2Connections => 1,
        GetLteData => 1
}
```

5. Optional: If you want to retrieve custom data from the EMS in addition to the data retrieved by default, complete the following steps.

a) Create a configuration file in the collector directory, or edit the default file `$NCHOME/precision/collectors/perlCollectors/Alcatel5620SamSoap/extraInfo.cfg`.

b) Edit the file and specify the data to be retrieved, as in the following example:

```
            Device =>
            {
                extraFields => [ { srcField => 'version', destField =>
 'm_Version', typeField => 'string' }]
            },
```

Where `srcField` is the name of the attribute in the SAM object, `destField` is the name of the field to which the data will be mapped within the extraInfo field, and typeField is an optional type descriptor.

The attribute you want to retrieve must be part of one of the objects already retrieved by the collector. The objects queried by the collector are:

- netw.NetworkElement
- equipment.PhysicalPort
- lag.Interface
- equipment.MediaAdaptor
- equipment.PhysicalPort
- equipment.DaughterCard
- equipment.Equipment
- equipment.Shelf
- vpls.L2AccessInterface
- vll.L2AccessInterface
- l3fwd.ServiceSite
- vprn.L3AccessInterface
- netw.PhysicalLink
- lldp.RemotePeer.

Valid types are `int` and `string`.

c) Save and close the configuration file.

d) Edit the `CustomData` section of the `$NCHOME/precision/collectors/perlCollectors/Alcatel5620SamSoap/Alcatel5620SamSoap Collector.cfg` collector configuration file. Specify the name and location of the configuration file that defines the extra information to be collected, as in the following example:

```
        CustomData =>
            {
            ExtraInfoCfg => 'extraInfo.cfg'
        },
```

6. Save the collector configuration file.

Configuring the AlcatelNR8PLIOOISN collector
To use data from the AlcatelNR8PLIOOISN collector in a network discovery, you must configure the connection details between the EMS and Network Manager.

About this task

You can also configure additional information to be collected from the EMS. To configure the AlcatelNR8PLIOOISN collector, complete the following steps:

Procedure

1. Edit the collector configuration file:
   `$NCHOME/precision/collectors/perlCollectors/AlcatelNR8PLIooIsn/AlcatelNR8PLIooIsn Collector.cfg`

2. Edit the `General` section of the configuration file. Configure the following properties:

   Debug
   The collector debug mode. Set the property to `0` to turn off debug. Set the property to 1 to turn debug on. The collector prints debug to the display (stdout).

   Listen

   The port that the collector listens on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is `8081`. The port must match the port you have configured in the insert into the collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.

   Timeout
   Timeout for the communication from the collector to Network Manager. The timeout is measured in seconds. The default value is 15 seconds.

   The following example shows the default values of these properties:

```
General =>
{
        Debug => 0,
        Listen => 8081,
        Timeout => 15
},
```

3. Edit the `DataSource` section of the configuration file. The following default code in that section allows Network Manager to communicate with the ISN and the IOO agents running on the Alcatel NR8 PL EMS:

```
    DataSource =>
    {

        Timeout => 30,

        IOONBI =>
        {
            Host => '192.168.1.2',
            IOOAgentPort => 5001,
            Timeout => 10,
            IOOPassword => 'alcatel',
            IOOInputStream => 4096
        }

        ISNNBI =>
        {
            Host => '192.168.1.2',
            ISNAgentPort => 5002,
            ISNUserName => 'ISNTest1',
            ISNReportUnprocessedDirectory => './ISNReport',
            ISNReportProcessedDirectory => './ISNReportProcessed',
            FtpUsername => 'ftpuser',
            FtpPassword => 'ftpuserpassword',
            FtpSourceDirectory => '/tmp/report',
            GunzipPath => '/usr/bin/gunzip',
        }

        DataAcquisition =>
        {
            GetEntities => 1,
            GetLayer1Connections => 1
        }
    }
```

   Set GetEntities to 1 if you want to collect entity information from the collector.

   Set GetLayer1Connections to 1 if you want to retrieve layer 1 data from the collector.

4. You can optionally configure details about the originating EMS by adding a `SourceInfo` subsection to the `DataSource` section of the configuration file. If you configure this EMS information , then data from these fields will be used by Network Manager to model the EMS.

   To configure details about the EMS, set values for the fields shown in the following code snippet:

```
        SourceInfo =>
        {
            Id => 1,
            Descr => 'Primary Data Source',
            # EmsHost => '',
            # EmsName => '',
            # EmsVersion => '',
            # EmsIdentifier => '',
            # EmsRole => '',
            # EmsStatus => '',
        },
```

5. Save the collector configuration file.

Configuring the GenericCsv collector

To use data from the GenericCsv collector in a network discovery, you must configure the access details from the collector to the data source and configure the collector to Network Manager listening port. Configuring the collector to data source access details involves specifying how the collector loads and interprets CSV files.

Procedure

1. Edit the collector configuration file:

   `$NCHOME/precision/collectors/perlCollectors/GenericCsv/GenericCsv Collector.cfg`

2. Edit the `General` section of the configuration file. Configure the following properties:

   Debug
   The collector debug mode. Set the property to `0` to turn off debug. Set the property to `1` to turn debug on. The collector prints debug to the display (stdout).

   Listen

   The port that the collector listens on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is `8081`. The port must match the port you have configured in the insert into the collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.

   Timeout
   Timeout for the communication from the collector to Network Manager. The timeout is measured in seconds. The default value is 15 seconds.

   The following example shows the default values of these properties:

```
General =>
{
        Debug => 0,
        Listen => 8081,
        Timeout => 15
},
```

3. Edit the `DataSource` section of the configuration file and specify the filename of the CSV loader module configuration file, as shown in the following example.

   The CSV loader module configuration file in turn specifies the CSV files and the `.drv` driver files that detail how each CSV file is parsed

```
DataSource =>
{
        CsvCfg => 'exampleCsv.cfg',
…
…
…
},
```

4. Save the collector configuration file.

Configuring the HuaweiU2000Imanager collector
To use data from the HuaweiU2000Imanager collector in a network discovery, you must configure the connection details between the EMS and Network Manager.

Procedure

1. Edit the collector configuration file:
   `$NCHOME/precision/collectors/perlCollectors/HuaweiU2000iManagerTL1/HuaweiU2000iManagerTL1Collector.cfg`
2. Edit the `General` section of the configuration file. Configure the following properties:

   Debug
   The collector debug mode. Set the property to `0` to turn off debug. Set the property to 1 to turn debug on. The collector prints debug to the display (stdout).

   Listen

   The port that the collector listens on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is `8081`. The port must match the port you have configured in the insert into the collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.

   Timeout
   Timeout for the communication from the collector to Network Manager. The timeout is measured in seconds. The default value is 15 seconds.

   The following example shows the default values of these properties:

```
General =>
{
        Debug => 0,
        Listen => 8081,
        Timeout => 15
},
```

3. Edit the `DataSource` section of the configuration file. Specify the hostname and port of the EMS, and the username and password to connect to the EMS, as shown in the following example:

```
DataSource =>
{
        Host => 192.168.1.2,
        Port => 8080

        Username => 'oss',
        Password => 'myPa55w0rd'

        GetEntities => 1

           DataAcquisition =>
           {
               StoreONTs => 1,
           }
…
…
,
```

   Set GetEntities to 1 if you want to collect entity information from the collector.

   Set StoreONTs to 1 if you want to retrieve Optical Network Termination (ONT) data.
4. You can optionally configure details about the originating EMS by adding a `SourceInfo` subsection to the `DataSource` section of the configuration file. If you configure this EMS information , then data from these fields will be used by Network Manager to model the EMS.

   To configure details about the EMS, set values for the fields shown in the following code snippet:

```
        SourceInfo =>
        {
            Id => 1,
            Descr => 'Primary Data Source',
            # EmsHost => '',
            # EmsName => '',
            # EmsVersion => '',
            # EmsIdentifier => '',
            # EmsRole => '',
            # EmsStatus => '',
        },
```

5. Save the collector configuration file.

Configuring the HuaweiU2000iManagerTL1DumpExport collector
To use data from the HuaweiU2000iManagerTL1DumpExport collector in a network discovery, you must configure the connection details between the EMS and Network Manager.

About this task

The HuaweiU2000iManagerTL1DumpExport collector processes data from Huawei Access Devices using the Huawei U2000 EMS.

To get data for the collector, complete the following steps:

1. Export the XML file from the Huawei U2000 EMS by using the `DMP-INVENTORY` command. Ensure that the EMS parameter name `NBI_INVENTORY_DUMP_VERSION` is set to 3.
2. Check that the XML file is in the correct format. The collector supports XML files with ME, SHELF, CARD, PORT, SERVICEPORT and PORTOFVLAN object data, in DATAPACKET format. For reference, an example of an XML file used by the collector is provided in the following location:`$NCHOME/precision/collectors/perlCollectors/HuaweiU2000iManagerTL1DumpExport/data/sample.xml`.
3. Copy the XML file from the EMS to the server where the collector is installed. Copy the file to the location specified in the `FtpDestinationDirectory` parameter in the collector configuration file, as described below. The default directory is `/opt/IBM/netcool/core/precision/collectors/perlCollectors/HuaweiU2000iManagerTL1DumpExport/data`.

To configure the collector, complete the following steps.

Procedure

1. Edit the collector configuration file:
   `$NCHOME/precision/collectors/perlCollectors/HuaweiU2000iManagerTL1DumpExport/HuaweiU2000iManagerTL1DumpExportCollector.cfg`
2. Edit the `General` section of the configuration file. Configure the following properties:

   Debug
   The collector debug mode. Set the property to `0` to turn off debug. Set the property to 4 to turn debug on. Setting this property to 1, 2, or 3 is equivalent to setting it to `0`. The collector prints debug to the display (stdout).

   Listen

   The port that the collector listens on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is 8081. The port must match the port you have configured in the insert into the collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.

   Timeout
   Timeout for the communication from the collector to Network Manager. The timeout is measured in seconds. The default value is 15 seconds.

QueueSize
The size of the queue that holds XML-RPC requests received by the collector that are yet to be processed. If the queue is full, that is, if the collector is receiving requests faster than it can process them, then requests may be dropped. This results in error messages in the XML-RPC Helper log. The default value is 40.

The following example shows the default values of these properties:

```
General =>
{
        Debug => 0,
        Listen => 8081,
        Timeout => 15,
        QueueSize => 40
},
```

3. Edit the `DataSource` section of the configuration file. Specify the hostname and port of the EMS, the username and password to connect to the EMS, and the details of the XML file, as shown in the following example:

```
DataSource =>
{
        Host => 192.168.1.2,
        Port => 8080

        Username => 'oss',
        Password => 'myPa55w0rd'


        FtpDestinationDirectory =>
        '/opt/IBM/netcool/core/precision/collectors/perlCollectors/
HuaweiU2000iManagerTL1DumpExport/data',
        XMLDumpFileName =>
        'MA5610S.xml,MA5606T.xml',
        Timeout => 30,

           DataAcquisition =>
           {
               GetEntities => 1,
           }
…
…
,
```

   Set GetEntities to 1 if you want to collect entity information from the collector.

   Set `FtpDestinationDirectory` to the full directory path to the XML files that you transfer from the EMS. The default directory is `/opt/IBM/netcool/core/precision/collectors/perlCollectors/HuaweiU2000iManagerTL1DumpExport/data`.

   Set `XMLDumpFileName` to the file name of all the XML files from the EMS. If there are multiple XML files, use commas to separate the file names. For example, `fileName1.xml,fileName2.xml`. For single XML file names, the notation is `fileName1.xml`. The default value is with multiple XML file names: `MA5606TN2000.xml,MA5606T.xml`.

   Set `Timeout` for the timeout for the communication from the collector to EMS. The timeout is measured in seconds. The default value is 30 seconds.

4. You can optionally configure details about the originating EMS by adding a `SourceInfo` subsection to the `DataSource` section of the configuration file. If you configure this EMS information , then data from these fields will be used by Network Manager to model the EMS.

   To configure details about the EMS, set values for the fields shown in the following code snippet:

```
        SourceInfo =>
        {
            Id => 1,
            Descr => 'Primary Data Source',
            # EmsHost => '',
            # EmsName => '',
            # EmsVersion => '',
            # EmsIdentifier => '',
            # EmsRole => '',
```

```
            # EmsStatus => '',
        },
```

5. Save the collector configuration file.

Configuring the Optical Blackbox collector
To use data from the Optical Blackbox collector in a network discovery, you must configure the connection details between the EMS and Network Manager.

Procedure

1. Edit the collector configuration file:
   `$NCHOME/precision/collectors/perlCollectors/OpticalBlackboxXml/OpticalBlackboxXml Collector.cfg`.
2. Specify the port the collector must listen on for XML-RPC requests from Network Manager.

   This port is also used by the collector to provide XML-RPC responses to Network Manager. By default the port is `8081`. A default timeout of 15 seconds is also configured. Find and edit the `General` section of the configuration file, as shown in the following example:

```
General =>
{
        Debug => 0,
        Listen => 8081
        Timeout => 15
},
```

   The port must match the port you have configured in the insert into the collectorFinder.collectorRules table in the `DiscoCollectorFinderSeeds.cfg` file when seeding the collector for a first discovery.
3. Edit the `DataSource` section of the configuration file and specify the filename of the XML file, as shown in the following example:

```
DataSource =>
{
            BlackboxXml => './exampleXmlData/blackbox.xml',

            DataAcquisition =>
            {
                GetEntities => 1,
                GetLayer1Connections => 1
            }
}
```

4. You can optionally configure details about the originating EMS by adding a `SourceInfo` subsection to the `DataSource` section of the configuration file. If you configure this EMS information , then data from these fields will be used by Network Manager to model the EMS.

   To configure details about the EMS, set values for the fields shown in the following code snippet:

```
        SourceInfo =>
        {
            Id => 1,
            Descr => 'Primary Data Source',
            # EmsHost => '',
            # EmsName => '',
            # EmsVersion => '',
            # EmsIdentifier => '',
            # EmsRole => '',
            # EmsStatus => '',
        },
```

5. Save the collector configuration file.

Starting collectors

Before discovery starts, all the collectors must be running. You must start the collectors or make sure the collectors are running before starting a discovery that includes collectors.

Starting Java collectors

Use this information to start collectors written in Java.

About this task

You start a Java collector by going to the Java collector bin directory at `$NCHOME/precision/collectors/javaCollectors/bin` and issuing a command-line interface command. Issue the following command to start a Java collector (note that the command is entered on one line; options are explained in Table 53 on page 317):

```
collector.sh -jar [ -Xmsminimum_memory-sizem ] [ -Xmxmaximum_memory-sizem ]
jar_file -propsFile file_name -port port_number [ -bg ]
```

Table 53. Explanation of command-line options

Option	Explanation
`-jar` *jar_file*	Required: path to the jar file for the collector to be run relative to the root Java collector directory; for example, `csv/csvcollector.jar`.
`-propsFile` *file_name*	Optional: path to the properties file for the collector. The default is specified by the collector; for example, for the CSV collector the default is `../csvcollector.properties`.
`-port` *port_number*	Optional: port on which to run the collector. Default value is 8080. **Note:** This value can also be specified in a collector properties file.
`-bg`	Optional, and on UNIX only. Use this parameter to run the collector in the background. **Note:** Do not use the standard UNIX & parameter to run a collector in the background. Use the `-bg` option only.
`[ -Xms`*minimum_memory-size*`m ]`	Optional: the minimum heap size for the collector. Increase the heap size settings if you receive errors relating to the collector process being out of memory. The default minimum heap size is 256M.
`[ -Xmx`*maximum_memory-size*`m ]`	Optional: the maximum heap size for the collector. Increase the heap size settings if you receive errors relating to the collector process being out of memory. The default maximum heap size is 768M.

Example: starting the Java CSV collector

To start the Java CSV collector, perform the following steps:

1. Go to the following directory:

```
$NCHOME/precision/collectors/javaCollectors/bin
```

2. Run the following command to start the Java CSV collector with the default .jar and property files, on port 8089, with minimum heap size of 512M and maximum heap size of 1024M:

```
./collector.sh -Xms512m   -Xmx1024m  -jar csv/csvcollector.jar -propsFile
csv/csvcollector.properties  -port 8089 -bg
```

Stopping Java collectors
Use this information to stop collectors written in Java.

About this task

You stop a Java collector by going to the Java collector bin directory at $NCHOME/precision/collectors/javaCollectors/bin and issuing a command-line interface command. A Java collector can be stopped on a local machine or on a remote machine by issuing this command. Issue the following command to stop a Java collector (note that the command is entered on one line; options are explained in Table 54 on page 318):

```
shutdown_collector.sh -port port_number -host host_name
```

Table 54. Explanation of command-line options

Option	Explanation
-port *port_number*	Required: port on which the collector to be shut down is running.
-host *host_name*	Optional: hostname of the server on which the collector to be shut down is running. Default value is localhost.

Example: stopping a Java collector running on the local server on port 8080

To stop a Java collector running on the local server on port 8080, perform the following steps:

1. Go to the following directory:

```
$NCHOME/precision/collectors/javaCollectors/bin
```

2. Run the following command:

```
shutdown_collector.sh –port 8080
```

Starting Perl collectors
Use this information to start collectors written in Perl.

About this task

You start a collector by going to the relevant collector directory and issuing a command-line interface command. Issue the following command to start a collector (note that the command is entered on one line; options are explained in the table below):

```
ncp_perl collector_script -cfg  COLLECTOR_CONFIG_FILE
[ -csvcfg CSV_COLLECTOR_CONFIG_FILE ] [ -listen PRECISION_PORT ]
[ -debug DEBUG ] [ -logdir ] [ -nologdir DIRNAME ]
[ -help ] [ -version ]
```

Table 55. Explanation of command-line options

Option	Explanation
collector_script	The name of the perl script that implements the collector; for example, main.pl.

Table 55. Explanation of command-line options (continued)	
Option	**Explanation**
`-cfg COLLECTOR_CONFIG_FILE`	Specifies the collector configuration file.
`-csvcfg CSV_COLLECTOR_CONFIG_FILE`	Use this optional parameter to specify the name of a CSV file to use as a data source. You can also specify this parameter within the collector configuration file. **Restriction:** This parameter is valid only when the data source is a CSV file.
`-listen PRECISION_PORT`	An alternative method to specify the port on which the collector must listen for requests from Network Manager. Only specify a value here if no port value has been specified in the SOAP-based collector configuration file or in the CSV-based collector configuration file.
`-debug DEBUG`	The level of debugging output (1-4, where 4 represents the most detailed output).
`-logdir DIRNAME`	Directs log messages for each process started by CTRL to `NCHOME/log/precision`.
`-nologdir DIRNAME`	Directs log messages for each process started by CTRL to a separate file in the specified directory.
`-help`	All Network Manager components have a special `-help` option that displays the command line options. The component is not started even if –`help` is used in conjunction with other arguments.
`-version`	All Network Manager components have a special `-version` option that displays the version number of the component. The component is not started even if –`version` is used in conjunction with other arguments.

Seeding an EMS discovery

Seed the EMS discovery by seeding the Collector finder. This is typically a one-time setup task required when a new collector is added to your installation.

About this task

To enable Network Manager to find the collectors, you must seed the Collector finder. Seeding the Collector finder involves specifying for each collector:

- The hostname of the device on which the collector is running.

  Note: The hostname is not required when the collector is running on the same server as the Network Manager discovery.
- The port on that device on which the collector is listening.

If a collector is running on the same host as Network Manager, then you need only specify the port.

Note: If you are rediscovering a device using the Collector finder, then specify the IP address of the device or subnet to rediscover using the Discovery Configuration GUI.

You can seed the Collector finder to perform a discovery or to perform a partial rediscovery of a single device or subnet. If you seed the Collector finder to perform a partial rediscovery, then you can also specify a single device or subnet retrieved by the collector.

You must seed the Collector finder with the host name of the device on which the collector is running, and the port on that device on which the collector is listening. If the collector is running on the same host as Network Manager, then you need to specify only the port.

Seeding the Collector for a first discovery

You seed the Collector finder for a first discovery by appending an insert into the collectorFinder.collectorRules table to the DiscoCollectorFinderSeeds.cfg configuration file. The following insert seeds the Collector finder with a host name of 172.16.25.1, and a port of 8082. This insert means that the collector is running on a host with IP address 172.16.25.0, which is different to the host on which Network Manager is running. The override number of retries for this collector is 5.

```
insert into collectorFinder.collectorRules
(
        m_Host,
        m_Port,
        m_NumRetries
)
values
(
        "172.16.25.1",
        8082,
        5
);
```

Discovering aggregated links

As part of an EMS discovery, you can retrieve information about some links that are logically aggregated.

About this task

Network Manager can discover and model link aggregation information from devices running the Alcatel 5620 SAM EMS.

To discover aggregated link information, complete the following tasks:

Procedure

1. Configure the Alcatel 5620Sam Java collector.
2. Run a discovery.
3. Open the Structure Browser and examine a device that contains aggregated links.
   The aggregated links are shown within the device containment.

Results

A Service Affected Event (SAE) is generated if any port that participates in an aggregated link has an alert of severity 5 or higher.

Turning off isolated suppression for RCA

To ensure that RCA works correctly for EMS discoveries, you must turn off isolated suppression.

About this task

Network Manager applies an RCA model to the network that assumes a single location from which polling is carried out. In EMS discoveries, there are, effectively, multiple polling locations. In order for RCA to function correctly, you need to turn off the RCA rule known as isolated suppression.

To turn off isolated suppression, complete the following tasks:

Procedure

1. Edit the following file: $NCHOME/precision/eventGateway/stitchers/RCA/ProcessProblemEvent.stch
2. Comment out the following line:

```
numberOfSuppressedEvents = ExecuteStitcher('IsolatedEntitySuppression');
```

3. Save and close the file.
4. Restart the Event Gateway, ncp_g_event.

Adding passive entities to the network

You can use the *blackbox* service to define passive layer 1 entities in your network so that these unmanaged entities are represented in your network. Passive entities are user-defined entities that are additional entities that you want to see in your topology to have a complete picture of your network environment.

About this task

Restriction: You can only add layer 1 entities to the network using the blackbox service.

About passive entities

Passive layer 1 entities are user-defined entities that are additional entities that you want to see in your topology to have a complete picture of your layer 1 network environment. They are passive in that you cannot use Network Manager to discover or manage them, but you can still see them represented in topology maps.

You can use passive entities in various ways. For example:

- To show unmanaged nodes in your network.
- To show devices that are unmanaged by Network Manager. For example, a cable provider uses filters between network equipment to increase or modify signals. In many cases, these filters are not manageable, but they can be added to the blackbox service to show better granularity of the network.

Configuring the blackbox collector

You must perform one-time configuration tasks before you can use the blackbox collector to add passive entities to the network.

About this task

Table 56. Configuring the blackbox collector

Configuration task	Further information
Configure the blackbox collector. To do this, you must edit the blackbox collector configuration file, which is located at:$NCHOME/precision/collectors/perlCollectors/OpticalBlackboxXml/	“Configuring Perl collectors” on page 302
Seed the blackbox collector discovery by configuring the $NCHOME/etc/precision/DiscoCollectorFinderSeeds.cfg file for your domain to use same port as stated in the blackbox collector configuration file OpticalBlackboxXmlCollector.cfg.	“Seeding an EMS discovery” on page 319

Adding passive entities

Import data about passive devices and other entities by editing and loading the `blackbox.xml` file.

Editing the blackbox.xml file

Define passive devices and other entities by editing the `blackbox.xml` file.

About this task

Procedure

1. Go to `$NCHOME/precision/collectors/perlCollectors/OpticalBlackboxXml/exampleXmlData` directory.
2. Edit the `blackbox.xml` file and supply the data you want to import inside the <blackbox> element. Use the appropriate tags inside the <blackbox>element for the entity information you want to import.

   When editing the `blackbox.xml` file, note the following rules:
3. Save and close the file.

Structure of blackbox.xml

The `blackbox.xml` file is used to load passive (unmanageable) entities in your network so that you can view them in a topology map.

Use this example of a `blackbox.xml` file to understand the syntax of statements allowed in the file.

```
1]    <blackbox>
2]
3]     <ne name="Passive_NE1" ip="10.20.1.2" type="Passive">
4]           <properties>
5]                   <property name="company" value="IBM"/>
6]           </properties>
7]           <entities>
8]               <entity tom_id="rack" type="FLEX_Bay" position="1" />
9]               <entity tom_id="shelf" type="FLEX_shelf" position="1-1" />
10]              <entity tom_id="shelf" type="FLEX_shelf" position="1-2" />
11]              <entity tom_id="card" type="OC12Card" position="1-2-1" />
12]              <entity tom_id="card" type="TRMTR_Slot_HS" position="1-9-1" />
13]              <entity tom_id="ptp" type="OC12_LS_PTP_V" position="1-2-1-1" />
14]              <entity tom_id="ptp" type="OC48_LINE_PTP_TRMTR_HS" position=
"1-9-1-1" />
15]              <entity tom_id="ctp" type="STS3_OC48_TRMTR_HS" position=
"1-9-1-1-25" />
16]          </entities>
17]     </ne>
18]
19]     <ne name="Passive_NE2" ip="10.11.1.30" type="ECI">
20]          <properties>
21]                  <property name="Vendor" value="ECI"/>
22]                  <property name="NeType" value="XDM100"/>
23]                  <property name="LocationName" value="MELAKA"/>
24]                  <property name="Address" value="THIRD FLOOR , COLLEGE ,
MELAKA , , Malaysia"/>
25]                  <property name="City" value="MELAKA"/>
26]                  <property name="State" value="ALOR GAJAH"/>
27]          </properties>
28]          <entities>
29]                  <entity type="FLEXPort" tom_id="ptp" label="PORT" position=
"1-1-3021-1" />
30]                  <entity type="FLEXPort" tom_id="ptp" label="PORT" position=
"1-1-3022-1" />
31]          </entities>
32]     </ne>
33]
34]     <ne name="Passive_NE3" ip="10.11.1.29" type="ECI">
35]          <properties>
36]                  <property name="Vendor" value="ECI"/>
37]                  <property name="NeType" value="XDM100"/>
38]                  <property name="LocationName" value="ARAU KANGAR"/>
39]                  <property name="Address" value="3 Ground , Sate Road ,
Arau, Kangar "/>
40]                  <property name="City" value="KANGAR"/>
41]                  <property name="State" value="PERLIS"/>
42]          </properties>
```

```
43]           <entities>
44]                   <entity type="FLEXPort" tom_id="ptp" label="PORT" position=
"1-1-3036-2" />
45]                   <entity type="FLEXPort" tom_id="ptp" label="PORT" position=
"1-1-3033-1" />
46]           </entities>
47]     </ne>
48]
49]     <tc>
50]           <aend ne="Passive_NE2" tp="1-1-3021-1"/>
51]           <zend ne="Passive_NE1" tp="1-9-1-1-25"/>
52]     </tc>
53]
54]     <tc>
55]           <aend ne="Passive_NE3" tp="1-1-3036-2"/>
56]           <zend ne="Passive_NE2" tp="1-1-3022-1"/>
57]     </tc>
58]
59]   </blackbox>
```

The table below describes this query.

Table 57. Example of a `blackbox.xml` file

Line numbers	Element	Description and attributes
1-59	<blackbox>	Blackbox. Does not take attributes.
3-17 19-32 34-47	<ne>	Network element. Takes the following attributes: **name** (Required) The name of the network element. Example: name="Passive_NE1". **type** (Required) A value of type ="Passive" indicates a passive entity. **ip** (Optional) IP address of the passive entity.
4-6 20-27 35-42	<properties>	The properties of the entity to which the element applies.
5 21-26 36-41	<property>	A property of the entity. The entity might be a network element or topological connection. **name** (Required) Name of the property. Example: name="company". **value** (Required) The property value. Example: value="IBM".
7-16 28-31 43-46	<entities>	The entities within a network element.

Table 57. Example of a `blackbox.xml` file (continued)

Line numbers	Element	Description and attributes
8-15 29-30 44-45	`<entity>`	An entity within a network element. **label** (Optional) Label for this entity. **position** (Required) The relative location of the entity within the network element. Example: position="1-2-1-1" means the first rack - second shelf - first card, first PTP. As a best practice, list the entity statements in a descending hierarchical order. **tom_id** (Required) The Telecom Object Model ID. Example: `tom_id="rack"`. Valid values are: • rack • shelf • card • ptp (physical termination point) • ctp (connection termination point) **type** (Optional) The type of entity. Example: `type="FLEX_Bay"`.
49-52 54-57	`<tc>`	A topological connection.
50 55	`<aend>`	The notional start of a topological connection. **domain** (Optional) The name of the domain to which the endpoint belongs (the domain name of the PTP and the network element to which the PTP belongs). You can use this attribute to specify that one of the endpoints resides in some other network domain. If the remote network element exists in the database, a topological connection is created. If it does not exist in the database, no topological connection is created. **ne** (Required) The name of the network element. **tp** (Required) The relative location of the termination point of the endpoint in the topological connection. Example: tp="1-1-3021-1" means the first rack - first shelf - card 3021 - first termination point.

Table 57. Example of a blackbox.xml file (continued)		
Line numbers	**Element**	**Description and attributes**
51 56	<zend>	The notional end of a topological connection. **domain** (Optional) The name of the domain to which the endpoint belongs (the domain name of the PTP and the network element to which the PTP belongs). You can use this attribute to specify that one of the endpoints resides in some other network domain. If the remote network element exists in the database, a topological connection is created. If it does not exist in the database, no topological connection is created. **ne** (Required) The name of the network element. **tp** (Required) The relative location of the termination point of the endpoint in the topological connection. Example: tp="1-1-3036-2" means the first rack - first shelf - card 3036 - second termination point.

Loading the blackbox.xml file
Start the OpticalBlackboxXml collector to load the blackbox.xml file and import the passive entities into the topology.

Before you begin
Make sure that there is no discovery currently running. However, the Discovery engine `ncp_disco` can be running.

About this task
Start the OpticalBlackboxXml collector in the background by issuing the following command:

```
ncp_perl main.pl -cfg OpticalBlackboxXmlCollector.cfg &
```

Collector Developer Guide

Advanced users can use the Collector Developer Guide to create custom discovery collectors.

The Collector Developer Guide provides reference information to support creating and troubleshooting your own discovery collector.

Access the Collector Developer Guide, and the Perl and Java Collector Framework documentation, at this URL: https://www.ibm.com/support/pages/announcing-nm-v42-collector-developer-guide-v17

Configuring MPLS discoveries

Configure an MPLS discovery to discover core MPLS networks and the VPNs that use these core networks. Advanced MPLS discovery configuration provides extra customization facilities.

About MPLS discovery

Administrators within service providers who offer Multiprotocol Label Switching (MPLS) VPN services can discover MPLS core networks and MPLS VPNs to enable NOCs within the service provider to monitor the health of customer VPNs.

Network Manager supports the discovery of the following VPNs running across MPLS core networks:

- Layer 3 VPNs

- Enhanced Layer 2 VPNs

For the enhanced Layer 2 VPNs, Network Manager discovers point-to-point pseudowires linking two provider edge (PE) routers.

The following sections specify the terminology and topology visualization conventions used in Network Manager to refer to MPLS networks.

Note: The graphics shown in this section are conceptual representations of an MPLS network only. You cannot see these conceptual views in the Network Views graphical user interface (GUI).

Layer 3 MPLS VPNs

Network Manager can visualize layer 3 MPLS VPN topologies in a core view or an edge view.

The core view and edge view differ as follows:

- The core view shows the provider-edge (PE) routers, and provides visibility of the provider core (P) routers and label switched path (LSP) data within the MPLS core, for each of the VPNs running across the MPLS core.
- The edge view shows the PE routers and the MPLS cloud only. It does not provide visibility of the devices in the core.

Enhanced Layer 2 MPLS VPNs

For enhanced Layer 2 VPNs, Network Manager provides only an edge view of your MPLS core network.

Network Manager displays an enhanced Layer 2 VPN as a collection of point-to-point pseudowires. This means that if an enhanced Layer 2 VPN contains more than two provider edge (PE) routers, then Network Manager displays that VPN in multiple views, each view consisting of a single PE to PE point-to-point connection.

Table 58 on page 326 shows examples of enhanced Layer 2 VPNs with two and more than two PEs. The table also provides the number of pseudowires, and hence the number of views that Network Manager displays for each VPN.

Table 58. Number of pseudowires for an enhanced Layer 2 VPN

Number of PEs in an Enhanced Layer 2 VPN	Number of point-to-point pseudowires	Number of Views Network Manager displays for this VPN
2	1	1
3	3	3
4	6	6

Standard and advanced MPLS discovery configuration

Configure a standard MPLS discovery to discover all of your MPLS network and uses default naming convention for the VPNs discovered. The standard MPLS discovery configuration also enables the display of service-affected events (SAEs) in the **Event Viewer**. Advanced MPLS discovery configuration provides extra customization facilities.

Configuration activities for an MPLS network include seeding, scoping, and the other standard discovery activities.

Standard and advanced MPLS discovery configuration differ as follows:

- Standard MPLS discovery: discovers all of your MPLS network and uses default naming convention for the VPNs discovered
- Advanced MPLS discovery: using the advanced configuration options you can:
  - Restrict the scope of the discovery to a particular VPN or VRF

- Configure your own VPN naming conventions
- Add label data for MPLS discoveries

After you have configured and run an MPLS discovery, your operators can monitor customer VPNs in the following ways:

- View topology maps of selected VPNs, showing the alert status of the VPNs and of the devices in the VPNs .
- Identify service-affected events (SAEs) in the **Event Viewer**. An SAE is an alert that warns operators that a critical customer service, for example, a customer VPN, has been affected by one or more network events. The underlying network events are on an interface on either a PE router or a CE router.

About Service Affected Events

A Service Affected Event (SAE) alert warns operators that a critical customer service has been affected by one or more network events.

An SAE is produced when one or more events occur on a Provider Edge (PE) or Customer Edge (CE) interface in a Virtual Private Network (VPN) or Virtual Private LAN Service (VPLS). The underlying network events are on an interface of a PE router or a CE router, or on the link between them. You must configure the MPLS discovery to infer the existence of CE routers so all possible SAEs are generated for your customer VPNs.

The following list gives two examples of SAE events generated on two different customer VPNs:

- SAE generated on customer-1 VPN because of an `Mpls VRF Down` trap on a PE router interface
- SAE generated on customer-3 VPN because of a `LinkDown` trap on a CE router interface

Each SAE appears as an alert in the **Event Viewer**. The appearance of the SAE warns operators that a customer VPN has been affected, possibly critically, by one or more network events. Operators can right-click the SAE and issue a command to view the underlying events that caused the SAE.

For more information about the **Event Viewer**, see the *IBM Tivoli Netcool/OMNIbus Web GUI Administration and User's Guide*.

Configuring standard MPLS discovery

Configure an MPLS discovery to discover core MPLS networks and the VPNs that use these core networks.

About this task

In addition to the standard discovery configuration activities, you must perform some MPLS-specific discovery configuration activities:

- Configure MPLS agents
- Optional: set VPNs to be named by Route Targets rather than by VRFs
- Configure SNP and Telnet to ensure that the agents can access network devices
- Configure Network Manager to infer the existence of CE routers. This step is necessary to enable operators to view service-affected events in the **Event Viewer**.

These EMS-specific discovery configuration activities are described in the following topics.

Configuring MPLS agents

As part of MPLS discovery configuration you must enable one or more MPLS agents. You can also resolve the problem of duplicate IP addresses in different Virtual Private Networks (VPNs) by configuring the AsAgent agent.

About this task

The following MPLS agents and the corresponding agent definition (.agnt) files are provided:

Procedure

- Juniper Telnet agent (JuniperMPLSTelnet.agnt)
- Juniper ERX router agent (UnisphereMPLSTelnet.agnt)
- Cisco MPLS Telnet agent (CiscoMPLSTelnet.agnt)
- Cisco MPLS SNMP agent (CiscoMPLSSnmp.agnt)
- Laurel MPLS Telnet agent (LaurelMPLSTelnet.agnt)

  Note: The Laurel MPLS Telnet agent is intended for RT- (RouteTarget) based discoveries only.

Results

These agents can discover MPLS VPN and Virtual Private LAN Service (VPLS) data from devices in the network.

Tip: Agents that retrieve VPLS information can retrieve large amounts of data. Enabling these agents can add significant processing time to the discovery process. If you do not need to rediscover VPLS information, disable these agents for a faster discovery.

Note: If you have an MPLS network that supports both Layer 3 and enhanced Layer 2 VPNs, then the same MPLS agents discover both types of VPN. Network Views can also partition both Layer 3 and enhanced Layer 2 VPNs simultaneously on the same core MPLS network.

If your MPLS network contains Cisco equipment, enable both the Cisco MPLS Telnet and Cisco MPLS SNMP agents. These two agents complement each other, as follows:

- Cisco MPLS SNMP agent targets only devices with an Internetwork Operating System (IOS) that fully supports SNMP-based MPLS discovery
- CiscoMPLSTelnet agent targets only devices running an IOS that does not fully support an SNMP-based discovery



Attention: Use caution when altering the CiscMPLSSnmp.agnt file. Some network devices might contain IOS versions that have a flaw that might affect the device when certain MPLS SNMP data is requested. These IOS versions have been filtered out by default in the CiscMPLSSnmp.agnt file.

In addition to these standard discovery configuration activities, you can also change the scope of the MPLS discovery by restricting the scope to specific VPNs or VRFs.

Configuring MPLS Telnet agents
The CiscoMPLSTelnet, JuniperMPLSTelnet, LaurelMPLSTelnet, and UnisphereMPLSTelnet agents obtain data from devices primarily through Telnet. You must enable these agents and configure Telnet access to ensure that MPLS Telnet agents can access devices and can understand the output from devices.

About this task

Do the following to configure Telnet access for MPLS Telnet agents:

Procedure

1. Populate the Telnet configuration file `TelnetStackPasswords.cfg` so the agents can access the target devices.
2. Configure the Telnet Helper so that agents can understand the output from devices.

Configuring MPLS SNMP agents
The CiscoMPLSSnmp agent obtains data from devices using SNMP. You must enable this agent and configure SNMP access to ensure that this agent can access devices and can understand the output from devices.

About this task

To configure SNMP access for MPLS SNMP agents:

Note: The CiscoMPLSSnmp.agnt attempts to retrieve the L2 VPNs using the telnet 'show' commands if the agent fails to retrieve the data via SNMP.

Procedure

1. Configure SNMP access to devices.
2. Configure the SNMP Helper so that agents can understand the output from devices.

Configuring the AsAgent agent
To resolve the problem of duplicate IP addresses in different VPNs, activate the AsAgent agent and provide Network Manager with a mapping file, `ASMap.txt`, that contains a complete list of devices in each VPN, together with an `AddressSpace` tag, which defines which VPN the device belongs to.

About this task

During an MPLS discovery, Network Manager might discover devices in different VPNs with identical IP addresses. In this case, Network Manager cannot differentiate between these devices and might resolve device connectivity incorrectly. The devices in question might be the CE routers at the edge of the VPNs, or might be devices within the VPNs.

In the `ASMap.txt` mapping file, provide a complete list of devices in each VPN, together with an `AddressSpace` tag, which defines which VPN the device belongs to.

Table 59 on page 329 provides a description of the AsAgent agent that you need to activate to resolve the problem of duplicate IP addresses.

Table 59. AsAgent agent

Agent name	Function
AsAgent	Enables Network Manager to uniquely identify devices in different VPNs with identical IP addresses, and thereby correctly resolve device connectivity. This agent works in conjunction with the `ASRetprocessing.stch` stitcher and with the `ASMap.txt` file in `NCHOME/precision/etc`.

Table 60 on page 329 provides the format of the `ASMap.txt` file by showing an example of the content of this file. Fields in this text file must be separated by tabs.

Table 60. Format of ASMap.txt file

Unique IP Address	Address Space	IP Address
192.168.0.1	CUSTOMER-1	192.168.2.1
192.168.0.2	CUSTOMER-1	192.168.2.21
192.168.0.3	CUSTOMER-1	192.168.2.22
192.168.0.4	CUSTOMER-2	192.168.2.1
192.168.0.5	CUSTOMER-2	192.168.2.31
192.168.0.6	CUSTOMER-2	192.168.2.32

Using RT values instead of VRF names

By default, VPNs are named based on the familiar VRF names. You can choose to use Route Targets instead to name VPNs.

Before you begin

Important:

If devices have previously been discovered with VRF naming, duplicate VPN entities might be displayed on the next discovery. For example, the same VPN entity can appear twice, once with the VRF name and once with the RT value. To avoid duplicate device entries, set the LingerTime of all devices in the topology to zero before running your next discovery. To do this, proceed as follows:

1. Log into the OQL Service Provider using the following command:

   ```
   ncp_oql -domain NCOMS -service Model
   ```

2. Run the following command to set the LingerTime to zero:

   ```
   update ncimCache.lingerTime set lingerTime = {LINGERTIME=0};
   go
   ```

About this task

For MPLS networks, Network Manager uses Route Target (RT)-based discovery, using VRF and RT information to determine which provider edge routers are involved in a VPN. The core view consists of all MPLS-enabled devices, and VPNs resolve based on RT information.

To use RT values instead of VRF names:

Procedure

1. Exit all instances of the **Discovery Configuration** GUI.
2. Go to the `NCHOME/etc/precision` directory.
3. Edit the `DiscoConfig.`*`DomainName`*`.cfg` file and set the **`m_RTVPNResolution`** field to 1 in the disco.config table. The default is 2 (use VRF).
4. Restart the ncp processes to read the configuration files again:

   ```
   itnm_stop ncp
   itnm_start ncp
   ```

   Alternatively, restart the ncp_config process.

What to do next

Note: RT-based discoveries do not require label data. However, you can retrieve the label data as described in “Fine-tuning label data” on page 335.

Inferring the existence of CE routers

You can infer the existence of your customers’ CE routers by making specifications in the advanced discovery configuration options within the Discovery Configuration GUI.

About this task

If the host on which Network Manager is installed has no access to your customers’ CE routers, then Network Manager cannot discover these routers directly. This situation typically occurs when the company providing MPLS services owns the PE routers but has no access to CE routers, which are owned by the customers running the VPNs.

Note: This situation does not occur if the company providing MPLS services owns and manages both PE and CE routers and therefore has access to both sets of devices.

To infer the existence of your customers' CE routers, specify this in the advanced discovery configuration options within the Discovery Configuration GUI.

Note: You should do this only where the PE interface is on a /30 subnet. In this case, the other device on the subnet must be the CE router, and the IP address of the CE will be the other address on the /30 subnet.

Limitations on inferring CE routers:

- Avoid inferring the existence of CE routers if your PE routers are connected to the CE routers by serial links and you know that there is IP address duplication among the CE routers and devices within the MPLS core network. Network Manager will remove from the topology any discovered MPLS core routers that have the same IP address as an inferred CE IP address.
- If your PE routers are connected to the CE routers by Ethernet, you can infer the existence of CE routers without needing to perform any other checks. In this case, Network Manager can determine the MAC address of the CE router. If Network Manager has discovered another device with the same MAC address, then it must be the CE router. In this case, Network Manager uses the discovered device data and does not infer the existence of the CE.

Configuring advanced MPLS discovery

Configure an advanced MPLS discovery for extra customization facilities not included in the standard MPLS discovery.

About this task

When you configure an advanced MPLS discovery, you must perform the following activities in addition to the activities required for a standard MPLS discovery.

- Define the scope of the MPLS discovery: enables you to restrict the scope of this discovery to a particular VPN or VRF.
- Specify a VPN name: enables you to configure your own VPN naming convention
- Fine-tune label data discovery: enables you to manually retrieve label data for MPLS discoveries

Configuring discovery of MPLS Traffic Engineered tunnels

To discover MPLS Traffic Engineered tunnels, enable the StandardMPLSTE agent, configure the information that is retrieved, and configure the scope of the discovery.

MPLS Traffic Engineered tunnel discovery modes

Set the discovery mode according to how much detail you want to retrieve.

A mode-switch is provided in the discovery agent configuration file that configures specific tunnel instances, which can be wildcarded, to retrieve different amounts of tunnel data. You can choose any of the following modes.

HeadEndHops (default)

In HeadEndHops mode, the agent retrieves head-end and tail-end of the tunnel, and the transit LSRs and next-hop interfaces are identified by querying the head-end LSR for computed and actual-route hop data. The actual and computed route data is retrieved from the mplsTunnelARHopTable and mplsTunnelCHopTable MIB tables respectively. This discovery mode does not store transit and tail-end tunnel instances against transit and tail-end LSRs. A connection is created in the MPLS TE topology between the head-end and tail-end LSR interfaces via transit device hops (if present) which are associated with the head-end LSR tunnel object for the appropriate tunnel interface.

MPLS cross-connect pointers that are discovered and resolved on the head tunnel will be resolved to the appropriate LSP ID where possible.

You can use this information to determine if the actual path taken by a tunnel is different to the path computed by the Compute Shortest Path First (CSPF) calculations. You can see the computed and

actual path, although there is no way to determine that an LSR is acting in a transit or tail capacity without looking at the head-end LSR tunnel data.

Note: Actual route data is only available if the Record Route Option (RRO) has been specified for the tunnel instance.

In the schema of the scope.mplsTe table, the HeadEndHops mode maps to value 1 of m_Mode.

HeadTailEnd

In HeadTailEnd mode, only MPLS TE tunnel head-end and tail-end points are resolved, by querying the head-end Label Switching Router (LSR). This mode provides the minimal amount of information about the MPLS TE tunnels. A connection in the MPLS TE topology is created between the head-end and tail-end LSR interfaces. A tunnel resource instance is associated with the head-end tunnel LSR entity.

In this mode, you cannot identify the transit LSRs, and computed and actual route data is not retrieved.

MPLS cross-connect pointers that are discovered and resolved on the head tunnel will be resolved to the appropriate LSP ID where possible.

In the schema of the scope.mplsTe table, the HeadTailEnd mode maps to value 2 of m_Mode.

AllLSRTunnelsAndHops

In AllLSRTunnelsAndHops mode, the agent retrieves the head-end and tail-end of the tunnel and identifies transit LSRs and next-hop interfaces by querying the head-end LSR for computed and actual route hop data. The actual and computed route data is retrieved from the mplsTunnelARHopTable and mplsTunnelCHopTable MIB tables respectively. This discovery mode stores transit and tail-end tunnel instances against transit and tail-end LSRs. The mode creates a connection in the MPLS TE topology between the head-end and tail-end LSR interfaces that are associated with the head-end (for the tunnel interface) and transit and tail-end LSR tunnel objects. Computed and actual-route connections are associated with Computed and Actual connection entity types, which are aggregated in sequence from the head-end LSR tunnel entity. A tunnel resource instance is associated with the head-end tunnel LSR entity.

You can use this information to determine if the actual path taken by a tunnel is different to the path computed by the CSPF calculations. You can see the computed and actual path and determine the transit or tail-end role of an LSR without looking at the headend LSR tunnel instance.

Note: Actual route data is only available if the Record Route Option (RRO) has been specified for the tunnel instance.

MPLS cross-connect pointers that are discovered and resolved on the head tunnel will be resolved to the appropriate LSP ID where possible.

In the schema of the scope.mplsTe table, the AllLSRTunnelsAndHops mode maps to value 3 of m_Mode.

Related tasks

Enabling the StandardMPLSTE agent
To discover MPLS TE tunnels, you must enable the StandardMPLSTE agent and add the relevant SNMP community strings.

Configuring the StandardMPLSTE agent
Configure which tunnels to discover, and what details to retrieve.

Enabling the StandardMPLSTE agent
To discover MPLS TE tunnels, you must enable the StandardMPLSTE agent and add the relevant SNMP community strings.

About this task

To enable the StandardMPLSTE agent, complete the following steps.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Configuration**.
2. From the **Domain** list, select the required domain.
3. Click the **Full Discovery Agents** tab.
   The **Agents List** is displayed, showing all available discovery agents for the selected discovery option.
4. Select the check box next to the StandardMPLSTE agent.
5. Click **Save**  .
6. If you want to rediscover MPLS TE tunnels, enable the StandardMPLSTE agent for partial rediscoveries.
   a) Click the **Partial Rediscovery Agents** tab.
   b) Select the check box next to the StandardMPLSTE agent.
   c) Click **Save**  .
7. Ensure that the SNMP community strings are configured correctly to access the devices in the MPLS TE tunnels.

Related concepts

MPLS Traffic Engineered tunnel discovery modes
Set the discovery mode according to how much detail you want to retrieve.

Related tasks

Configuring the StandardMPLSTE agent
Configure which tunnels to discover, and what details to retrieve.

Configuring the StandardMPLSTE agent
Configure which tunnels to discover, and what details to retrieve.

About this task

To configure the StandardMPLSTE agent, complete the following steps.

Procedure

1. Back up and edit the file `NCHOME/etc/precision/DiscoScope.cfg`.
2. Locate and edit the insert into the scope.mplsTe table, or create a new insert. Create or edit an insert similar to the following:

```
insert into scope.mplsTe
(
      m_Protocol,
   m_Zones,
   m_Mode,
   m_TunnelFilter
)
values
(
   1,
   [{m_Subnet = '192.168.1.0', m_NetMask = 24 }],
   2,
   1
);
```

   This insert configures the agent to behave in the following way:

   - It uses IPv4.
   - It includes (`m_Tunnelfilter=1`) the subnet 192.168.1.* in the discovery of tunnel heads.
   - It retrieves data for the head and tail of the tunnel but not for the transit routers.
3. Save and close the file.
4. Stop and restart the discovery engine, the **`ncp_disco`** process, for your configuration changes to take effect.

Related concepts

MPLS Traffic Engineered tunnel discovery modes
Set the discovery mode according to how much detail you want to retrieve.

Related tasks

Enabling the StandardMPLSTE agent
To discover MPLS TE tunnels, you must enable the StandardMPLSTE agent and add the relevant SNMP community strings.

Defining the scope of an MPLS/VPN discovery

When configuring the discovery of one or more Virtual Private Networks (VPNs) running across an MPLS core, you can restrict the scope of this discovery to a particular VPN name or VPN Routing and Forwarding (VRF) table name.

About this task

You restrict the scope by configuring the optional DiscoAgentDiscoveryScoping section in the *.agnt file. The configurable options are described in Table 61 on page 334.

Table 61. Defining MPLS scoping requirements

Option	Function
IncludeVRF	Allows the discovery of the named VRF
IncludeVPN	Allows the discovery of the named VPN
ExcludeVPN	Does not discover any VRFs within the named VPN
ExcludeVRF	Does not discover the specified VRF

The order of precedence for `Exclude` and `Include` within the DiscoAgentDiscoveryScoping section is:

1. Exclude
2. Include

The order of precedence for VRF and VPN within the DiscoAgentDiscoveryScoping is:

1. VRF
2. VPN

For example, if you include a VPN, but another filter excludes a VRF in your VPN, then the VRF is excluded. If a VPN is excluded, but another filter includes a VRF within that VPN, then the VRF is included.

VRF names are case sensitive and an asterisk (*) represents a wildcard for any VRF or VPN name when used in the name part of the configuration. The wildcard can be used with any of the above options.

Scoping by VPN name works only when the VRF names configured on the devices discovered by the MPLS agents are in the Cisco-recommended VRF format. A VRF is named based on the VPN or VPNs serviced, and the topology type. The format for the VRF names are:

```
V [number assigned to make the VRF name unique]: [VPN_name]
```

For example, in a VPN called `precision`, a VRF for a hub edge router would be:

```
V1:precision
```

A VRF for a spoke edge router in the `precision` VPN would be:

```
V1:precision-s
```

A VRF for an extranet VPN topology in the `precision` VPN would be:

```
V1:precision-etc
```

The following example scopes a discovery in a system where there are four VRFs: `V65:Precision-etc`, `V65:Precision-s`, `V65:Precision`, and `V44:AcmeSheds`.

```
//2 VRFs are to be included
//
DiscoAgentDiscoveryScoping
{
    IncludeVRF = “V65:Precision-etc”;
    IncludeVRF = “V44:AcmeSheds”;
}
//All 4 VRFs are to be included
//
DiscoAgentDiscoveryScoping
{
    IncludeVPN = “Precision”;
    IncludeVRF = “V44:AcmeSheds”;
}
```

Fine-tuning label data

The MPLS agents used to discover MPLS networks do not retrieve label data by default. However, you can set the agents to manually retrieve label data.

Results

You can retrieve label data manually with the following insert in the DiscoAgentDiscoveryScoping section of the appropriate MPLS agent's `.agnt` file:

```
DiscoAgentDiscoveryScoping
{
    GetMPLSLabelData = 1;
}
```

Note: Not all MPLS agents support this setting. Check the description of the agent for information about whether the specific agent supports this setting. If supported, the agent retrieves label data in addition to the data used for MPLS discovery. However, the additional label data is not used by any default stitchers and is not stored in NCIM. Using this option to retrieve label data requires custom data collection with custom stitchers set up to consume the label data. For more information, see Chapter 18, “Discovering and using custom data,” on page 417. For help with writing custom stitchers, contact IBM Support.

Fix Pack 2 *Configuring discovery of Huawei VPNs*

To visualize VPN information from Huawei devices, you must configure the discovery.

About this task

Network Manager can discover and visualize User-facing Provider Edge (UPE) and Network Provider Edge (NPE) devices with pseudowire interfaces that are part of a Huawei VPN. It can also build Virtual Switch Interface (VSI) instances for the NPEs.

To configure discovery of Huawei VPNs, complete the following steps.

Procedure

1. Ensure that the output of the UPE and NPE devices is set to `display vsi verbose`.
2. Ensure that `m_RTBasedVPNs` in the `disco.config` database table is set to 1.
3. Ensure that the HuaweiMPLSTelnet discovery agent is enabled.
4. Back up and edit the following stitchers: `BuildMPLSContainers.stch`, `ResolveVPLSConnections.stch`, and `MPLSProcessing.stch`.
5. Set the value of `modelHuaweiVPN` to 1 in each stitcher.

6. Run a discovery.

What to do next

After discovery is finished, view the devices in a network view by filtering on VPLS VPN.

Configuring NAT discoveries

Configure a Network Address Translation (NAT) discovery to discover NAT environments, by mapping the address-space identifier for a NAT domain to the IP address of the associated NAT gateway device.

About Network Address Translation

The number of available IP addresses in the current 32–bit format is not enough to meet the growth in demand for access to the Internet. Network Address Translation (NAT) was designed as a short-term solution to this problem by providing a method of connecting multiple computers to an IP network using either a single unique public IP address, or a small number of unique public IP addresses.

NAT is commonly used in corporations, where a NAT router sits at the edge of the private network (referred to in this context as a *stub* domain) and translates the IP addresses attached to packets entering and leaving the stub domain. The NAT router, which effectively acts as an agent between the Internet and the local network, maintains a list of the mappings between public and private addresses.

Note: A stub domain is a local network using internal IP addresses. The network can use unregistered, private, IP addresses for internal communication—these addresses must be translated into unique, public, IP addresses when communicating outside the network. The addresses used internally by a given stub domain can also be used internally by any other stub domain.

For example, when a computer within the private network requests information from the public network, the NAT router automatically translates the private address of that computer into the public address of the domain, which is the only address that is transmitted to the public network. When the requested information is returned, the NAT router consults its internal list of public to private address mappings in order to forward the information to the appropriate computer.

There are a number of different ways to configure a NAT environment. The following descriptions detail the most common types of NAT environment.

Static NAT Environments

In a static NAT environment, the NAT router maps private and public addresses on a one-to-one basis, that is, the private address of a given device always maps to the same public address. This type of NAT environment is commonly used for devices that need to be accessible to the public network.

Dynamic NAT environments

In a dynamic NAT environment, the NAT router dynamically allocates public IP addresses, from a group of addresses, to devices on the private network that wish to communicate with the public network. A variation on dynamic NAT, *overloading* or PAT (Port Address Translation), maps multiple private addresses to the same public address using different ports.

Private Address Ranges

The Internet Assigned Numbers Authority (IANA) has assigned several address ranges to be used by private networks.

Address ranges to be use by private networks are:

- Class A: `10.0.0.0` to `10.255.255.255`
- Class B: `172.16.0.0` to `172.31.255.255`
- Class C: `192.168.0.0` to `192.168.255.255`

An IP address within these ranges is therefore considered non-routable, as it is not unique. Any private network that needs to use IP addresses internally can use any address within these ranges without any

coordination with IANA or an Internet registry. Addresses within this private address space are only unique within a given private network.

All addresses outside these ranges are considered public.

About NAT discovery

You can use Network Manager to manage NAT environments, though there are some restrictions on the types of NAT environment that are currently supported.

Network Manager can interrogate known, supported NAT gateways to obtain a list of public to private IP address mappings of devices in NAT domains. Alternatively, these mappings can be supplied manually. Network Manager can then discover those devices behind the NAT gateways that have a public IP address.

Each NAT domain has a unique address-space identifier. Each device in a NAT domain has the appropriate address-space identifier appended to its record. This enables the devices to be managed (for example, polled).

Restrictions on NAT discovery

There are several restrictions on the management of NAT environments using Network Manager.

Management of NAT environments using Network Manager is restricted by the following conditions:

- Network Manager can discover one or more NAT environments, but they must all use static NAT address mapping.
- Network Manager can discover devices in multiple NAT domains, regardless of whether the private IP addresses of the devices are duplicated in other NAT domains. However, the public IP address of each device in each domain must be unique.
- Devices within a NAT domain that have only private IP addresses cannot be discovered or managed by Network Manager.
- The discovery process must discover the NAT environment from outside, that is, from the public network.
- Virtual IP addresses such as Hot Standby Routing Protocol (HSRP) addresses cannot be mapped. The real physical address must be used.
- The following must be supplied before the discovery is run:
  - The addresses of all supported NAT gateways.
  - The NAT gateway translations must be discovered, either automatically or by supplying the NATTextFileAgent discovery agent with a flat file of public-to-private IP address mappings.

Differences in a NAT discovery process flow

The process flow of a NAT discovery differs from the process flow of a normal discovery.

Downloading translation information

NAT translation information is downloaded by the NAT agents into the translations.NATTemp database table before the finders process any other entities.

All other discovered devices are inserted into the finders.pending table while the BuildNATTranslation.stch stitcher creates a global translation table and stores it in the translations.NAT database table.

The finders, helpers, and other components that need to access devices can use this table to look up the address of any device behind a NAT gateway.

Creating the topology

When the topology is created, the `AddBaseNATTags.stch` stitcher adds NAT information to the topology record of each device in a NAT domain.

Table 62 on page 338 shows the information that is added to the topology record for each device.

Table 62. NAT information added to a device record	
Column	**Description**
`ExtraInfo->m_AddressSpace`	The name of the NAT address space to which the device belongs. This value is set in the `translations.NATAddressSpaceIds` table. If the discovery is not using NAT, or if the device is in the public domain, this value is NULL.
`ExtraInfo->m_NATTranslated`	A Boolean integer indicating whether the device lies behind a NAT gateway.
`ExtraInfo->m_InsideLocalAddress`	The private address of the device.
`ExtraInfo->m_OutsideGlobalAddress`	The public address of the device.

Configuring a NAT discovery

Configure a NAT discovery to discover NAT environments and to enable Network Manager to manage NAT environments.

About this task

You set most of the NAT discovery settings from the Discovery Configuration GUI, except for the following tasks:

- Configure the NATTextFileAgent agent to provides support for any unsupported NAT gateway devices
- Configure the NATGateway agent to correct the potential problem of incorrect connectivity when the NAT gateway is not in the public address space.

Quick reference for NAT discovery configuration

Use this information as a step-by-step guide to configuring a NAT discovery..

The steps are described in the following table.

Table 63. Quick reference for NAT discovery configuration		
Action	**Using the GUI**	**Using the command line**
1. Configure the discovery to use network address translation. You can do this using the Discovery Configuration GUI, or using the command line.	“Configuring NAT translation” on page 194	“Enabling NAT translation” on page 340
2. Define each NAT gateway device and its corresponding address space. You can do this using the Discovery Configuration GUI, or using the command line.		“Defining address spaces for NAT gateways” on page 340

Table 63. Quick reference for NAT discovery configuration (continued)

Action	Using the GUI	Using the command line
3. Seed the Ping finder with the IP address of each NAT gateway device.	"Seeding discovery" on page 183	Guidance for seeding a discovery "DiscoPingFinderSeeds.cfg configuration file" on page 217 Guidance for seeding a NAT discovery "Seeding discovery with NAT gateway addresses" on page 341
4. Define a scope zone for each NAT gateway device. **Note:** You do not need to define a scope zone for any NAT Gateway devices whose IP address is already within any other scope zones defined for the discovery. **Note:** Do not define an address space for the NAT gateway devices or for public subnet scopes. Address space can only be defined for private subnets.	"Scoping discovery" on page 182	Guidance for scoping a discovery "DiscoScope.cfg configuration file" on page 220 Example: how to define a scope zone for a private NAT subnet "Defining a scope zone within a NAT domain" on page 341
5. Define scope zones for the public subnets associated with each NAT address space. **Note:** Do not define an address space for the NAT gateway devices or for public subnet scopes. Address space can only be defined for private subnets.		
6. Where possible, define scope zones for the private subnet associated with each NAT address space. **Restriction:** You can only define a scope zone for a private NAT address space where the subnet and netmask combination of the private subnet is unique within the discovery configuration. Make the following settings when defining this scope: 1. Uncheck the **Add to Ping Seed List** option. You must do this because private subnets are not pingable. 2. Define an address space for this private subnet. The advantages of adding a scope zone for each private NAT address space are as follows: • This ensures that only addresses in that private space are fed back during the discovery. • If the NAT Gateway device and the devices within the associated NAT address space are routers. then adding a scope zone for that private NAT address space limits the download of unnecessary routing data.		

Table 63. Quick reference for NAT discovery configuration (continued)		
Action	**Using the GUI**	**Using the command line**
7. Enable NAT agents as follows: • For supported NAT Gateway devices, enable the CiscoNATTelnet or NATNetScreen agent. • For unsupported NAT Gateway devices, create a NAT mapping file and enable the NATTextFileAgent agent	"Activating agents" on page 189	"Enabling agents for supported NAT gateway devices" on page 342 "Enabling agents for unsupported NAT gateway devices" on page 342

Enabling NAT translation

You can set the discovery system to use NAT translation by editing $NCHOME/etc/precision/DiscoConfig.cfg to create or amend an insert into disco.NATStatus to set m_UsingNAT to 1 and m_NATStatus to 0.

About this task

The completed insert must resemble the following:

```
insert into disco.NATStatus
(
        m_UsingNAT,
        m_NATStatus
)
values
(
        1,
        0
);
```

Defining address spaces for NAT gateways

To specify the IP address of your NAT gateways and the address space identifier you want to use for each associated NAT domain, edit `DiscoConfig.cfg` to create or amend an insert into `translations.NATAddressSpaceIds`.

About this task

Follow these guidelines when defining address spaces for NAT gateways:

Procedure

- The IP address must be the public IP address that is accessible from the management server.
- The address space field can be any descriptive string, but avoid special characters such as quotes. Use the standard rules for DNS names for the address space because the address space might make up part of the name of these devices.

Results

The following example insert configures the discovery system for two NAT gateways.

```
insert into translations.NATAddressSpaceIds
(
        m_NATGatewayIP,
        m_AddressSpaceId
)
values
(
         '172.16.1.112',
        'NATDomain1'
);

insert into translations.NATAddressSpaceIds
(
```

```
        m_NATGatewayIP,
        m_AddressSpaceId
)
values
(
         '172.16.1.104',
        'NATDomain2'
);
```

Defining a scope zone within a NAT domain

You can customize inclusion and exclusion zones for individual NAT domains, using the `m_AddressSpace` column of the `scope.zones` table.

About this task

The following example insert defines an inclusion zone for a private subnet associated with a NAT domain.

```
insert into scope.zones
(
    m_Protocol, m_Action, m_Zones, m_AddressSpace
)
values
(
        1,
        1,
        [
            {
                m_Subnet="172.16.2.*",
            }
        ],
        "NATDomain1"
);
```

The above example defines one inclusion zone. Network Manager discovers any device with an IP address starting with `"172.16.2"`, that is, in the private `172.16.2.0` subnet with a mask of `255.255.255.0`, and that also belongs to the NAT address space `NATDomain1`. The protocol is set to `1`, that is, IP.

Note: Do not define an address space for the NAT gateway devices or for public subnet scopes. Address space can only be defined for private subnets.

Seeding discovery with NAT gateway addresses

Seed a NAT discovery by inserting into the Ping finder the IP addresses of the main routers within the system. Also seed the discovery with the IP addresses of the NAT gateway IPs.

About this task

In a NAT-based discovery, the discovery must discover the NAT gateways before discovering the rest of the network, so the NAT gateways must first be found with a finder.

Network Manager is configured to trigger the seeding of all the NAT gateways if NAT translation has been enabled. However, the triggering relies on the Ping finder being active. If seeding is done, for example, using only the File finder, then the NAT gateways are not pinged even if NAT translation has been enabled. It is good practice, therefore, to seed the discovery with all the NAT gateways. You can do this using the File finder, Ping finder, or any other method.

You can also seed the discovery with NAT gateways using the Discovery Configuration GUI.

Enabling NAT agents

If you are running a NetScreen® Firewall or a Cisco® Router as a NAT gateway, you must use either the CiscoNATTelnet agent or the NATNetScreen agent.

About this task

Ensure that you enable the appropriate NAT translation agents. These agents must run to discover the NAT gateways. If they are not run, discovery cannot complete as it cannot properly discover the network without first discovering the NAT Gateways.

The NAT agents are currently CiscoNATTelnet, NATNetScreen and NATTextFileAgent. The CiscoNATTelnet agent works on Cisco IOS routers providing NAT translation and is not certified for PIX firewalls. The NATNetScreen agent is for NetScreen firewalls.

If you are using a NAT gateway other than a NetScreen Firewall or a Cisco Router, you must use the Perl agent NATTextFileAgent.pl, as described in "Enabling agents for unsupported NAT gateway devices" on page 342.

Enabling agents for supported NAT gateway devices
The CiscoNATTelnet and NATNetScreen agents connect directly to the NAT gateways to download the address mappings. You can configure these agents.

Before you begin

Before running these agents, you must do the following tasks:

- Enable NAT translation
- Configure trap handling

About this task

To configure and run the agents:

Procedure

1. Enable the agents. There is an insert into the disco.agents table in the DiscoAgents.cfg configuration file for every installed discovery agent. To activate an agent, you must alter the insert so that the m_Valid column for that agent is set to 1. To deactivate an agent, ensure that m_Valid=0.

   The following example insert activates the CiscoNATTelnet agent.

   ```
   insert into disco.agents
   (
           m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence,
           m_DebugLevel, m_LogFile
   )
   values
   (
           'CiscoNATTelnet', 1, 8, 0, 2, 4,
           "$NCHOME/log/precision/CiscoNatTelnet.log"
   );
   ```

2. Run a discovery.

Enabling agents for unsupported NAT gateway devices
The NATTextFileAgent is provided as a backup if your NAT translation device is unsupported. You can configure this agent.

Before you begin

Before running the NATTextFileAgent agent, you must do the following tasks:

- Enable NAT translation
- Configure trap handling

About this task

The NATTextFileAgent reads a flat file called NATTranslations.txt that contains the NAT translations present on a particular NAT gateway. This allows the discovery an avenue to support a network containing a currently unsupported NAT gateway. This agent does not download its information from the NAT gateways, but reads a list of private to public IP address mappings from a flat file.

To configure and run the agent:

Procedure

1. Install the Perl API. All Perl agents require the API to run. The API is installed by default in Network Manager.

   To check whether the API is installed, check that the following file exists:

   ```
   $NCHOME/precision/bin/ncp_perl
   ```

   If the file is listed, then the Perl API is installed.
2. Create a NAT mapping file to be read by the agent that contains the public to private address mappings. Your NAT mapping file must be in a format that can be read by the agent, that is, the values must be valid IP addresses specified in columns separated by tabs.

   By default, the agent uses the file $NCHOME/etc/precision/NATTranslations.txt. If you want to create your own mappings, you must back up and edit this default file. To make the agent use the non-default NAT mapping file, edit the following line in $NCHOME/precision/disco/agents/Perlagents/NATTextFileAgent.pl:

   ```
   my $natFileName  = "$ENV{$NCHOME}/etc/precision/NATTranslations.txt";
   ```
3. The NAT mapping file contains the following columns:
   - The IP address of the NAT gateway of the NAT domain to which the device belongs. You must specify mappings for all NAT gateways in the same file.
   - The outside global address of the device, that is, the public address of the device.
   - The inside local address of the device, that is, the private address of the device.

   The following example shows a NAT mapping file for two gateways having IP addresses of 1.2.3.4 and 1.2.3.9 respectively.

   ```
   // NATGatewayIP              PublicIP                PrivateIP
   1.2.3.4                      2.3.4.5                 10.10.1.1
   1.2.3.4                      2.3.4.6                 10.10.1.2
   1.2.3.9                      2.3.6.1                 10.10.1.1
   1.2.3.9                      2.3.6.2                 10.10.1.2
   ```

   Note: From the perspective of the management station, the public IP address of a particular gateway translation is not necessarily the same as the public address that the management stations sees. The public address is the IP address that the gateway retrieves from one port and then translates and places on another port. This difference is important to note when you have chained gateways, where an IP address can be translated multiple times. The public IP is effectively the IP address that is closer to the management domain.
4. Enable the agent. There is an insert in the disco.agents table in the DiscoAgents.cfg configuration file for every installed discovery agent. To activate an agent, alter the insert so that the m_Valid column for that agent is set to 1. To deactivate an agent, ensure that m_Valid=0.

   The following example insert activates the NATTextFileAgent agent.

   ```
   insert into disco.agents
   (
           m_AgentName, m_Valid, m_AgentClass, m_IsIndirect, m_Precedence, m_IsPerl
   )
   values
   (
           'NATTextFileAgent', 1, 8, 0, 2, 1
   );
   ```
5. Ensure that the NATTimer.stch stitcher has been configured to trigger a rediscovery against the NAT gateways. By default, the NATTimer.stch stitcher runs every hour. You can alter this interval by locating the following line in the stitcher file and changing the integer value:

   ```
           ActOnTimedTrigger( ( m_Interval ) values ( 1 ) ; ) ;
   ```
6. Run a discovery.

Enable agent for NAT gateway devices in private address space
When the NAT gateway is not in the public address space, you can enable the NATGateway agent to correct the potential problem of incorrect connectivity.

About this task

The discovery assumes that the management interface of the NAT gateway is in public address space. If this is not the case, Network Manager cannot identify the address space of interfaces on the NAT gateway device, which might result in incorrect connectivity. For example, when a VPN is used to access the management interface, the NAT gateway management interface is not in the public address space.

The NATGateway agent enables Network Manager to determine whether a given interface on a NAT gateway device is on the public or private side of the NAT gateway, and thereby correctly resolve device connectivity.

To overcome this problem, activate the NATGateway agent and provide Network Manager with a mapping file, NATGateways.txt. In this file, provide a list of all NAT gateway devices together with the interfaces on each device and a field to indicate whether the interface is on the public or private side of the NAT gateway.

This agent works in conjunction with the NATGatewayRetProcessing.stch stitcher and with the NATGateways.txt file in NCHOME/precision/etc

Table 64 on page 344 provides the format of the NATGateways.txt file by showing an example of the content of this file. Fields in this text file must be separated by tabs.

Table 64. Format of NATGateways.txt file

Base name	Inside or outside	Interface IP address
1.1.1.4	outside	172.16.4.10
1.1.1.4	inside	10.52.2.10
sca_T1ukP_16	outside	192.168.36.93
sca_T1ukP_16	outside	192.168.36.98

Example: Configuring a NAT discovery

This example illustrates how to define address spaces using the NATTextFileAgent agent and how to set up associated discovery scopes.

Before you begin

Do the following tasks before running through the steps in this example:

- Configure the discovery to use network address translation.
- Seed the Ping finder with the IP address of each NAT gateway device.

About this task

In this example the NAT gateway devices are unsupported. This means that the NATTextFileAgent agent must be used in this NAT discovery.

The NATTextFileAgent agent uses a NAT mapping file, with the following content. There are three NAT gateway devices, with mappings for each of the devices in the associated address spaces.

```
//First NAT gateway and mappings
//NATGateway              PublicIP                    Private IP
201.201.201.201         61.61.61.1              192.168.1.1
201.201.201.201         61.61.61.2              192.168.1.2
201.201.201.201         61.61.61.3              192.168.1.3
201.201.201.201         61.61.61.4              192.168.1.4
```

```
201.201.201.201         61.61.61.5              192.168.1.5
201.201.201.201         61.61.61.6              192.168.1.6

//Second NAT gateway and mappings
//NATGateway               PublicIP                       Private IP
202.202.202.202         62.62.62.1              192.168.1.1
202.202.202.202         62.62.62.2              192.168.1.2
202.202.202.202         62.62.62.3              192.168.1.3
202.202.202.202         62.62.62.4              192.168.1.4
202.202.202.202         62.62.62.5              192.168.1.5
202.202.202.202         62.62.62.6              192.168.1.6

//Third NAT gateway and mappings
//NATGateway               PublicIP                       Private IP
203.203.203.203         63.63.63.1              192.168.3.1
203.203.203.203         63.63.63.2              192.168.3.2
203.203.203.203         63.63.63.3              192.168.3.3
203.203.203.203         63.63.63.4              192.168.3.4
203.203.203.203         63.63.63.5              192.168.3.5
203.203.203.203         63.63.63.6              192.168.3.6
```

For the first and second address spaces private IP address space is not unique. For both of these address spaces, the private IP address space is defined by a subnet and netmask combination of 192.168.1.0/29.

Based on this NAT gateway device and address space data, define discovery scopes as follows.

Procedure

1. Define each NAT gateway device and its corresponding address space.
   In this example, the names of the three NAT address spaces are RTP1, RTP2, and RTP3. For example, for the third NAT gateway device, the following insert defines the NAT device and its associated address space, RTP3:

   ```
   insert into translations.NATAddressSpaceIds
   (
       m_NATGatewayIP, m_AddressSpaceId
   )
   values
   (
   "203.203.203.203", "RTP3"
   );
   ```

2. Define a scope zone for each NAT gateway device.

   Note: You do not need to define a scope zone for any NAT Gateway devices whose IP address is already within any other scope zones defined for the discovery.

   For example, for the first NAT gateway device, the following insert defines the scope zone:

   ```
   insert into scope.zones
   (
       m_Protocol, m_Action, m_Zones, m_AddressSpace
   )
   values
   (
           1,
           1,
           [
               {
                   m_Subnet="201.201.201.201",
                   m_NetMask=32
               }
           ],
           ""
   );
   ```

3. Define scope zones for the public subnets associated with each NAT address space.
   For example, for the third public subnet, the following insert defines the scope zone:

   ```
   insert into scope.zones
   (
       m_Protocol, m_Action, m_Zones, m_AddressSpace
   )
   values
   (
   ```

```
        1,
        1,
        [
            {
                m_Subnet="63.63.63.0",
                m_NetMask=29
            }
        ],
        ""
);
```

4. Define a scope zone for the private subnet associated with the third NAT address space only.

   Restriction: You can only define a scope zone for a private NAT address space where the subnet and netmask combination of the private subnet is unique within the discovery configuration. This excludes the first and second private subnet.

   For the third private subnet, the following insert defines the scope zone:

```
insert into scope.zones
(
    m_Protocol, m_Action, m_Zones, m_AddressSpace
)
values
(
        1,
        1,
        [
            {
                m_Subnet="192.168.3.0",
                m_NetMask=29
            }
        ],
        "RTP3"
);
```

5. Enable the NATTextFileAgent agent.

What to do next

Now you can launch the NAT discovery.

Post-configuration NAT tasks

After you have configured NAT discoveries, you can complete several post-configuration tasks.

Tracking the progress of a NAT discovery

During the discovery of the NAT-translation devices, you can track the discovery status in the disco.NATStatus values.

About this task

During the discovery, you initially see only the NAT-translating devices displayed in the agent despatch and returns tables. All the other data returned from the finders is stored in finders.pending database table while the discovery of the NAT-translating devices takes place.

Issue the following OQL select statement to see the discovery status:

```
select * from disco.NATStatus;
```

This statement displays a value from 0 to 4. The meaning of the value is:

Procedure

- 0: NAT discovery in initial state. NAT devices have not been processed.
- 1: NAT discovery initiated. NAT gateway IPs have been sent to the Ping finder to verify their existence
- 2: NAT discovery is running.

- 3: NAT discovery processing. All the NAT gateways have been processed and the discovery is now building the `translations.NAT` table. This table ensures the correct discovery of the rest of the network.
- 4: NAT discovery complete. The entries in the finders.pending table have been moved into the finders.processing table, and the discovery continues as normal.

Results

Use the results of this query to debug a problematic NAT discovery. The value indicates whether any discovery problems are caused by NAT or caused by the standard (non-NAT) part of the discovery process.

Debugging a NAT discovery

To analyze a NAT discovery, use ncp_oql to follow the data through the system from the start (finders) to the end until you determine where the data is incorrect. Incorrect data indicates whether the problem is with an agent, a device, or a stitcher.

About this task

There are several queries that are useful when debugging a discovery, both NAT-based and not NAT-based.

The following OQL query indicates which agents are currently either being started (`m_State=1`), starting (`m_State=2`) or running (`m_State=3`):

```
select * from agents.status where m_State <> 0 AND m_State <> 4;
```

This query tells you which agents the current phase is waiting for in order to complete. The discovery is waiting for the agents that are meant to complete in the current phase and are in state 1, 2 or 3.

```
select * from <agentName>.despatch
where m_UniqueAddress NOT IN
      ((
      select m_UniqueAddress from <agentName>.returns where m_LastRecord = 1
      ));
```

Using the first query, you can see which agents are still running in a particular phase.

Using the following query, you can determine which entity that particular agent is processing. This can be useful in determining a problem device within your network:

```
select * from translations.ipToBaseName where m_IpAddress = '<ip>';
```

This second query shows you what base address and base name is being used for a particular IP as well as whether this IP address is considered to be in scope.

Activating the Containment Model for use with NAT

The `NATAddressSpaceContainers.stch` stitcher creates virtual objects for each address space that contains the entities within that address space. You can activate this stitcher by uncommenting the line, // ExecuteStitcher("NATAddressSpaceContainers");, in the file `$NCHOME/precision/disco/stitchers/BuildContainment.stch`.

Viewing NAT environments using Topoviz Network Views

Using Topoviz Network Views, you can create network views based on the values of any column in the topology record of an entity. A NAT Address Spaces Dynamic Distinct view is created automatically if you activated NAT discovery as part of your discovery configuration.

About this task

As an example of viewing NAT environments, you could created a filtered network view or a Dynamic Distinct view on the following field in the NCIM topology database:

- ipEndPoint table
- addressSpace field

Note: A NAT Address Spaces Dynamic Distinct view is created automatically if **Enable Network Address Translation (NAT) Support** is turned on as part of your discovery configuration.

Configuring cross-domain discoveries

To enable links between devices in different domains to be shown in the network and topology views, you must configure and run cross-domain discoveries on the different domains.

About this task

Configuring cross-domain discovery is an advanced procedure that requires knowledge of the discovery data flow, the OQL query language, the database structure, and details of your network connectivity and composition.

Related reference

RCA considerations in a cross-domain network

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.

About cross-domain discovery

Cross-domain discovery can be configured to join two or more discovered domains together.

For performance or operational reasons, networks are often discovered in sections, known as *discovery domains*. For example, if your network is so large that discovering it in one discovery takes too long, you might choose to split network discovery into different domains.

Discovering the network in domains can be more convenient and faster. You can also choose to have different configuration options for different domains. For example, each domain has its own poll policies. However, there are disadvantages to discovering the network in pieces. If a device in domain A is connected to a device in domain B, this connection is not represented in the topology database or in the GUI. Domains must be viewed separately.

If you want to visualize multiple domains linked together in one network view, you must enable, configure, and run cross-domain discoveries. Connections between devices in different domains are found and added to the topology.

When all discovered domains have been aggregated, **Network Views** can be composed of devices from all domains. In the **Network Hop View**, searches for devices can span domains.

Note: Cross-domain network views can not be polled; only network views from individual domains can be polled.

Considerations for splitting the network into domains

Links between devices in different domains are not as easy to discover as links between devices within one domain.

It is important to scope your discovery domains to ensure the minimum of links between domains. For example, you would not normally split the network such that highly connected switches were in different domains. Natural splits for domains are often along geographical lines.

Restriction:

However you split your network, you must ensure that any given device appears in only one domain. That is, the discovery domains must not overlap if you want to join them together using cross-domain discovery.

Related reference

RCA considerations in a cross-domain network

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.

Enabling cross-domain linking

The first step of configuring cross-domain linking is to enable the links between domains in the `DiscoConfig.cfg` configuration file. By default, cross-domain linking is disabled.

Procedure

1. Back up and edit the `$NCHOME/etc/precision/DiscoConfig.domain.cfg` file.
2. Make the following settings:
   - Set **m_EnableCrossDomainProcessing** to 1.
   - Set **m_InferPEsUsingBGP** to 0 to disable the inference of Provider Edge (PE) devices. PE device inference is incompatible with cross-domain discoveries. This setting can also be made in the **Advanced** tab of the **Network Discovery Configuration** GUI by clearing **Enable Inference of PEs using BGP data on CEs**.
3. Repeat these steps in the `DiscoConfig.domain.cfg` file of each domain that you want to be linked.
4. In the `LinkDomainsPopulateDomainAdjacencies.stch` stitcher file, define the adjacencies between domains in the tmpDomainAdj.adjacencies table.

   Use INSERT statements to define the adjacencies. Sample INSERT statements are provided in the file. Each INSERT statement defines only one adjacency. You can put the INSERT statements in any order.

   For example, to define adjacencies between two domains that are called NORTH and SOUTH use the following INSERT statement:

   ```
   insert into tmpDomainAdj.adjacencies values ('NORTH', 'SOUTH');
   ```

   The following example shows the INSERT statements to use when you have three domains: EUROPE, ASIA, and AMERICA. EUROPE is adjacent to both ASIA and AMERICA.

   ```
   insert into tmpDomainAdj.adjacencies values (EUROPE, ASIA);
   insert into tmpDomainAdj.adjacencies values (EUROPE, AMERICA);
   ```

   To define an extra adjacency between ASIA and AMERICA, use another INSERT statement:

   ```
   insert into tmpDomainAdj.adjacencies values (ASIA, AMERICA);
   ```

Configuring cross-domain links

To configure cross-domain links, decide which method of linking is appropriate for your network and configure the relevant stitchers.

About this task

Configuring cross-domain links between Layer 1 devices

You can enable cross-domain links between Layer 1 devices. You can enable or disable the creation of links between devices in different domains that use Layer 1 protocols. You can also configure how the links are created.

Procedure

1. Back up and edit the stitcher file `NCHOME/precision/disco/stitchers/LinkDomains.stch`.

2. Locate the line that begins `int linkViaLayer1NameInterface`.
3. Set the value to `1` to enable connections between Layer 1 devices in different domains to be created as links in the topology.
4. Configure the following advanced parameters, which apply to links created using all enabled technologies and methods:

   previewChanges
   Set to `1` if you want the links not to be created but only saved to a log file. To view previewed links, set the **ncp_disco** process to `-messagelevel debug` before starting the discovery. Messages about the connections that would have been created are logged to the `NCHOME/log/precision/ncp_disco.`*`DOMAIN`*`.log` file. Set to `0` if you want the links to be created.

   lowLayerResolutionMode
   If more than one type of connection exists between two ports, you can choose at what level that connection is created. It is often more useful to use the more specific, lowest level connection.

   - Set to `0` to create only the connection that was found by the cross-domain stitchers.
   - Set to `1` to create the connection **only** between the ports that are stacked at the lowest level under an interface. For example, where a POS interface is stacked over a SONET port, this option creates the connection only between the SONET ports. If you enable this option, the stitching takes longer.
   - Set to `2` to create the connection between the interfaces **and** their lowest stacked ports. For example, where a POS interface is stacked over a SONET port, this option creates one connection between the SONET ports and one connection between the POS interfaces. If you enable this option, the stitching takes longer.

Configuring cross domain links between other device technologies

You can create cross-domain links between devices using device technologies such as /30 and pseudowire. You can also configure how the links are created.

Procedure

1. Back up and edit the stitcher file `NCHOME/precision/disco/stitchers/LinkDomains.stch`.
2. Optional: To enable /30 connections between devices in different domains to be created as links in the topology, locate the line that begins `int linkViaSlash30Subnet` and set the value to `1`.
   a) Configure the following parameters:

   preventLinkPropagation
   Set to `0` to add the /30 connection even if there is an existing Layer 2 connection between the two network entities. Set to `1` to prevent a /30 connection being added if there is already an existing Layer 2 connection from the entity.

   linkSlash30InLayer2
   Set to `0` to prevent /30 links being added as Layer 2 links. Set to `1` to add /30 links as Layer 2 links. Setting both this property and `linkSlash30InLayer3` to `0` prevents /30 connections being created, even though they are discovered, which can make the discovery take longer. You can set both properties to `1` to add /30 links as both Layer 2 and layer 3 links.

   linkSlash30InLayer3
   Set to `0` to prevent /30 links being added as Layer 3 links. Set to `1` to add /30 links as Layer 3 links. Setting both this property and `linkSlash30InLayer2` to `0` prevents /30 connections being created, even though they are discovered, which can make the discovery take longer. You can set both properties to `1` to add /30 links as both Layer 2 and layer 3 links.
3. Optional: To enable pseudowire connections between devices in different domains to be created as links in the topology, locate the line that begins `int linkViaPseudoWires` and set the value to `1`.
   a) Configure the following parameters:

resolvePWViaFarEndIP
Set to `0` to disable the creation of links using far-end Pseudowire IP addresses. Set to `1` to create links using far-end Pseudowire IP addresses.

resolvePWViaLabels
Set to `0` to disable the creation of links using Pseudowire inverse label searching. Set to `1` to create links using Pseudowire inverse label searching.

resolvePWViaVPLSInterface
Set to `0` to disable the creation of links using Virtual Private Label Switching (VPLS) duplicates. Set to `1` to create links using Virtual Private Label Switching (VPLS) duplicates.

4. Optional: To enable BGP session connections between devices in different domains using the BGP session information downloaded by the bgp agents, locate the line that begins `int linkViaBGPSessions` and set the value to `1`.
   a) Configure the following parameters:

linkBGPInLayer2
Set to `1` to place the link into the L2 topology if the BGP cross domain processing finds a BGP link between domains.

Set to `0` to disable.

linkBGPInLayer3
Set to `1` to place the link into the L3 topology if the BGP cross domain processing finds a BGP link between domains.

Set to `0` to disable.

linkEstalbishedSessionsOnly
Set to `1` to connect the two BGP interfaces if a cross domain BGP session is found for which the status is not established.

Set to `0` to connect the two BGP interfaces if a cross domain BGP session is found only if the status has been established.

linkBGPSessionsStrictly
Set to `1` to enable linking via general IP matches, should a strict BGP session match fail.

Set to `0` to disable.

5. Optional: To enable CDP connections between devices in different domains to be created using the CDP agent return data, locate the line that begins `int linkViaCDP` and set the value to `1`.
   a) Configure the following parameters:

linkViaCDPAtLowestInterface
Set to `0` to connect together found interfaces.

Set to `1` to attempt to recurse in order to connect at the lowest level port/interface.

linkViaCDPAtLayer2
Set to `1` to place the link into the L2 topology if a CDP link between domains is found.

Set to `0` to disable.

linkViaCDPAtLayer3
Set to `1` to place the link into the L3 topology if a CDP link between domains is found.

Set to `0` to disable.

6. Optional: To use MPLS TE agents neighbor data to resolve connections between devices in separate domains, locate the line that begins `int linkViaMPLSTE` and set the value to `1`.
   a) Configure the following parameters:

linkViaMPLSTEAtLayer2
Set to `1` to place the link into the L2 topology if an MPLS TE link between domains is found.

Set to `0` to disable.

linkViaMPLSTEAtLayer3
Set to `1` to place the link into the L3 topology if an MPLS TE link between domains is found.

Set to `0` to disable.

linkViaMPLSTEAtMPLSTE
Set to `1` to enables the creation of an MPLS TE connection if an MPLS TE link can be identified.

Set to `0` to disable.

7. Optional: To use OSPF agents neighbor data to resolve connections between devices in separate domains, locate the line that begins `int linkViaOSPF` and set the value to `1`.
   a) Configure the following parameters:

linkViaOSPFAtLayer2
Set to `1` to place the link into the L2 topology if an OSPF link between domains is found.

Set to `0` to disable.

linkViaOSPFAtLayer3
Set to `1` to place the link into the L3 topology if an OSPF link between domains is found.

Set to `0` to disable.

linkViaOSPFAtOSPF
Set to `1` to enables the creation of an OSPF connection if an OSPF link can be identified.

Set to `0` to disable.

8. Optional: To use PIM agents neighbor data to resolve connections between devices in separate domains, locate the line that begins `int linkViaPIM` and set the value to `1`.
   a) Configure the following parameters:

linkViaPIMAtLayer2
Set to `1` to place the link into the L2 topology if a PIM link between domains is found.

Set to `0` to disable.

linkViaPIMAtLayer3
Set to `1` to place the link into the L3 topology if a PIM link between domains is found.

Set to `0` to disable.

linkViaPIMAtPIM
Set to `1` to enables the creation of a PIM connection if a PIM link can be identified.

Set to `0` to disable.

9. Configure the following advanced parameters, which apply to links created using all enabled technologies and methods:

previewChanges
Set to `1` if you want the links not to be created but only saved to a log file. To view previewed links, set the **ncp_disco** process to `-messagelevel debug` before starting the discovery. Messages about the connections that would have been created are logged to the `NCHOME/log/precision/ncp_disco.`*`DOMAIN`*`.log` file. Set to `0` if you want the links to be created.

lowLayerResolutionMode
If more than one type of connection exists between two ports, you can choose at what level that connection is created. It is often more useful to use the more specific, lowest level connection.

- Set to `0` to create only the connection that was found by the cross-domain stitchers.
- Set to `1` to create the connection **only** between the ports that are stacked at the lowest level under an interface. For example, where a POS interface is stacked over a SONET port, this option creates the connection only between the SONET ports. If you enable this option, the stitching takes longer.
- Set to 2 to create the connection between the interfaces **and** their lowest stacked ports. For example, where a POS interface is stacked over a SONET port, this option creates one connection between the SONET ports and one connection between the POS interfaces. If you enable this option, the stitching takes longer.

Adding and editing cross-domain links manually

If you do not see links between devices in different domains, you can manually create or configure them.

About this task

To add or edit cross-domain links, complete the following steps:

Procedure

1. Back up and edit the stitcher file `NCHOME/precision/disco/stitchers/LinkDomainsLoadPresetConnections.stch`.
2. Uncomment the OQL insert statement.
3. Copy one OQL insert statement for each connection that you want to make.
4. Edit the OQL insert statement and add the details of the connection that you want to create, using the following parameters:

   entryNo
   A unique numeric ID for this row. Start at 1 and increase to n.

   action
   Set to ADD to add a connection.

   aEndDiscoDomainName
   The domain in which the device at the beginning of the connection was discovered. This connection is created only after a discovery is run in this domain.

   aEndDiscoEntityName
   The entityName of the device at the beginning of the connection.

   zEndNCIMDomainName
   The domain in which the device at the end of the connection is located. If a discovery is run in only this domain, this connection is not created.

   zEndNCIMEntityName
   The entityName of the device at the end of the connection.

   topologyEntityType
   The NCIM topology entityType of the connection.

Configuring cross-domain links using interface descriptions

You can create connections between all interfaces that match a search of the interface descriptions.

About this task

To search for interfaces and create connections between them, complete the following steps:

Procedure

1. Back up and edit the stitcher file `NCHOME/precision/disco/stitchers/LinkDomainsLoadInterfaceDescriptionMatches.stch`.
2. Copy one OQL insert statement for each connection that you want to make.
3. Edit the OQL insert statement and add the details of the connection that you want to create, using the following parameters:

   entryNo
   A unique numeric ID for this row. Start at 1 and increase to n.

   action
   Set to ADD to add a connection.

   onlyAdminUp
   Set to 1 to restrict the search to interfaces whose administrative status is Up. Set to 0 to include all interfaces, whether their administrative status is Up or Down.

Administrative status is the desired state of the interface. A network administrator can set the administrative status of an interface to Up, Down, or Testing.

aEndDiscoMatchType
Set to EXACT to perform an exact text search or REGEX to perform a regular expression search for source interfaces in the databases of **ncp_disco**.

aEndDiscoDomainName
The domain in which the device at the beginning of the connection was discovered. This connection is created only after a discovery is run in this domain.

aEndDiscoSearchTerm
The search term against which an interface in the aEndDiscoDomainName domain must match in the databases of **ncp_disco**.

zEndNCIMMatchType
Set to EXACT to perform an exact text search or REGEX to perform a regular expression search for target interfaces in the NCIM database.

zEndNCIMDomainName
The NCIM domain in which to search for target interfaces in the NCIM database.

topologyEntityType
The NCIM topology entityType of the connection in the NCIM database.

Results

All interfaces matching the search are connected together.

Example

The following example shows an insert that connects all interfaces in the NCOMS domain whose descriptions contain the string `connection to vmhost_network` to all interfaces in the NCOMSADJ domain whose descriptions also contain the string `connection to vmhost_network`:

```
INSERT INTO linkDomains.interfaceDescriptionMatch
(
    entryNo,
    action,
    onlyAdminUp,
    aEndDiscoMatchType,
    aEndDiscoDomainName,
    aEndDiscoSearchTerm,
    zEndNCIMMatchType,
    zEndNCIMDomainName,
    zEndNCIMSearchTerm,
    topologyEntityType
)
VALUES
(
    1,                              // entryNo
    'ADD',                          // action
    1,                              // onlyAdminUp - must be up
    'EXACT',                        // aEndDiscoMatchType
    'NCOMS',                        // aEndDiscoDomainName
    'connection to vmhost_network', // aEndDiscoSearchTerm
    'EXACT',                        // zEndNCIMMatchType
    'NCOMSADJ',                     // zEndNCIMDomainName
    'connection to vmhost_network', // zEndNCIMSearchTerm
    72                              // topologyEntityType
);
```

The following example shows an insert that connects all interfaces in the NCOMS domain whose descriptions match the regular expression `ELON(GW|WR|AR)` to all interfaces in the NCOMSADJ domain whose descriptions contain the string `connection to PE2_ASBR_AS2`:

```
INSERT INTO linkDomains.interfaceDescriptionMatch
(
    entryNo,
    action,
    onlyAdminUp,
    aEndDiscoMatchType,
```

```
    aEndDiscoDomainName,
    aEndDiscoSearchTerm,
    zEndNCIMMatchType,
    zEndNCIMDomainName,
    zEndNCIMSearchTerm,
    topologyEntityType
)
VALUES
(
    2,                              // entryNo
    'ADD',                          // action
    1,                              // onlyAdminUp - must be up
    'REGEX',                        // aEndDiscoMatchType
    'NCOMS',                        // aEndDiscoDomainName
    'ELON(GW|WR|AR)',               // aEndDiscoSearchTerm
    'EXACT',                        // zEndNCIMMatchType
    'NCOMSADJ',                     // zEndNCIMDomainName
    'connection to PE2_ASBR_AS2',   // zEndNCIMSearchTerm
    72                              // topologyEntityType
);
```

Running cross-domain discoveries

Before creating your cross-domain network views, run cross-domain discoveries.

Before you begin

Before running cross-domain discoveries, configure how the cross-domain links are built.

About this task

To run your cross-domain discoveries, complete the following steps:

Procedure

1. Run a full discovery in your first domain.
2. Run a full discovery in your second domain.
3. Run a full discovery in any other domains.

   After discovering all domains once, some connections might be duplicated, and your links between domains might not be as expected.
4. Run a second full discovery in every domain.

   Any inferred connections are replaced by discovered connections.

Results

The links between each domain are added by the cross-domain stitchers. The AGGREGATION domain is created by the aggregation stitching, which runs at the end of discovery (and whenever the topology is updated).

What to do next

Create any network views you want, using the AGGREGATION domain.

Creating cross-domain network views

After running your cross-domain discoveries, create cross-domain network views to visualize your network. You can create standard or dynamic network views.

Before you begin

Before creating a cross-domain network view, you must configure and run cross-domain discoveries for each domain that you want to aggregate.

About this task

To create network views across domains, complete the following steps:

Procedure

1. Click the **Incident** icon and select **Network Availability** > **Network Views**.
2. Within **Network Views**, in the **Libraries** tab, click **New View**  .
3. Complete the **General** tab as follows:

   Name
   Type a name for the network view, dynamic view, or network view container.

   Important: It is best practice to use network view names containing Latin characters only. Network views names containing non-Latin characters (for example Cyrillic characters) are not supported as they cannot be imported and exported when migrating to a new version of Network Manager.

   Parent
   Select the node under which the view appears in the hierarchy in the **Navigation Tree**. To display the view on the top level, select NONE.

   Type
   Select a type of network view. Because the resulting network view will contain all devices from all discovered networks, consider the size of the network views so as not to create unnecessary load on the server.

   Fill in the other fields as appropriate for that type of network view.
4. Click the **Filter** tab. Complete the tab as follows:

   Domain
   Select the **AGGREGATION** domain.

   Fill in the other fields as appropriate for that type of network view.
5. Click **OK**.

   The new view is added to the navigation tree in the **Navigation Panel**. If you added the view to a container, expand the container node to see the new view in the tree.

Results

Your network view shows devices from all discovered domains.

Example: Small network or Proof of Concept (POC)

If you want to check whether two or more domains have been discovered and joined together as you expect, you could recreate all the network views that are usually created automatically after a discovery finishes. Do this only if you are sure that the total of the resulting network views will not have an adverse performance impact. For example, you might want to do this while testing cross-domain discovery, on a non-production system. To create all the usual network views, complete the following steps:

1. Create a new network view of type **Dynamic Views - Template**.
2. Select the domain **AGGREGATION**.
3. Select the template **IP Default**.

Checking cross-domain links

After creating your cross-domain network views, check that you see the links between domains that you expect.

Before you begin

Before checking the links between domains, ensure that you have run a full cross-domain-enabled discovery twice in each domain.

About this task

If you do not see the links between the domains that you expect, complete the following steps:

Procedure

1. Ensure that the appropriate agents are enabled for the technologies and devices that are on the edges of your domains.
2. Ensure that all the appropriate technologies are enabled for the cross-domain stitching.
3. Check that the boundary between domains is appropriate.

   It is important to scope your discovery domains to ensure the minimum of links between domains. For example, you would not normally split the network such that highly connected switches were in different domains. Natural splits for domains are often along geographical lines.

   Restriction:

   However you split your network, you must ensure that any given device appears in only one domain. That is, the discovery domains must not overlap if you want to join them together using cross-domain discovery.
4. If you know that links between devices in different domains are present, but they are not shown in the Network Views, you can add or edit the links manually.

Fix Pack 2 Configuring geographical discoveries

Before you can view devices in their geographical context, you must configure a geographical discovery. Choose a method of enriching devices with geographical information.

Fix Pack 4 Configuring a base mapping layer

To visualize network topology overlaid onto a contextual map, configure the base mapping layer that you want to use.

About this task

The base layer provides the geographical base upon which the devices are overlaid. Devices and event status are contained within the topology layer and status layer, which are automatically calculated and are always visible.

Fix Pack 4 You can define one or more base layers for each mapping provider. One base layer is defined as the default. Users can select a different base layer for their session in the geographical views.

To add or change a custom layer, see “Adding custom map layers” on page 363.

To add or change a base layer, complete the steps below:

Procedure

1. Ensure that you have the correct licenses for any mapping provider that you intend to use.

No licenses for any mapping providers are included with Network Manager. The default mapping base layer is OpenStreetMap, which does not require a license.

2. Back up and edit the following file: `$JazzSM_HOME/profile/installedApps/JazzSMNode01Cell/isc.ear/ncp_gis.war/resources/config.json`
3. Edit the `mapProvider` section.

   The following code extracts show an example section from the `config.json` file.

```
"mapProvider": {
        "baseMapLayerName": "IBM Network Management",
        "baseLayer": "OpenStreetMap",
        "baseLayersDefn": [
            {
                "baseLayerName": "OpenStreetMap",
                "baseLayerType": "osm"
            },
            {
                "baseLayerName": "Microsoft Bing",
                "baseLayerType": "bing",
                "imagerySet": "Road",
                "key": "INSERT_YOUR_MICROSOFT_KEY"
            },
            {
                "baseLayerName": "OpenLayer XYZ Format",
                "baseLayerType": "xyz",
                "urls": ["https://api.mapbox.com/styles/v1/mapbox/
streets-v9/tiles/256/{z}/{x}/{y}?access_token=INSERT_YOUR_KEY"]
            },
            {
                "baseLayerName": "My Custom GeoServer WMS Layer",
                "baseLayerType": "wms",
                "url": "https://localhost:8443/geoserver/wms",
                "params": {
                }
            },
            {
                "baseLayerName": "My Custom GeoServer WMTS Layer",
                "baseLayerType": "wmts",
                "capabilities": {
                }
            }
        ],
```

4. Configure a new or existing base layer. Edit the existing section for a supported map provider or copy and edit it. You can include multiple base layers for one mapping provider. For example, you can define multiple base layers that all use the OpenStreetMap provider by editing the `baseLayer` value.

   Important: Do not delete any empty sections from the `config.json` file. Missing sections can cause errors.

   a) Specify any unique descriptive name for the `baseLayerName`.

      In the preceding example, the `baseLayerName` value for the first base layer is set to `OpenStreetMap` on line 3.

   b) Specify the mapping provider for this base layer in the `baseLayerType` parameter.

      You can specify only one of the following defined keywords for the `baseLayerType` value:

      - `bing` for Bing Maps
      - `osm` for Open Streetmap
      - `wms` for Web Map Service
      - `wmts` for Web Mapping Tile Service
      - `xyz` for OpenLayer XYZ format

5. Configure any mapping provider-specific parameters necessary.

   Mapping providers might support additional parameters, however, only those parameters that are listed in the default configuration are supported by Network Manager. Refer to the mapping provider documentation for information about what values are acceptable for these parameters.

6. Ensure that the `baseLayer` parameter exactly matches one of the `baseMapLayerName` values.

The `baseLayer` parameter defines the default base layer. You can only specify one `baseLayer` parameter.

7. Save and close the file.

   The geographical views must be closed and reopened before a user can see any changes to base layers.

Fix Pack 3 Enriching discovery with geographical data

To display geographical maps, you must enrich the network topology with geographical data.

About this task

Enriching the network topology with geographical data requires several steps:

- Retrieve the geographical data from an external database or system.
- Add the geographic data to the topology.
- Create a collection in Network Manager that collects devices to a geographic location.
- Optionally, create collections to represent hierarchies of geographical regions.

Note: Customizing discovery is an advanced procedure. You must be familiar with the discovery data flow, general database administration, and the stitcher language.

To retrieve the geographical data, two options are to add it to the `SysLocation` field in the device record by using a script, or to export it from a database to a `.csv` file. Other options are possible depending on how your data is stored.

To create collections and add geographical data to the topology, you must use the stitcher language. Two example methods are supplied with the product: the `GeoBySysLocation` and `GeoByLookup` stitchers.

- The `GeoBySysLocation` stitcher calls a PostLayerProcessing stitcher `AddGeoLocationData`. The `AddGeoLocationData` stitcher extracts the location description, longitude, and latitude from the `SysLocation` field during a network discovery. The location data populates the `m_ExtraInfo->geographicLocation` field in the `workingEntities.finalEntity` database table to build the geographic location collection.
- The `GeoByLookup` stitcher calls a DNCIM InferDNCIMObjects stitcher `PopulateDNCIM_CustomGeography`. The `GeoByLookup` stitcher builds a hierarchy of geographic regions-> geographic locations-> devices, by resolving the geographical data from an external database. The `GeoByLookup` stitcher can be called by a customer stitcher such as the example `ACMEDeviceLocationEnrich` stitcher. The example `ACMEDeviceLocationEnrich` stitcher resolves IP addresses to geographical locations to create a hierarchy of Address -> City -> State -> Country.

Restriction:

Ensure that each level in a hierarchy is unique. The following examples are incorrect because the city and state values are the same:

```
IP, ADDRESS, CITY, STATE, COUNTRY, LATITUDE, LONGITUDE
192.168.0.1,"113620 Redwood Gulch Rd, Cupertino, CA 95014,
 USA",PLAL,PLAL,US,37.15458,-122.05561
192.168.0.2,"113620 Redwood Gulch Rd, Cupertino, CA 95014,
 USA",CA,CA,US,37.15458,-122.05561
```

The following example is incorrect because the city and country values are the same.

```
IP, ADDRESS, CITY, STATE, COUNTRY, LATITUDE, LONGITUDE
Redwood Gulch Rd, Cupertino, CA 95014,
USA",US,CA,US,37.15458,-122.05561
```

The following example is incorrect because the PLAL value was used previously as a city and is used here as a state. CA was used as a state and is used here as a country.

```
IP, ADDRESS, CITY, STATE, COUNTRY, LATITUDE, LONGITUDE
192.168.0.3,"113620 Redwood Gulch Rd, Cupertino,
CA 95014, USA",Redwood,PLAL,CA,37.15458,-122.05561
```

The following example is correct, because the geographical hierarchy is properly ordered and contained.

```
IP, ADDRESS, CITY, STATE, COUNTRY, LATITUDE, LONGITUDE
192.168.0.1,"113620 Redwood Gulch Rd, Cupertino,
CA 95014, USA",PLAL,CA,US,37.15458,-122.05561
```

Incorrect hierarchies are not included in the geographical views. The PopulateDNCIM_CustomGeography stitcher removes incorrect hierarchies from the topology and logs each occurrence in the ncp_disco logs with an error similar to the following:

```
Cyclic collection detected : Entity with entityId
123 should not be collected with collectingEntityId 8912
```

Fix Pack 3 *Example: Configuring geographical discovery by using location data from device records*

One way to enrich devices with geographical data is by using location data from the device records.

About this task

The AddGeoLocationData stitcher is provided as a demonstration of one method of adding geographic information from device records. This stitcher assumes that valid geographical data was added to the device record in the SysLocation field in the following format: *location address*;*longitude*;*latitude*. Your network administrator can set the SysLocation field on devices manually or by using a script. To use a different field, modify the stitcher.

To configure discovery to use geographical data from the SysLocation, complete the following steps.

Procedure

1. Back up and edit the following stitcher: $NCHOME/precision/disco/stitchers/PostLayerProcessing.stch.
2. Add the following line at the appropriate place in the file for your discovery data flow, for example, at the end:

   ```
   ExecuteStitcher('AddGeoLocationData');
   ```

3. Run a discovery.

   The AddGeoLocationData stitcher adds geographic data to the appropriate fields within the m_ExtraInfo field.

Fix Pack 2 *Example: configuring geographical discovery by using location data from a file*

One way to enrich devices with geographical data is by importing location data from a comma-separated values (.csv) file into the NCIM topology database.

About this task

To enrich devices with geographic data by using a .csv file, adapt the following example steps to your needs.

Procedure

1. Create a .csv file that contains location information for the devices you want to view in geographical views. For example, you can export the data from a database to file.

The location information must be in the following format:

```
ip,address,city,state,country,latitude,longitude
```

The following example shows the top two lines of a `.csv` file.

```
IP, ADDRESS, CITY, STATE, COUNTRY, LATITUDE, LONGITUDE
192.168.0.1,"113620 Redwood Gulch Rd, Cupertino, CA 95014, USA",
PLAL,CA,US,37.15458,-122.05561
```

The sample `ACMEDeviceLocationEnrich.stch` stitcher expects the `.csv` file to be in this format. If you want to use a different format, you must modify the stitcher.

2. Create a database table in the NCIM topology database to store geographical data.

   The sample `ACMEDeviceLocationEnrich.stch` stitcher expects this table to be called `ACMEGeoLocation`. If you want to use a different table name, you must modify the stitcher.

   For Db2, the database table must contain fields of the following types and specifications:

   ```
   IP VARCHAR(255) NOT NULL,
   ADDRESS VARCHAR(255),
   CITY VARCHAR(255),
   STATE VARCHAR(255),
   COUNTRY VARCHAR(255),
   LATITUDE DECIMAL(10 , 8) NOT NULL DEFAULT 0,
   LONGITUDE DECIMAL(11 , 8) NOT NULL DEFAULT 0
   ```

   For Oracle, the database table must contain fields of the following types and specifications:

   ```
   IP VARCHAR(255) NOT NULL,
   ADDRESS VARCHAR(255),
   CITY VARCHAR(255),
   STATE VARCHAR(255),
   COUNTRY VARCHAR(255),
   LATITUDE NUMBER(15,8) DEFAULT 0  NOT NULL,
   LONGITUDE  NUMBER(15,8) DEFAULT 0  NOT NULL
   ```

3. Import the geographical data from the `.csv` file into the database table you created using your choice of database tool.

   For example, to load a file `core_lat_long_all.csv` into a table `ACMELOOKUPGEOLOCATION` in an NCIM database on Oracle by using Oracle loader V12.1, run the following command from the Oracle directory.

   ```
   sqlldr SYSTEM/PASSWORD@mySchema control=/opt/oracle/load.ctl
   ```

   Where the `load.ctl` file contains the following code.

   ```
   LOAD DATA
       infile 'core_lat_long_all.csv'
       REPLACE
       INTO TABLE NCIM.ACMELOOKUPGEOLOCATION
       fields terminated by ',' optionally enclosed by '"'
       (
    IP,
    ADDRESS,
    CITY,
    STATE,
    COUNTRY,
    LATITUDE,
    LONGITUDE
       )
   ```

   The commands for other versions or other tools are different.

4. Back up and edit the following stitcher: `$NCHOME/precision/disco/stitchers/DNCIM/InferDNCIMObjects.stch`.
5. Uncomment this line:

   ```
   ExecuteStitcher('ACMEDeviceLocationEnrich', domainId,
   ```

```
isRediscovery, dynamicDiscoNode );
```

6. Run a discovery.

Results

The `PopulateDNCIM_CustomGeography.stch` stitcher adds devices to the correct geographical collections based on their location data.

The `ACMEDeviceLocationEnrich.stch` stitcher populates the geographic tables in the DNCIM discovery database with data from the NCIM database table that you created.

Fix Pack 3 Integrating an online or offline Web Map Service

You can set up a WMS (Web Map Service) integration to display geographical maps. Some WMS integrations can display maps without internet access.

About this task

To integrate a WMS, complete the following steps:

Procedure

1. Install a mapping provider that implements the WMS standard. Refer to the documentation for the version of the provider that you are installing.
2. Configure the mapping provider to use SSL.
3. Run the mapping provider and verify that it is working correctly.

   You might want to cache a public WMS as a test.
4. Configure a base mapping layer to use a WMS mapping provider.
   a) Back up and edit the following file: `$JazzSM_HOME/profile/installedApps/JazzSMNode01Cell/isc.ear/ncp_gis.war/resources/config.json`
   b) Configure a new section within the `baseLayersDefn` section and specify the correct parameters.

      The WMS parameters that are accepted by OpenLayers are listed at http://openlayers.org/en/latest/apidoc/ol.source.TileWMS.html. The parameters that you must use are defined in the WMS request specification of your WMS server.

      Use the following syntax for parameters: `"PARAM_NAME":"PARAM_VALUE"`. To avoid errors, use a validation tool to validate the JSON.
   c) To configure the new base layer as the default, set the `baseLayer` value to the `baseLayerName` of the new base layer.

   The following example configures parameters for a GeoServer integration.

```
"baseMapLayerName": "IBM Network Management",
    "baseLayer": "My Custom GeoServer WMS Layer",
    "baseLayersDefn": [
        {
            "baseLayerName": "My Custom GeoServer WMS Layer",
         "baseLayerType": "wms",
            "url": "https://offline:443/geoserver/wms",
            "params": {
                "REQUEST": "GetMap",
                "FORMAT": "image/png",
                "SRS": "EPSG:4326",
                "BBOX": "-104.2822265625,33.5302734375,-87.4072265625,40.78125",
                "VERSION": "1.1.1",
                "STYLES": "",
                "SERVICE": "WMS",
                "WIDTH": "768 ",
                "HEIGHT": "330",
                "TRANSPARENT": "true",
                "LAYERS": "topp:states"
            }
```

```
        .....
        }
```

If you do not specify any parameters in the `"params": {}` section, leave the section empty.

Important: Do not delete any empty sections from the `config.json` file. Missing sections can cause errors.

For offline map integration, the `baseLayerType` value must be `wms`.

Specify any unique descriptive name for the `baseLayerName`.

Adding custom map layers

You can add custom layers to geographical maps. For example, you can add geographical information that is made available by a public server.

Before you begin

Configure an offline mapping provider, if you want the custom layer to be available offline. See "Integrating an online or offline Web Map Service" on page 362 for more information.

About this task

Fix Pack 4 You can define one or more custom layers and one base layer. Users can select different custom layers for their session in the geographical views.

The following mapping provider types are supported:

- ArcGISRest
- Web Map Service (WMS)
- Web Mapping Tile Service (WMTS)

To add a custom map layer, complete the following steps:

Procedure

1. Back up and edit the following file: `$JazzSM_HOME/profile/installedApps/JazzSMNode01Cell/isc.ear/ncp_gis.war/resources/config.json`
2. Create a new `customLayers` section or uncomment one of the default sections, and specify the correct parameters.

   The WMS parameters that are accepted by OpenLayers are listed at http://openlayers.org/en/latest/apidoc/ol.source.TileWMS.html. The parameters that you must use are defined in the WMS request specification of your WMS server. Use the following syntax for parameters: `"PARAM_NAME":"PARAM_VALUE"`. To avoid errors, use a validation tool to validate the JSON.

   The ArcGISRest parameters that are accepted by OpenLayers are listed at https://openlayers.org/en/latest/apidoc/module-ol_source_TileArcGISRest-TileArcGISRest.html.

   The WMTS parameters that are accepted by OpenLayers are listed at https://openlayers.org/en/latest/apidoc/module-ol_source_WMTS.html.

   The following example configures parameters for one WMS custom layer, one ArcGISRest custom layer, and one WMTS custom layer.

```
"customLayers": [
   {
       "layerName": "Drainage Divisions",
       "layerType": "wms",
       "url": "http://geoserver.nationalmap.nicta.com.au/
admin_bnds_abs/ows",
       "params": {
          "LAYERS": "admin_bnds:ADD_2011_AUST"
       }
    "selected":"true"
```

```
   }, {
       "layerName": "Prohibited Areas",
       "layerType": "arcGISRest",
       "url": "http://services.ga.gov.au/gis/rest/services/
NM_Reserves/MapServer",
       "params": {}

    "selected":"false"
    }
       {
       "layerName": "New Zealand Earthquakes",
       "layerType": "wmts",
       "capabilities": {
             "url": "https://openlayers.org/en/v4.3.4/examples
/data/WMTSCapabilities.xml",
             "layer": "layer-7328",
             "matrixSet": "EPSG:3857"
             }
    "selected":"false"
       }
]
```

If you do not supply any `arcGISRest` parameters, include the line but leave it empty: `"params": {}`. Removing or omitting sections from the file causes errors.

The `layerName` parameter can be set to any unique descriptive value.

For custom map layers, the `layerType` value must be one of the following supported values:

- `arcGISRest`
- `wms`
- `wmts`

Custom layers that have the `selected` parameter set to `true` are displayed by default in the geographical views. The user can show or hide any custom layers.

3. Define multiple layers if you want. The z-index of the layers is defined by their position in the file. To display one custom layer above another layer, define it higher in the file. The topology layer is displayed on top of the other layers, and the base layer is displayed underneath the other layers.

Fix Pack 2 Verifying geographical discoveries

After you run a geographical discovery, verify that it ran successfully.

About this task

To check that the geographical discovery ran successfully, complete the following steps:

Procedure

1. Check the logs for the **ncp_disco** process in the `$NCHOME/log/precision` directory to find out if the geographical stitchers that you configured are working.
2. Query the location tables in the NCIM topology database to find out if they are populated with geographic data.

   For example, use the following command:

   ```
   select * from NCIM.geographicLocation
   ```

3. Log into Dashboard Application Services Hub. Open the following URL:

   ```
   https://server:port/ibm/console/nm_rest/topology/devices/geo/all
   ```

   If geographical discovery completed successfully, JSON results for the geographically enriched devices are displayed.
4. Open a geographic view and check that the expected devices appear.

Fix Pack 4 Configuring IP SLA discoveries

Before you can monitor IP SLA agreements, you must configure and run an IP SLA discovery.

About this task

An IP SLA discovery associates supported IP SLA events with devices and links in the Network Manager topology. You can visualize probes that are configured to monitor IP SLA response times, and their source and destination devices, by using network views.

Procedure

1. Install the Netcool/OMNIbus Knowledge Library version 4.8.1 or above.
2. Configure the Netcool/OMNIbus Knowledge Library for use with the SNMP Probe.

   This step allows event status to be associated with IP SLA Probes in the network topology.
3. Enable the appropriate IP SLA agent.

   This step allows IP SLA Probes to be discovered. For more information on the IP SLA agents available, see *Service Level Agreement agents* in the *IBM Tivoli Network Manager Reference*.
4. Run a discovery.

What to do next

You can now monitor IP SLA response times by using the network views, **Network Hop View**, and **Structure Browser**. See *Monitoring IP Service Level Agreement (IP SLA) configurations* in the *IBM Tivoli Network Manager User Guide*.

Chapter 14. Monitoring network discoveries

You can monitor the state and progress of your network discoveries. In addition you can monitor full discovery and partial discovery by running OQL queries from the command line.

Monitoring network discovery from the GUI

From the **Active Discovery Status** page, you can monitor the status and progress of the current full or partial discovery, investigate the work of the discovery agents, and view details of the last full discovery.

About this task

From the **Active Discovery Status** page, you can also start and stop full and partial discoveries.

Monitoring full and partial discoveries

You can monitor the state and progress of your full or partial discovery using the GUI.

Monitoring full and partial discovery progress

You can use the **Monitoring** tab to monitor the progress of the current full or partial discovery through each of the discovery phases.

About this task

Complete the following steps to monitor the progress of the current full or partial discovery.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Status**.
2. Select a domain.
3. Click the **Monitoring** tab.
4. Start either a full or partial discovery by selecting the option from the Start Discovery button  .

Results

The following phases are shown in the table.

Interrogating Devices
During this phase, devices are first discovered by the finders, and then information is retrieved from the devices by the agents. This phase is also known as phase 1.

Resolving Addresses
During this phase, the agents resolve IP to MAC address translations. This phase is also known as phase 2.

Downloading Connections
During this phase, the switch agents download the forwarding tables from the switches in the network. This phase is also known as phase 3.

Correlating Connections
During this phase, the connectivity between the devices is calculated, the containment model is created, and the network topology is built. This phase is also known as phase -1.

You can see which phase the current discovery is in by looking at the **Status** column of the table. If a phase has not started, this column is empty. If a phase is in progress, this column shows a spinning wheel icon. If a phase has completed successfully, this column shows a green tick icon.

© Copyright IBM Corp. 2006, 2021

Status
Shows the status of a particular phase. The column shows the following kinds of status.

Table 65. Discovery phase status

State	Icon	Description
Completed		If a phase has completed successfully, this column shows a green tick icon.
In progress		If a phase is in progress, this column shows a spinning wheel icon.
Not started		If a phase has not started, this column is empty.

You can see how long each phase is taking in the **Elapsed Time** column in the table. Each phase takes a different amount of time depending on the scope of the discovery, the complexity of the network, and the amount of detail being retrieved from the devices. If the elapsed time continues to increase, and the work completed does not increase, the discovery might be encountering problems.

Remember: In the first phase, the count of IP addresses discovered stops increasing part way through the phase. This is part of the normal operation of the discovery. The count of IP addresses discovered only increases during the first part of the phase, while the finders discover new devices. In the latter part of the phase, the discovery agents retrieve information from these devices, and new IP addresses are not discovered.

The **Discovery Agents** section shows the progress of the discovery agents. If you think that a phase is taking too long to complete, click the **Discovery Agents** tab to see what the discovery agents are doing.

You can see the progress within a phase in the **Work Completed** column in the table. For the first phase, this column shows the number of IP addresses found so far. For the other phases, this column shows the percentage of work completed in the phase.

Comparing full discoveries

You can use the **Monitoring** tab to compare the current discovery to the previous full discovery.

About this task

You cannot compare partial discoveries. Within the table, the data in the **Previous** columns is for the last full discovery. Complete the following steps to compare fully discoveries.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Status**.
2. Within **Network Discovery Status**, click the **Monitoring** bar.

Results

You can see the time that is taken to complete each phase of the previous discovery in the **Previous** subcolumn of the **Elapsed Time** column.

Note: To display discovery times from all previous discoveries, run the **disco_profiling_data.pl** script from the command line. For more information about the **disco_profiling_data.pl** script, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

The time that is taken for each phase depends on the scope of the discovery, the complexity of the network, and the amount of detail that is retrieved from the devices. If the network is not changed significantly, and the discovery scope and settings are not changed significantly, but the elapsed time for a phase in the current discovery is more than the time taken for the same phase in the previous discovery, the discovery might be encountering problems.

You can see how many IP addresses are being found in the current discovery and how many were found in the previous discovery in the **Work Completed** column in the table. If fewer IP addresses are found in the current discovery, there might be a problem with the scope of the discovery, or with SNMP access to devices.

Monitoring ping finder progress

You can use the **Ping Finder Status** table to monitor the progress of the ping finder during a full discovery.

About this task

To open **Ping Finder Status**, complete the following steps.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Status**.
2. Within **Network Discovery Status**, click the **Ping Finder Status** tab.

Results

You can use the **Ping Finder Status** table to see which IP addresses and subnets are discovered up to this point. If the ping finder is processing a subnet, you can also see which IP address was last pinged.

The **Ping Finder Status** table contains the following information:

Address
A list of IPs and subnets discovered to this point.

Netmask
For each subnet, this column indicates the netmask value.

Last Pinged
The last IP address pinged.

Status
Indicates whether the Ping finder is still pinging this device or subnet or whether it has completed pinging.

Table 66. Ping finder status

State	Icon	Description
Completed		Ping finder has completed the pinging of this subnet or IP address.
Started		Ping finder is currently pinging this subnet or IP address.
Stopped		Ping finder has not started pinging this subnet or IP address.
Awaiting status		System is awaiting Ping finder status for this subnet or IP address.

Monitoring discovery agent progress

You can use the **Agents Status** section to monitor the progress of the discovery agents during a full or partial discovery.

About this task

Discovery agents gather data from discovered devices. This data is used during the Correlating connectivity phase of the discovery (phase -1) to build the network connectivity and containment.

You can use the **Agents Status** to answer these and other questions as the discovery is running:

- Are all agents running okay?

- Have any agents failed?
- Are any agents failing to complete?
- Which device is a particular agent currently working on?

Complete the following steps to open **Agents Status** and to monitor the progress of the discovery agents.

Procedure

1. Click the **Discovery** icon and select **Network Discovery Status**.
2. Click the **Agents Status** tab.

   The **Agents Status** section contains two tables, the **Agents Status** table at the top and the **Entity Status** table below. The **Agents Status** table toolbar contains the following controls.

   Filter Agents by Phase
   Use the phase drop-down list to select a discovery phase. The agents table then displays the following agents:

   - *Full or partial discovery*: all discovery agents that have started during the current discovery and that are scheduled to finish in the discovery phase that you selected.

   Refresh 
   Refreshes the data in both the Agents Status and Entity Status table. The icon changes to the **Refreshing** icon  while the table data is being refreshed. You cannot refresh the tables again until the refresh has completed.

   The **Agents Status** table lists all the agents that have started so far during this discovery and contains the following information. This information is updated every 20 seconds. When you first open this table, it is sorted by descending order of **State**.

   Agent
   Discovery agents that have started during the current discovery and that are scheduled to finish in the discovery phase that you selected.

   Completes in Phase
   The phase in which the discovery agent completes.

   State
   Current state of the discovery agent. Possible states, in the default descending order, are listed in the following table.

   Table 67. Agent states

State	Value	Icon	Description
Died	5		The agent has terminated unexpectedly. This is a potential discovery problem.
Finished	4		The agent is still running but has finished processing of all the entities in its queue. The agent is still available to process any further agents placed in the queue.
Running	3		The agent is currently processing entities.
Starting	2		The agent is starting up.
Not running	1		The Agent is not running.

   Total Entities
   The total number of entities that this agent is required to process. This number varies as follows:

- *Full or partial discovery*: the total number of entities that this agent is required to process increases as the discovery progresses and the finders discover more devices that need to be processed by the agent.

Outstanding Entities
The number of entities that are waiting to be processed by this agent. This number can go up and down during the discovery. The number increases initially as the discovery progresses and the finders discover more devices that need to be processed by the agent. As the agent completes processing entities, this number decreases until it reaches zero.

Note: If this value does not go down to zero during the discovery, this means that the agent was unable to complete processing on one or more entities and there is a potential discovery problem.

3. Click an agent in the **Agents Status** table.

The **Entity Status** table lists the entities that have been processed or are currently being processed by this agent. The **Entity Status** table responds to changes in the **Agents Status** table. The table updates in the following situations: when a new agent is selected in the **Agents Status** table; when changing the filtering of the **Entity Status** table by **All** or **Queue**; and when the **Agents Status** table **Refresh** button is pressed. When you first open this table, it is sorted by descending order of **State**.

Agent_name
Use this radio button to specify whether to display all entities (All) or only entities queued for processing (Queue). The default setting is Queue.

All
Set the **Details** table to display all entities for this agent. This includes entities that have been queued for processing by the agent, entities currently being processed by the agent, and entities that have already been processed by the agent.

Queue
Set the **Details** table to display only entities that have been queued for processing by this agent.

Identifier
Identifier for entities processed by this agent. If **All** is selected, then this column displays entities processed, in processing, or queued for processing by this agent. If **Queued** is selected, then this column displays entities queued for processing by this agent.

State
Current state of the entity. Possible states, in the default descending order, are listed in the following table:

Table 68. Entity states

State	Value	Icon	Description
Died	5		Processing of the entity terminated unexpectedly. Either the agent was stopped manually, or there is a discovery problem.
Finished	4		An agent has completed processing this entity.
Running	3		An agent is currently processing this entity.
Starting	2		An agent is beginning to process this entity.
Not running	1		This entity is not currently being processed.

Elapsed Time
The time taken for the agent to process this entity, expressed in the format HH:MM:SS. This value is only displayed for those entities that have completed processing.

Despatch Time
The date and time at which the agent began processing this entity. This value is only displayed for those entities for which processing has begun or completed.

Return Time
The date and time at which the agent retrieved data for this entity. This value is only displayed for those entities that have completed processing.

SNMP Access
Indicates whether the agent was able to access this entity using SNMP.

Monitoring discovery progress from the command line

When the **ncp_disco** process is running, you can monitor the progress of the discovery by using the OQL Service Provider, the **ncp_oql** process, to query the discovery databases to determine what is happening at any time.

Monitoring full and partial discoveries

You can monitor the state and progress of your full or partial discovery using the command line.

About this task

The queries presented in the subsequent topics have been generalized for all discovery scenarios and are not limited to the layer 3 discovery.

The examples are given only to demonstrate the amount of flexibility you have when retrieving information from databases using OQL. Using the schematic definitions of all the databases and knowledge of OQL syntax, you can construct queries that answer any questions you have regarding the current status of the discovery process.

You can issue simple queries to find out, for example, what the **ncp_disco** process is currently doing, which discovery agents have discovered devices, or how many devices have been discovered so far. You can also issue complex queries to find out, for example, which devices have been discovered by a specific discovery agent, or which discovery agents have interrogated a specific device.

For information on starting the OQL Service Provider, including prerequisites, see the *IBM Tivoli Network Manager Reference*.

Sample discovery status queries

You can use queries similar to these examples to find out the status of different parts of the discovery.

Sample: Determining which address the Ping finder is pinging

The following query returns the current address being pinged by the Ping finder:

```
select m_CurrentAddress from pingFinder.status;
go
.
{
                m_CurrentAddress=192.168.0.1;
}
```

Sample: Identifying the current phase of the discovery

The following example shows how to identify the current phase of the discovery. The results of the above query show that the discovery process is still in data collection phase 1.

```
select * from disco.status;
go
.
{
                m_DiscoveryMode=0;
```

```
                m_Phase=1;
                m_BlackoutState=0;
                m_CycleCount=0;
                m_ProcessingNeeded=0;
                m_FullDiscovery=0;
}
```

Sample: Identifying the status of a NAT discovery

This example shows how to identify the status of the NAT discovery.

```
select m_NATStatus from disco.NATStatus;
go
.
{
                m_NATStatus=3;
}
```

Sample: Identifying which agents are enabled

This example shows how to identify whether you have enabled the appropriate discovery agents.

```
select m_AgentName, m_Valid from disco.agents
where m_Valid = 1;
go
...
{
                m_AgentName='Details';
                m_Valid=1;
}
{
                m_AgentName='AssocAddress';
                m_Valid=1;
}
{
                m_AgentName='IpRoutingTable';
                m_Valid=1;
}
{
                m_AgentName='IpForwardingTable';
                m_Valid=1;
}
```

Sample: identifying the status of the discovery stitchers

The following example shows how to identify the status of the stitchers by querying the stitchers.status table.

```
select * from stitchers.status
where m_State > 0 ;
go
.........
{
                m_Name='AgentRetToInstrumentationSubnet';
                m_State=3;
}
{
                m_Name='DetailsRetProcessing';
                m_State=3;
}
.....
.....
{
                m_Name='DetectionFilter';
                m_State=3;
}
{
                m_Name='FnderProcToDetailsDesp';
                m_State=3;
}
{
                m_Name='FnderRetProcessing';
```

```
                m_State=3;
}
```

The results from the query show the current status of all stitchers that have been called by the discovery process so far. Note that the results shown above have been abbreviated.

Sample: identifying which agents are active

The following example shows how to query the status of the agents in the agents database.

```
select * from agents.status
where m_State > 0 ;
go
..
{
                m_Name='Details';
                m_State=3;
                m_NumConnects=1;
}
{
                m_Name='IpRoutingTable';
                m_State=3;
                m_NumConnects=1;
}
```

The results of the above query show that only the Details agent and the IpRoutingTable agent are active (that is, they have a state greater than zero).

Sample device queries

You can use queries similar to these examples to identify devices that meet certain criteria, for example, devices that have been found by the finders.

Sample: Identifying which devices have been found by the finders

The following example shows how to identify devices that have been found by the finders.

```
select * from finders.processing;
go
....
{
                m_UniqueAddress='172.20.12.253';
                m_Protocol=1;
                m_Creator='IpRoutingTable';
}
{
                m_UniqueAddress='172.20.22.61';
                m_Protocol=1;
                m_Creator='IpRoutingTable';
}
{
                m_UniqueAddress='172.20.0.221';
                m_Protocol=1;
                m_Creator='IpRoutingTable';
}
{
                m_UniqueAddress='10.10.35.17';
                m_Creator='PingFinder';
}
```

The above query shows the devices discovered by the Ping finder as well as devices reported as a result of connections discovered by the `IpRoutingTable` Discovery agent.

Sample: Identifying devices that have been sent to the Details agent

The following example shows how to identify devices that have been sent to the Details agent.

```
select * from Details.despatch;
go
............................................................
.............................
```

```
{
                m_UniqueAddress='10.10.38.82';
}
{
                m_UniqueAddress='10.10.38.83';
}
.....
.....
{
                m_UniqueAddress='10.10.38.84';
}
{
                m_UniqueAddress='10.10.38.87';
}
{
                m_UniqueAddress='10.10.38.88';
}
{
                m_UniqueAddress='10.10.38.89';
}
{
                m_UniqueAddress='10.10.38.90';
}
```

Sample: Identifying devices that have been returned from the Details agent

To identify which devices have returned from the Details agent, query the `returns` table of the Details agent, as shown below.

```
select * from Details.returns;
go
.............................................................
.............................
{
                m_UniqueAddress='10.10.8.255';
                m_UpdAgent='Details';
                m_HaveAccess=1;
                m_Description='Ascend Max-HP T1/PRI S/N;
                m_ObjectId='1.3.6.1.4.1.529.1.2.6';
                m_LastRecord=1;
}
{
                m_UniqueAddress='10.10.9.1';
                m_UpdAgent='Details';
                m_Name='minotaur.Kazeem.San.COM';
                m_HaveAccess=0;
                m_LastRecord=1;
}
.....
.....
{
                m_UniqueAddress='10.10.9.2';
                m_UpdAgent='Details';
                m_Name='cyclops.Kazeem.San.COM';
                m_HaveAccess=0;
                m_LastRecord=1;
}
{
                m_UniqueAddress='10.10.9.3';
                m_UpdAgent='Details';
                m_Name='centaur.Kazeem.San.COM';
                m_HaveAccess=0;
                m_LastRecord=1;
}
```

Sample: Identifying all devices discovered until now

The following example shows how to identify all known network entities.

```
select m_Name, m_ObjectId, m_UniqueAddress
from workingEntities.finalEntity;
go
..............................
{
                m_Name='10.10.8.255';
                m_ObjectId='1.3.6.1.4.1.529.1.2.6';
```

```
                m_UniqueAddress='10.10.8.255';
}
{
                m_Name='minotaur.Kazeem.San.COM';
                m_UniqueAddress='10.10.9.1';
}
.....
.....
{
                m_Name='cyclops.Kazeem.San.COM';
                m_UniqueAddress='10.10.9.2';
}
```

Sample: Identifying which agents have discovered devices

The following example shows how to identify the agents that have discovered devices.

```
select m_Name, m_Creator
from workingEntities.finalEntity;
go
..............................
{
                m_Name='b11-m1-2611.Kazeem.San.COM[ 0 [ 2 ] ]';
                m_Creator='IpRoutingTable';
}
{
                m_Name='b-ayo.Kazeem.San.COM';
                m_Creator='Details';
}
{
                m_Name='b11-m1-2611.Kazeem.San.COM[ 0 [ 1 ] ]';
                m_Creator='IpRoutingTable';
}
.....
.....
{
                m_Name='b11-m1-2611.Kazeem.San.COM';
```

Sample: Determining the different types of interfaces discovered

The following example shows how to identify the interface types on each discovered device. Use the `select DISTINCT` keyword to deduplicate the multiple interface type entries that are stored for each device in the `workingEntities.finalEntity` table.

```
select DISTINCT m_LocalNbr->m_IfType, m_ObjectId
from workingEntities.finalEntity
where m_EntityType = 2;
go
..............................
{
        m_IfType=166;
        m_ObjectId='1.3.6.1.4.1.9.1.222';
}
{
        m_IfType=6;
        m_ObjectId='1.3.6.1.4.1.9.1.222';
}
{
        m_IfType=1;
        m_ObjectId='1.3.6.1.4.1.9.1.222';
}
{
        m_IfType=6;
        m_ObjectId='1.3.6.1.4.1.9.1.310';
}
{
        m_IfType=22;
        m_ObjectId='1.3.6.1.4.1.9.1.310';
}
..............................
```

Sample: Monitoring agents in the current phase

Use the following example query to identify which agents the current phase is waiting on before it can complete. The following example query does this by listing the names of all the agents that finish in the current phase and that meet the following criteria:

- The agent is valid. (`m_State <> 0`)
- The agent is currently in use. (`m_State <> 1`)
- The agent status is not yet complete. (`m_State <> 4`)

```
select m_Name from agents.status
where
m_State <> 0 AND
m_State <> 1 AND
m_State <> 4 AND
m_CompletionPhase IN (( select m_Phase from disco.status )) ;
```

Once you have identified which agents still need to complete in the current phase, use the following query to determine which devices those agents are working on.

```
select
    m_Name,
    m_UniqueAddress,
    m_ObjectId,
    m_Description
from
    <agentName>.despatch
where
    m_UniqueAddress NOT IN
    (( select m_UniqueAddress from <agentName>.returns where m_LastRecord = 1 )) ;
```

Sample network entity queries

You can use queries on the instrumentation database to identify whether network entities such as subnets and VLANs have been discovered. The instrumentation database tables store a record of every discovered device.

Sample: Identifying the number of discovered subnets

The following example query returns details of the discovered subnets.

```
select * from instrumentation.subNet;
go
.....................................
{
                m_SubNet='172.20.67.0';
                m_NetMask='255.255.255.0';
}
.....
.....
{
                m_SubNet='172.20.70.0';
                m_NetMask='255.255.254.0';
}
{
                m_SubNet='172.20.95.0';
                m_NetMask='255.255.255.0';
}
( 81 record(s) : Transaction complete )
```

Sample: Identifying discovered VLANs

The following example query returns details of the discovered VLAN IDs.

```
select * from instrumentation.vlan;
go
.....................................
{
                m_Vlan=23;
```

```
}
{
                m_Vlan=65;
}
.....
.....
{
                m_Vlan=677;
}

( 4826 record(s) : Transaction complete )
```

Sample complex discovery queries

You can use queries similar to these examples to identify devices that meet certain criteria, for example, devices that have been found by particular discovery agents.

Identifying devices that have been sent to a specific agent

The following example query identifies devices that have been sent to the IpRoutingTable agent.

```
select m_Name, m_ObjectId, m_Description
from IpRoutingTable.despatch;
go
..............................
{
       m_Name='10.10.63.193';
       m_ObjectId='1.3.6.1.4.1.9.1.108';
       m_Description='Cisco Internetwork Operating System Software
IOS (tm) 7200 Software (C7200-JS-M), Version 12.0(4)T,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by Cisco Systems, Inc.
Compiled Thu 29-Apr-99 06:27 by kpma';
}
.....
.....
{
       m_Name='10.10.71.248';
       m_ObjectId='1.3.6.1.4.1.9.1.258';
       m_Description='Cisco Internetwork Operating System Software
IOS (tm) MSFC Software (C6MSFC-IS-M), Version 12.0(7)XE1, EARLY DEPLOYMENT
RELEASE SOFTWARE (fc1)
TAC:Home:SW:IOS:Specials b-ayo k-az-eem for info
Copyright (c) 1986-2000 by Cisco Systems, Inc.
Compiled Fri 04-Feb-00 00:';
}
```

Identifying devices that have been returned by a specific agent

The following example query identifies devices returned by the IpRoutingTable discovery agent.

```
select m_Name from IpRoutingTable.returns;
go
..............................
{
                m_Name='10.10.71.248';
}
.....
.....
{
                m_Name='10.10.71.248';
}
{
                m_Name='10.10.71.248';
}
```

Identifying additional devices that have been discovered by a specific agent

An agent can discover additional devices by interrogating a device. In this situation, the additional device would be in the `returns` table of that agent, but not the `despatch` table. You can identify which devices are present in the `IpRoutingTable.returns` table, but not in the `IpRoutingTable.despatch` table,

by performing a join between the `IpRoutingTable.despatch` and `IpRoutingTable.returns` tables, as in the following example.

```
select IpRoutingTable.returns.m_Name from
IpRoutingTable.returns, IpRoutingTable.despatch
where
IpRoutingTable.returns.m_Name <>
IpRoutingTable.despatch.m_Name;
go
.....................................
{
                m_Name='10.10.71.237';
}
.....
.....
{
                m_Name='10.10.71.55';
}
{
                m_Name='10.10.71.51';
}
```

Identifying the devices that an agent has enqueued

The following example returns those devices in the despatch table that have not yet been returned.

```
select * from <agent>.despatch
where
(
    m_UniqueAddress NOT IN
        (( select m_UniqueAddress from <agent>.returns where m_LastRecord = 1 ))
);
```

Sample queries for locating a specific device

To see whether a specific device has been discovered, you can use queries similar to these examples to search through the discovery data flow.

Sample: Identifying whether a device is present in the workingEntities database

The following example query determines if the device is present in the workingEntities database.

```
select * from workingEntities.finalEntity
where m_UniqueAddress ='10.10.63.239';
go
.
( 0 record(s) : Transaction complete )
```

Sample: Identifying whether a device has been returned from the AssocAddress agent

If the device is not present in the workingEntities database, you can use the following example query to determine if the device has been returned from the AssocAddress agent.

```
select * from AssocAddress.returns
where m_UniqueAddress = '10.10.63.239';
go
.
( 0 record(s) : Transaction complete )
```

Sample: Identifying whether a device has been returned from the Details agent

If the device has not been returned from the AssocAddress agent, you can use the following example query to determine if the device has been returned from the Details agent.

```
select * from Details.returns
where m_UniqueAddress = '10.10.63.239';
go
```

```
.
( 0 record(s) : Transaction complete )
```

Sample: Identify whether a device has been sent to the Details agent

If the device has not been returned from the Details agent, you can check if the device has been sent to the Details agent by querying the `Details.despatch` table, as shown below. This result indicates that the device has been sent to the Details agent, but has not yet been processed.

```
select * from Details.despatch
where m_UniqueAddress='10.10.63.239';
go
.
{
                m_UniqueAddress='10.10.63.239';
}
( 1 record(s) : Transaction complete )
```

Sample: Identifying whether a device has been discovered by the finders

If the device is not in the `Details.despatch` table, you can query the `finders` database, as shown below. This result shows that the device has been discovered by the finders.

```
select * from finders.processing
where m_UniqueAddress='10.10.63.239';
go
.
{
                m_UniqueAddress='10.10.63.239';
}
( 1 record(s) : Transaction complete )

select * from finders.returns
where m_UniqueAddress='10.10.63.239';
go
.
( 0 record(s) : Transaction complete )
```

Sample collector discovery queries

You can use queries similar to these examples to monitor devices discovered using the collectors.

Sample: Determining the directionality of connections between layer 1 devices discovered by collectors

The following example shows how to check that the collector has retrieved directionality of connections between devices. Allowed values for connection directionality are as follows:

- 0 for bidirectional (two-way link)
- 1 for unidirectional (one-way link)

```
select * from CollectorLayer1.returns;
go
....
{
               m_UniqueAddress='10.1.1.18';
               m_Name='OpticalNE1';
               m_ManagerId='localhost:8081_1';
               m_HaveAccess=1;
               m_Protocol=1;
               m_UpdAgent='CollectorLayer1';
               m_LocalNbr={
                       m_InterfaceId='STS 1';
                       m_LocalNbrName='OpticalNE1';
               };
               m_RemoteNbr={
                       m_RemoteNbrName='OpticalNE2';
                       m_InterfaceId='PTP 1';
                       m_Unidirectional=1;
               };
```

```
}

{
        m_UniqueAddress='10.1.1.20';
        m_Name='OpticalNE3';
        m_ManagerId='localhost:8081_1';
        m_HaveAccess=1;
        m_Protocol=1;
        m_UpdAgent='CollectorLayer1';
        m_LocalNbr={
                m_InterfaceId='STS 2';
                m_LocalNbrName='OpticalNE3';
        };
        m_RemoteNbr={
                m_RemoteNbrName='OpticalNE1';
                m_Unidirectional=1;
        };
}
```

This result fragment shows two devices.

Device 10.1.1.18

This device has a local neighbour named OpticalNE1 and a remote neighbour named OpticalNE2.

Device 10.1.1.18 has a unidirectional link to its remote neighbour OpticalNE2; that is, a one-way link in the direction 10.1.1.18 –> OpticalNE2.

Device 10.1.1.20

This device has a local neighbour named OpticalNE3 and a remote neighbour named OpticalNE1.

Device 10.1.1.20 has a unidirectional link to its remote neighbour OpticalNE1; that is, a one-way link in the direction 10.1.1.20 –> OpticalNE1.

Chapter 15. Classifying network devices

On completion of discovery, Network Manager automatically classifies all discovered network devices based on a predefined device class hierarchy. You can change the way network devices are classified.

Changing the device class hierarchy

Change the device class hierarchy to change the way network devices are classified. A common situation that requires a change to the class hierarchy is when the discovery process identifies an unclassified device, that is, a device that is not defined in the class hierarchy.

About this task

Following a discovery, you can check whether any devices are unclassified by running the following reports:

- Devices with Unclassified SNMP Object IDs report
- Devices with Unknown SNMP Object IDs report

Listing the existing device classes

Before you edit AOC definitions and reinstantiate the topology, list the device classes that are currently in use.

About this task

You list existing device classes by querying the ncp_model databases. The query returns the names of the classes to which devices in the current topology have been instantiated. Substitute your domain name and username where NCOMS and admin are specified.

Procedure

1. Log into the OQL Service Provider using the following command:

   ```
   ncp_oql -domain NCOMS -username admin -service ncim
   ```

   You can also issue this query using the Management Database Access page.
2. Specify the relevant password when prompted.
3. Type the following query:

   ```
   select * from entityClass;
   go
   ```

   The following table shows an example of the output of this query:

Table 69. Query results

Class identifier	Class name	Super class identifier	Class type	Manager name
1	Core		Core	PrecisionIP
2	EndNode	1	EndNode	PrecisionIP
1815493792	Brother	2	PhysicalHost	PrecisionIP
1899974822	BrotherPrinter	1815493792	Printer	PrecisionIP
1367768986	HPOfficePro85xx	2	Printer	PrecisionIP

© Copyright IBM Corp. 2006, 2021

Table 69. Query results (continued)

Class identifier	Class name	Super class identifier	Class type	Manager name
200	PhysicalHost	2	EndNode	PrecisionIP
163	AIX®	200	Router	PrecisionIP
3	HPPrinter	200	EndNode	PrecisionIP
202	HypervisorHost	200	HypervisorHost	PrecisionIP
209	PowerHyperHost	202	HypervisorHost	PrecisionIP
210	VMWareHyperHost	202	HypervisorHost	PrecisionIP
4	Linux	200	EndNode	PrecisionIP
117	NoSNMPAccess	200	EndNode	PrecisionIP
211	PowerVMControl	200	EndNode	PrecisionIP
129	Sun	200	EndNode	PrecisionIP
134	Windows	200	EndNode	PrecisionIP
161	zOS	200	EndNode	PrecisionIP
201	VirtualHost	2	VirtualMachine	PrecisionIP
207	VirtualAIXHost	201	VirtualMachine	PrecisionIP
206	VirtualLinuxHost	201	VirtualMachine	PrecisionIP
205	VirtualSolarisHost	201	VirtualMachine	PrecisionIP
204	VirtualWindowsHost	201	VirtualMachine	PrecisionIP
138	InferredDevice	1	NetworkDevice	PrecisionIP
85	InferredCE	138	Router	PrecisionIP
86	InferredHub	138	Switch	PrecisionIP

Creating and editing AOC files

Create and edit AOC files to classify unclassified devices or to change the class hierarchy of your topology.

About this task

If the discovery process identified an unclassified device, you can classify the device by creating a new AOC file that is specific to the device class to which this device belongs.

You can edit AOCs in either of two ways: update the **ncp_class** databases, or modify the AOC file definitions:

- If you want to edit the current AOC definitions by updating the **ncp_class** database directly, then use the **Management Database Access** or the OQL Service Provider.
- If you want to modify the AOC file definitions, then complete the following steps.

Procedure

1. Go to the NCHOME/precision/aoc directory.
2. Back up any files that you want to edit.
3. Create a text file or edit an existing AOC file by using a text editor.

Restriction: Only alphanumeric characters and the underscore (_) character may be used for AOC filenames. Any other characters, for example the hyphen (-) are forbidden.

4. Refer to the AOC syntax and structure in the *Active Object Class files* in the *IBM Tivoli Network Manager Reference* to construct or edit your AOC file. Consider adapting an existing AOC file for a similar device type.
5. To apply the AOC to a particular type of device, you must edit the instantiate rule in the AOC file. Use one of the following methods to restrict the AOC to your chosen device type:
   - If the device type can be uniquely identified by using the EntityOID, then use EntityOID in the instantiate rule. Most of the default AOC files use this method.
   - If the device type can be uniquely identified by using any of the following fields from the `workingEntities.finalEntity` database table, then use its alias in the instantiate rule.

     Table 70. workingEntities.finalEntity database table mappings

workingEntities.finalEntity	**Alias**
m_Name	EntityName
m_ObjectId	EntityOID
m_HaveAccess	IsActive
m_Description	Description
m_EntityType	EntityType

     For example, the `Tellabs63xx.aoc` file uses the alias `Description` to reference the `m_Description` field: `instantiate_rule = "Description = 'Tellabs 6340_OLD'`.
   - If the device type can be uniquely identified using a different field from the `workingEntities.finalEntity` database table, then specify the field name. For example, the `VirtualHost.aoc` file uses the `ExtraInfo->m_IsVirtualMachine` field in the `workingEntities.finalEntity` database table: `instantiate_rule = "ExtraInfo->m_IsVirtualMachine = 1"`.
6. If you created a new AOC file, add an insert to the `class.classIds` database table in the `ClassSchema.cfg` configuration file.
7. Edit the startup options for the **ncp_class** process and set the `-read_aocs_from` option to ensure that the new or changed AOC files are read.
8. Restart the **ncp_class** process after you change the AOC files. After **ncp_class** is restarted and running, restart the **ncp_disco** process.
9. Ensure that a domain-specific version of any new AOC files is present in the `NCHOME/precision/aoc` directory.
10. Back up and remove the class cache files in the `NCHOME/var/precision` directory.

    For example, remove the following cache files:

    ```
    Class.Cache.class.activeClasses.NCOMS
    Class.Cache.class.staticClasses.NCOMS
    ```

11. Run a full discovery and check that the results match the changes you made.

Applying AOC changes to the topology and to the reports

After you have updated the AOC definitions and passed the changes to **ncp_class**, you can apply the changes to the topology by waiting for the next discovery to complete or by restarting the discovery at the point when the topology is passed from ncp_disco to ncp_model.

About this task

When the next full discovery completes, the AOC changes that you made are automatically applied to the network topology.

If you do not want to wait for the next full discovery, use the appropriate stitcher to restart the discovery at the required point. To re-instantiate the containment model, you must start the stitcher that sends the topology from **ncp_disco** to **ncp_model**.

Procedure

1. Log into the OQL Service Provider or access the **Management Database Access**.
2. Issue the following query to the `disco.status` table to confirm that the **ncp_disco** process is in rediscovery mode:
   `select * from disco.status;`

   Here is a sample response.

   ```
   m_DiscoveryMode=1;
   m_Phase=1;
   m_BlackoutState=0;
   m_CycleCount=0;
   m_ProcessingNeeded=0;
   m_FullDiscovery=0;
   ```

   From the results returned by the query, you can see that **ncp_disco** is currently in the rediscovery mode, that is, m_DiscoveryMode=1.
3. Start the SendTopologyToModel stitcher.

   The SendTopologyToModel sends the topology from ncp_disco to ncp_model.

   a) Ensure that you are in the OQL Service Provider or the **Management Database Access**.

   b) To insert the stitcher into the stitchers.actions table, issue the following command:

   ```
   insert into stitchers.actions
   ( m_Name )
   values
   ( 'SendTopologyToModel' );
   ```

   After your OQL insert is accepted, the stitcher is invoked and the network topology is sent to ncp_model. When the topology is sent, it is instantiated in accordance with the modified AOC hierarchy.
4. In order to ensure that the newly classified devices are removed from the Devices with Unclassified SNMP Object IDs report and the Devices with Unknown SNMP Object IDs report, perform the following steps:

   a) Clarify exactly which new sysObjectId values are being mapped by the new or edited AOC files. For example, the original AOC files mapped the following sysObjectId values:

   - 1.2.3.4
   - 1.5.6.*

   Then two new sysObjectId values are added to the system: 1.9.8 and 1.5.6.7. In the AOC file, the sysObjectId value 1.5.6.7 is covered by the mapping 1.5.6.*. However, the AOC file must be updated to add the sysObjectId value 1.9.8.

   b) Clarify which AOC files are mapped by the NCIM topology database mappings table.

   The mappings table is used by the Devices with Unclassified SNMP Object IDs report and the Devices with Unknown SNMP Object IDs report to determine which data to show in the reports.

This table is not automatically updated when you edit AOC files and restart the Topology manager, `ncp_class`, and consequently these reports continue to show the new sysObjectId values as unclassified and unknown. The mappings in the mappings table are also more specific than the mappings in the AOC files.

For example, the NCIM topology database mappings table might contain the following data:

Table 71. Example data

mappingGroup	mappingKey	mappingValue	Description
sysObjectId	1.2.3.4	Device Type A	Description of Device Type A
sysObjectId	1.5.6.1	Device Type B	Description of Device Type B
sysObjectId	1.5.6.2	Device Type C	Description of Device Type C

In the AOC file, only the sysObjectId value 1.9.8 needed to be added because the generic mapping 1.5.6.* covered the new sysObjectId value 1.5.6.7. However, in the NCIM topology database mappings table both sysObjectId values 1.9.8 and 1.5.6.7 must be added.

c) From the command line, update the NCIM topology database mappings table with relevant records for the new sysObjectId values.
For example, to add records for the two new sysObjectId values 1.9.8 and 1.5.6.7, issue the following SQL insert statements:

```
insert into mappings (mappingGroup, mappingKey, mappingValue) values
('sysObjectId', '1.9.8', 'device_type');
insert into mappings (mappingGroup, mappingKey, mappingValue) values
('sysObjectId', '1.5.6.7', 'device_type');
```

Where *device_type* is the device type to which the sysObjectId value must be mapped.

For information on the NCIM topology database mappings table, see the *IBM Tivoli Network Manager Reference*.

Results

After the AOC changes have been applied to the topology, either automatically by waiting for the next discovery, or manually by performing the steps in this topic, you will notice the following changes are applied to network polling and visualization.

- When you define a new polling policy, the new classes you defined are displayed in the Classes tab in the **Poll Policy Editor**.
- When you visualize the network using network views, the network view tree now displays the classes defined in the modified class hierarchy.
- If you updated the NCIM topology database mappings table as described then the Devices with Unclassified SNMP Object IDs report and the Devices with Unknown SNMP Object IDs report no longer return any devices.

AOC file samples

Use the AOC file samples to understand how Network Manager assigns discovered devices to the device classes in the class hierarchy.

EndNode class

Use this sample EndNode class AOC file to understand how Network Manager assigns discovered devices to the EndNode class.

Sample

The following sample AOC file fragment assigns devices to the EndNode class using the filter defined in the instantiate_rule clause.

```
//************************************************************
//
// File : EndNode.aoc
//
//************************************************************
active object 'EndNode'
{
super_class = 'Core';
instantiate_rule = "EntityOID like '1 \.3\.6\.1\.4\.1\.2021\.' OR
EntityOID = '1.3.6.1.4.1.2021' OR
EntityOID = '1.3.6.1.4.1.1575' OR
EntityOID like '1 \.3\.6\.1\.4\.1\.11\.2\.3\.9\.' OR
EntityOID = '1.3.6.1.4.1.11.2.3.9' OR
(EntityType = 1 AND EntityOID IS NULL)
OR
...
OR
(
EntityOID = '1.3.6.1.4.1.1977'
)
OR
(
EntityOID like '1\.3\.6\.1\.4\.1\.2136\.'
)
OR
...
```

For the EndNode class the instantiate_rule is very long. It consists of multiple lines comparing the EntityOID, (this is the sysObjectID of the device), to various values, joined together by an OR operator. There are different versions of the OR comparison:

EntityOID = '1.3.6.1.4.1.2021'
: This filter is looking for an exact match of the EntityOID to the value 1.3.6.1.4.1.2021. If the match is not exact, then the comparison fails and the device is not assigned to the EndNode class.

EntityOID like '1\.3\.6\.1\.4\.1\.11\.2\.3\.9\.'
: This filter is looking for a match like the value 1\.3\.6\.1\.4\.1\.11\.2\.3\.9\. The \. is required to make sure that the . (period) is matched. Also, notice that the value ends in \. This allows matching OIDs that start with the specified value but have additional values following the last . (period) specified.

NetworkDevice class

Use this sample NetworkDevice class AOC file to understand how Network Manager assigns discovered devices to the NetworkDevice class.

Sample

The following sample AOC file fragment assigns devices to the NetworkDevice class using the filter defined in the instantiate_rule clause.

```
//************************************************************
//
// File : NetworkDevice.aoc
//
```

```
//************************************************************
active object 'NetworkDevice'
{
super_class = 'Core';
instantiate_rule = 'EntityType = 1 OR // Chassis
EntityType = 2 OR // Interface
EntityType = 3 OR // LogicalInterface
EntityType = 5 OR // Card
EntityType = 6 OR // PSU
EntityType = 8 OR // Module
EntityType = 0';
...
```

For the NetworkDevice class, the instantiate_rule tries to match device types. The following examples are filters that are used in the instantiate_rule.

EntityType = 1
Matches all entities discovered that are chassis devices. Chassis devices have the field entityType set to a value of 1 in the NCIM topology database entityData table.

EntityType = 2
Matches all entities discovered that are ports or interfaces. Ports and interfaces have the field entityType set to a value of 2 in the NCIM topology database entityData table.

EntityType = 3
Matches all entities discovered that are logical interfaces. Logical interfaces have the field entityType set to a value of 3 in the NCIM topology database entityData table.

EntityType = 5
Matches all entities discovered that are cards. Cards have the field entityType set to a value of 5 in the NCIM topology database entityData table.

EntityType = 6
Matches all entities discovered that are power supply units (PSUs). PSUs have the field entityType set to a value of 6 in the NCIM topology database entityData table.

EntityType = 8
Matches all entities discovered that are modules. Modules have the field entityType set to a value of 8 in the NCIM topology database entityData table.

AOC specific to device class

Use this sample AOC file to understand how Network Manager assigns discovered devices to the device class at a lower level in the class hierarchy.

Sample

The following sample AOC file fragment assigns devices to the EWindowsNetHarmoni class using the filter defined in the instantiate_rule clause. This is an EndNode device.

```
//************************************************************
//
// File : EWindowsNetHarmoni.aoc
//
//************************************************************
active object 'EWindowsNetHarmoni'
{
super_class ='EndNode';

instantiate_rule = "EntityOID like '1 \.3\.6\.1\.4\.1\.1977\.1\.6\.1279\.'";
...
```

For the EWindowsNetHarmoni class, the following parameters are defined in the AOC file. The instantiate_rule parameter is long. It consists of multiple lines comparing the EntityOID, (this is the sysObjectID of the device), to various values, joined together by an OR operator. There are different versions of the OR comparison:

super_class ='EndNode'
This parameter establishes the device as belonging to the EndNode class. The EWindowsNetHarmoni class inherits all the attributes of the EndNode class.

instantiate_rule = "EntityOID like '1 \.3\.6\.1\.4\.1\.1977\.1\.6\.1279\.'"
This filter is looking for a match to the value 1\.3\.6\.1\.4\.1\.11\.2\.3\.9\. The \. is required to make sure that the . (period) is matched. Also, notice that the value ends in \. This allows matching OIDs that start with the specified value but have additional values following the last . (period) specified.

Entity types

The entityType table contains all the entity types that are available in the NCIM topology database.

The following table lists the entity types available in the topology database.

Table 72. Network Manager entities

Entity type	Entity type name	Category	NCIM table	Description
0	Unknown	Element		
1	Chassis	Element	physicalChassis	Main node device.
2	Interface	Element	networkInterface	Interfaces with entityType 2 can be discovered and polled.
3	Logical Interface	Element	networkInterface	Interfaces with entityType 3 are inferred but are not directly accessible. Hot Standby Routing Protocol (HSRP) virtual IP interfaces are an example of logical interfaces.
4	Local VLAN	Element	localVlan	VLAN port on a main node device.
5	Module	Element	physicalCard	Card within a switch or router. The term *module* is used to avoid confusion with the term *card* which is used in layer 1 networks.
6	PSU	Element	physicalPower Supply	Power® supply unit within a main node device.
7	Logical Collection	Collection		Examples of logical collections include MPLS VPNs, global VLANs and subnets. NCIM can also model OSPF areas.
8	Daughter Card	Element		The child of a network card.
9	Fan	Element	physicalFan	Fan component within a main node device.
10	Backplane	Element	physicalBackplane	Backplane component within a main node device. Backplanes usually contain slot entities.
11	Slot	Element	physicalSlot	Slot component within a main node device. Slots usually contain module entities.
12	Sensor	Element	physicalSensor	Sensor component within a main node device.
13	Virtual Router	Element	virtualRouter	Represents a instance of a virtual router within a chassis device.
14	CPU	Element	cpu	Represents Central Processing Units (CPUs).
15	Subnet	Collection	subnet	Logical collection that lists the IP address in a class A, B, or C subnet.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
16	Global VLAN	Collection	globalVlan	Collection of VLAN entities across multiple chassis devices that combine to form a virtual network.
17	VPN	Collection	networkVpn	Logical collection of IP address collected within a VPN.
18	HSRP Group	Collection	hsrpGroup	Represents an Hot Standby Routing Protocol (HSRP) group logical collection. The Cisco HRSP implements a virtual router with its own IP and MAC addresses. This virtual router forms an HSRP group that consists of a number of real interfaces, only one of which is active at any given time. The active interface forwards IP traffic that is sent to the virtual router, and the other interfaces in the group stand by ready to become active if the active interface fails.
19	Stack	Element		Collection of chassis devices as defined by the Entity MIB.
20	VRF	Element	vpnRouteForwarding	Represents a VPN routing and forwarding table.
21	OSPF Routing Domain	Collection	ospfRoutingDomain	Represents an OSPF routing domain.
22	OSPF Service	Service	ospfService	Represents an OSPF service running on a device.
23	OSPF Area	Collection	ospfArea	Represents an OSPF area.
24	VTP Domain	Collection	vtpDomain	Represents a VLAN trunking protocol domain.
25	Other	Element	physicalOther	Stores attributes of a component whose entity type the discovery was unable to determine. This occurs if the physical entity class is known, but does not match any of the supported values.
26	BGP Service	Service	bgpService	Represents a BGP service.
27	BGP AS	Collection	bgpAutonomousSystem	Represents a BGP autonomous system.
28	BGP Route	Attribute	bgpRouteAttribute	Represents a BGP route.
29	BGP Cluster	Collection	bgpCluster	Represents a BGP cluster.
30	BGP Network	Service	bgpNetwork	Represents a BGP network.
31	ISIS Service	Collection		Represents an ISIS service.
32	ISIS Level	Element		Represents the ISIS level.
33	OSPF Pseudo-Node	Element		Represents an OSPF pseudo-node.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
34	ITNM Service	Collection	itnmService	The base type for other services such as ISIS Service.
35	MPLS TE Service	Service	mplsTEService	Represents a Multi Protocol Label Switching Traffic Engineered (MPLS TE) service
36	MPLS TE Tunnel	Element	mplsTETunnel	Represents an MPLS TE tunnel
37	MPLS TE Resource	Element	mplsTETunnel Resource	Represents an MPLS TE resource
38	MPLS LSP	Element	mplsLSP	Represents an MPLS Label Switch Path (LSP)
40	IP Connection	Element	ipConnection	Represents a connection using TCP/IP.
41	PIM Service	Service	pimService	Represents a Protocol Independent Multicast (PIM) service.
42	PIM Network	Collection	pimNetwork	Represents a PIM network.
43	IPMRoute Service	Service	ipMRouteService	Represents an IP Multicast Routing service.
44	IPMRoute Upstream	Element	ipMRouteUpstream	Stores the upstream (RPF) route statistics for each device or Multicast Distribution Tree (MDT).
45	IPMRoute Downstream	Element		Stores the downstream route statistics per device or MDT.
46	IPMRouteMdt	Collection	ipMRouteMdt	Stores the Collection entities representing the MDTs for each Multicast Source or Group.
47	IPMRouteSource	Element	ipMRouteSource	Represents Multicast Sources, as contained by the MDT.
48	IPMRouteGroup	Element	ipMRouteGroup	Represents Multicast Groups, as contained by the MDT.
49	IP Path	Collection	ipPath	Represents a network path between IP devices.
50	IP End point	Protocol Endpoint	ipEndPoint	Represents a logical IP end point that is implemented by a physical interface.
51	VLAN Trunk End point	Protocol Endpoint	vlanTrunkEndPoint	Represents a logical VLAN trunk end point that is implemented by a physical interface.
52	Frame Relay End point	Protocol Endpoint	frameRelay EndPoint	Represents a logical Frame Relay end point that is implemented by a physical interface.
53	OSPF End point	Protocol Endpoint	ospfEndPoint	Represents a logical OSPF end point that is implemented by a physical interface.
54	ATM End point	Protocol Endpoint	atmEndPoint	Represents a logical ATM end point that is implemented by a physical interface.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
55	VPWS End point	Protocol Endpoint	vpwsEndPoint	Represents a logical VPWS end point that is implemented by a physical interface.
56	BGP End Point	Protocol Endpoint	bgpEndPoint	Represents a logical BGP end point that is implemented by a physical interface.
57	ISIS End Point	Protocol Endpoint		Represents a logical ISIS end point that is implemented by a physical interface.
58	MPLS Tunnel End Point	Protocol Endpoint	mplsTETunnelEndPoint	Represents a logical MPLS tunnel end point that is implemented by a physical interface.
59	TCP/UDP End Point	Protocol Endpoint		Represents a logical TCP/UDP end point that is implemented by a physical interface.
60	PIM End Point	Protocol Endpoint	pimEndpoint	Represents the Protocol Independent Multicast (PIM) end points discovered in the network and their associated attributes.
61	IPMRoute End Point	Protocol Endpoint	ipMRouteEndPoint	Stores information on the IP Multicast Routing Protocol End Points.
62	IGMP End Point	Protocol Endpoint	igmpEndPoint	Stores information on the Internet Group Membership Protocol (IGMP) End Points.
63	Network Service Entity End Point	Protocol Endpoint	networkServiceEntityEndPoint	Helps model relationships related to the management of frame relay links.
67	LAG End Point	Protocol Endpoint	lagEndPoint	Represents a logical Link Aggregation Group (LAG) end point that is implemented by a physical interface.
68	Probe End point	Protocol Endpoint	probeEndPoint	Fix Pack 3 Represents the source or target end point of a probe operation, implemented by a physical interface.
70	Topology	Topology		Grouping of connections which belong to a topology.
71	Layer 1 Topology	Topology		Grouping of connections which belong to a Layer 1 topology.
72	Layer 2 Topology	Topology		Grouping of connections which belong to a Layer 2 topology.
73	Layer 3 Meshed Topology	Topology		Grouping of connections which belong to a Layer 3 meshed topology.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
74	Converged Topology (Layer 1 - Layer 3)	Topology		Based on data available in NCIM, groups together connections at the lowest layer for which data is available.
75	MPLS TE Topology	Topology		Grouping of connections which belong to an MPLS TE topology.
77	Pseudo Wire Topology	Topology		Grouping of connections which belong to a Pseudo Wire topology.
78	OSPF Topology	Topology		Represents an OSPF topology.
79	BGP Topology	Topology		Represents a BGP topology.
80	IP Path Topology	Topology	ipPath	Represents an IP path.
81	PIM Topology	Topology		Represents PIM topologies.
82	Local VLAN Topology	Topology		Represents local VLAN topologies.
83	IPMRoute Topology	Topology		Represents an IP Multicast Routing topology.
84	VPLS Pseudo Wire Topology	Topology		Respresents a VPLS Pseudo Wire Topology.
85	Virtualization Topology	Topology		Represents a virtualization topology.
86	Microwave Topology	Topology		Represents a microwave topology.
87	RAN Topology	Topology		Represents a radio access network topology.
90	LTE Control Plane	Topology		Represents the devices and connectivity that make up the LTE control plane.
91	LTE User Plane	Topology		Represents the devices and connectivity that make up the LTE user plane.
92	Probe Topology	Topology		Represents the probe source/target connectivity.
110	Generic Collection	Collection	genericCollection	A collection that is not of any other type.
111	Geographic Location	Element	geographicLocation	Represents a geographic location.
112	Geographic Region	Collection	geographicRegion	Represents a geographic region.
113	VLAN Ports	Collection	vlanCollection	Represents a collection of the ports on a given named VLAN or, if no name is provided, on a given VLAN identifier.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
120	IGMP Service	Service	igmpService	Represents an Internet Group Management Protocol (IGMP) service.
121	IGMP Groups	Collection	igmpGroup	Stores multicast group collections for which there are associated Internet Group Membership Protocol (IGMP) end points in the igmpEndPoint table.
122	VSI (Virtual Switch Instance)	Element	virtualSwitch Instance	Represents a virtual switch instance (VSI) configured on a Provider Edge (PE) device that is associated with a Virtual Private LAN Service (VPLS) Virtual Private Network (VPN) instance.
123	Data Center	Element		Represents a data center.
124	Virtual Cluster	Collection	virtualCluster	Represents a cluster of virtual machines.
125	Virtual Management Service	Service	virtualMgmtService	Represents a virtual management service.
126	Hypervisor	Element	hypervisor	Represents a hypervisor.
127	Port Group	Collection	portGroup	Represents a port group.
128	EMS System	Element	emsSystem	Represents an EMS system accessed by a collector.
130	RAN GSM Cell	Element	ranGSMCell	Represents a GSM cell.
131	RAN UTRAN Cell	Element	ranUtranCell	Represents a UTRAN cell.
132	RAN Sector	Element	ranSector	Represents a RAN sector.
133	RAN NodeB Local Cell	Element	ranNodeBLocalCell	Represents a NodeB Local Cell.
134	RAN Location Area	Collection	ranLocationArea	Represents a RAN Location Area.
135	RAN Routing Area	Collection	ranRoutingArea	Represents a RAN Routing Area.
136	RAN Packet Core	Collection		Represents RAN packet switch core entity.
137	RAN Circuit Core	Collection		Represents a RAN circuit switched core entity.
138	RAN Radio Core	Collection	ranRadioCore	Represents a RAN radio core entity.
139	RAN Transceiver	Collection	ranTransceiver	Represents a RAN transceiver.
150	LTE Sector	Element	eUtranSector	Represents a geographic area of radio coverage and is implemented and supported by physical radio equipment. An LTE sector implements one or more LTE cells.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
151	LTE Cell	Element	eUtranCell	Represents a geographical area of radio coverage and is implemented and supported by physical radio equipment, such as towers, amplifiers, and antennas.
152	MME Function	Element	mmeFunction	The Mobility Management Entity (MME) is the main signalling control element in the core network and is the key control node for enabling user equipment access to the core network. The role of the MME is implemented within a network hardware node and is modelled by NCIM using the mmeFunction entity type. Multiple mmeFunction instances can be implemented within a single network hardware node.
153	Tracking Area	Collection	trackingArea	Represents an LTE tracking area.
154	SGW Function	Element	sgwFunction	The Serving Gateway (SGW) resides in the user plane where it forwards and routes packets to and from the eNodeB and packet data network gateway (PGW). The role of the SGW is implemented within a network hardware node and is modelled by NCIM using the sgwFunction entity type. Multiple sgwFunction instances can be implemented within a single network hardware node.
155	PGW Function	Element	pgwFunction	The Packet Data Network Gateway (PGW) provides user plane connectivity to packet data networks. The role of the PGW is implemented within a network hardware node and is modelled by NCIM using the pgwFunction entity type. Multiple pgwFunction instances can be implemented within a single network hardware node.
156	ENB Function	Element	enbFunction	The eNodeB device manages the radio air interface communication with users of the LTE network. Each eNodeB device controls one or more cells, which are geographic areas of radio coverage. The role of the eNodeB is implemented within a network hardware node and is modelled by NCIM using the enbFunction entity type. Multiple enbFunction instances may be implemented within a single network hardware node.
157	LTE Pool	Collection	ltePool	Generic modelling mechanism for groups of pooled LTE entities, and currently used to model MME pools, PGW pools, and SGW pools. As an example, in order to model an MME pool, the relationship between the ltePool entity and associated mmeFunction entities is modelled using the collects table.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
158	PLMN	Element	plmn	Models a Public Land Mobile Network (PLMN). A PLMN is a network that provides land mobile telecommunications services to the public. Each operator providing mobile services has its own PLMN.
159	HSS Function	Element	hssFunction	Models the LTE Home Subscriber Service (HSS). The HSS manages subscriber identities, service profiles, authentication, authorization, and quality of service (QoS), and acts as the master repository for subscriber profiles, device profiles and state information.
160	PCRF Function	Element	pcrfFunction	Models the LTE Policy and Charging Rules Function (PCRF). The PCRF manages the policy and charging for uplink and downlink service flows and the permitted EPS bearer QoS.
161	EIR Function	Element	eirFunction	Models the LTE Equipment Identity Register (EIR). The EIR keeps track of mobile devices which should either be banned from using the network or monitored. When a mobile phone is stolen its identity it is added to the EIR blacklist and the result is that this phone will never be able to attach to the network for service. Usually each network has its own EIR which is often combined with the HSS node. It is possible for multiple operators to share a common EIR which enables the blacklisted information to be more easily and widely available.
163	LTE Control Plane	Collection	controlPlane ViewCollection	Supports the dynamic collection views under **LTE Network Topology** > **Control Plane by Tracking Area** in the Network Views. Each instance of this entity type collects the eNodeBs in the corresponding tracking area, together with the devices that these eNodeBs are connected to on the control plane.
164	LTE User Plane	Collection	userPlane ViewCollection	Supports the dynamic collection views under **LTE Network Topology** > **User Plane by Tracking Area** in the Network Views. Each instance of this entity type collects the eNodeBs in the corresponding tracking area, together with the devices that these eNodeBs are connected to on the user plane.
170	Aggregated Link	Collection	aggregatedLink	Represents a network link between Link Aggregation Groups (LAGs)
171	Link Aggregation Group	Element	aggregationGroup	Represents a Link Aggregation Group (LAG).
190	Probe Service	Service	probeService	Fix Pack 3 Represents the service that provides probes on a device.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
191	Probe	Collection	probe	Fix Pack 3 Represents configured network probes and their attributes.
192	Probe Collection	Collection	probeCollection	Fix Pack 3 Provides a collection facility for probes or probe collections.
200	LTE S1-U	Topology	entityData	Topology type for LTE S1-U connectivity.
201	LTE S5	Topology	entityData	Topology type for LTE S5 connectivity.
202	LTE S8	Topology	entityData	Topology type for LTE S8 connectivity.
203	LTE S1-MME	Topology	entityData	Topology type for LTE S1-MME connectivity.
204	LTE S10	Topology	entityData	Topology type for LTE S10 connectivity.
205	LTE S11	Topology	entityData	Topology type for LTE S11 connectivity.
206	LTE SGi	Topology	entityData	Topology type for LTE SGi connectivity.
207	LTE Gx	Topology	entityData	Topology type for LTE Gx connectivity.
208	LTE S3	Topology	entityData	Topology type for LTE S3 connectivity.
209	LTE S4	Topology	entityData	Topology type for LTE S4 connectivity.
210	LTE S6a	Topology	entityData	Topology type for LTE S6a connectivity.
211	LTE S13	Topology	entityData	Topology type for LTE S13 connectivity.
212	LTE X2	Topology	entityData	Topology type for LTE X2 connectivity.
250	NR Sector	Element	eUtranSector	5G New Radio Sector Entity.
251	NR Cell DU	Element	NRCellDU	Represents a geographical area of radio coverage and is implemented and supported by physical radio equipment for 5G LTE, such as towers, amplifiers, and antennas.
252	NR Cell CU	Element	NRCellCU	Represents a geographical area of radio coverage and is implemented and supported by physical radio equipment for 5G LTE, such as towers, amplifiers, and antennas.

Table 72. Network Manager entities (continued)

Entity type	Entity type name	Category	NCIM table	Description
253	GNB DU Function	Element	gnbFunction	The gNodeB device manages the radio air interface communication with users of the 5G network. Each gNodeB device controls one or more cells, which are geographic areas of radio coverage. The role of the gNodeB is implemented within a network hardware node and is modelled by NCIM using the gnbFunction entity type. Multiple gnbFunction instances may be implemented within a single network hardware node.
254	GNB CUCP Function	Element	gnbFunction	The gNodeB device manages the radio air interface communication with users of the 5G network. Each gNodeB device controls one or more cells, which are geographic areas of radio coverage. The role of the gNodeB is implemented within a network hardware node and is modelled by NCIM using the gnbFunction entity type. Multiple gnbFunction instances may be implemented within a single network hardware node.
255	GNB CUUP Function	Element	gnbFunction	The gNodeB device manages the radio air interface communication with users of the 5G network. Each gNodeB device controls one or more cells, which are geographic areas of radio coverage. The role of the gNodeB is implemented within a network hardware node and is modelled by NCIM using the gnbFunction entity type. Multiple gnbFunction instances may be implemented within a single network hardware node.

Chapter 16. Keeping discovered topology up-to-date

After a discovery has completed, you can keep the discovered topology updated by scheduling a discovery, manually discovering devices, and removing devices.

Scheduling discoveries

After a full discovery completes, you can schedule further discoveries by inserting the time, date, and day for discoveries to run in the `FullDiscovery.stch` stitcher file.

Procedure

1. Back up the `NCHOME/precision/disco/stitchers/FullDiscovery.stch` file.
2. Create separate instances of the `FullDiscovery.stch` file for each domain in the network.

   To create a domain-specific instance, insert `.domain` into the file name.

   For example, `FullDiscovery.NCOMS.stch`.

   If you do not have separate `FullDiscovery.stch` files for each domain, all the domains on the network are discovered.
3. Schedule the discovery of the first domain.

   In a `FullDiscovery.domain.stch` file, uncomment one of the ActOnTimedTrigger lines. Then, change it to run the discovery at a certain time.

   For example, to schedule the discovery for 11:00 PM every day, change the line as follows:

   ```
   ActOnTimedTrigger(( m_TimeOfDay ) values ( 2300 ) ; );
   ```
4. Repeat the steps in the `FullDiscovery.stch` file for each domain on the network.

Examples

- To schedule a discovery on the sixth day of the week since Sunday (that is, on Saturdays) at 11 PM:

  ```
  ActOnTimedTrigger(( m_DayOfWeek , m_TimeOfDay )
   values ( 6 , 2300 ) ; ) ;
  ```

  Sunday = 0, Monday = 1, Tuesday = 2, Wednesday = 3, Thursday = 4, Friday = 5, Saturday = 6.
- To schedule a discovery on the 13th day of each month at 2 PM:

  ```
  ActOnTimedTrigger(( m_DayOfMonth , m_TimeOfDay )
  values ( 13 , 1400 ) ; );
  ```
- To schedule a discovery at intervals of 13 hours:

  ```
  ActOnTimedTrigger(( m_Interval ) values ( 13 ) ; );
  ```

Viewing discovery status for a device

From the topology GUIs you can view information about a device, including when it was first discovered, last discovered, and last rebooted.

Before you begin

To perform this procedure, you must be in the **Network Views** or in the **Network Hop View**, and the topology map must be displaying the device of interest.

© Copyright IBM Corp. 2006, 2021

Procedure

1. In the topology map, right select one or more devices.
2. Select **Discovery...** > **Show Discovery Overview**.
3. The **Discovery Overview** window displays the following information:

Access IP

IP address through which this entity was discovered and through which it is monitored.

Class Name

Class of devices to which this device belongs.

Current

Current date and time.

Entity Name

IP address, DNS name, or sysName for this device. For example, an IP address such as 172.20.1.7, or a DNS name, such as company-abc.net.

Entity Created

Date and time when the entity was first uploaded to the NCIM topology database.

Interface Filtered

A flag to show whether the device has had interface filtering applied to it or not. If you do not see all the information for the device that you expect, the information might have been filtered out. You can do an SNMP walk of the device using the MIB Browser with the option to ignore filtering.

Last Discovered

Date and time when the Details agent last accessed the device.

Last Modified

Date and time when the last detected change on the device was reflected in the NCIM topology database. For example, if an interface name on the device changes, this is the time at which that change was uploaded to NCIM.

Last Reboot

Date and time when the device was last rebooted. This is calculated based on the sysUpTime MIB value retrieved from the device, so **Last Reboot** is only available if the sysUpTime was retrieved. For example, for devices with no SNMP access, the value of **Last Reboot** is NULL.

Manually discovering a device or subnet

You can manually discover devices so that the network topology in Network Manager matches the network.

About this task

Sometimes you might know that one or more devices have had their configuration changed, and want to discover them again regardless of whether the system has detected the change from traps sent by the devices.

You can manually discover a device or subnet in the following ways:

- You can use the Discovery Configuration GUI to specify individual devices or complete subnets to be discovered.
- You can discover specific devices or sets of devices from the Hop View or Network Views.
- You can make inserts to the finders.rediscovery table using ncp_oql, specifying the IP address or subnet to be discovered.

Note: Do not use manual discoveries to remove devices from the topology. Devices that are no longer accessible remain in the topology until their LingerTime reaches zero and another discovery is run, or until you remove them using the RemoveNode.pl script. Do manual discoveries only against devices that are operational but whose configuration has been changed.

Manually discovering a device or subnet using the GUI

You can configure and launch discovery of a device or subnet from the Discovery Configuration GUI. You can customize the discovery configuration to make partial discovery run as quickly as possible.

Enabling partial discovery agents

You can configure partial discovery by enabling the appropriate agents from the **Partial Discovery Agents** tab in the Discovery Configuration GUI.

About this task

You can speed up the time taken for a partial discovery by selecting only those agents essential to discover new or modified devices.

Configuring advanced partial discovery settings

Among the advanced partial discovery settings that you can configure are feedback, rebuilding of layer, and remote neighbor parameters.

Configuring feedback settings

You can specify feedback settings when configuring a partial discovery with the GUI.

About this task

Feedback is the mechanism by which data returned by agents is used to find other devices. Examples of feedback data include the IP address of remote neighbors, or the subnet within which a local neighbor exists.

The feedback mechanism allows any new IP addresses to be fed back into the discovery and thus increases the size of the discovered network. You need to balance completeness of the discovered topology (feedback *on*) with greater speed of discovery (feedback *off*).

You can choose from the following options after selecting the **Advanced** tab within the Configuration option in the Discovery Configuration GUI:

- **No Feedback**: Feedback is off for all discoveries. This option provides speed but discovers only those devices specified to the finders, and hence provides an incomplete topology. However, this setting ensures that discoveries complete in the quickest possible time.
- **Feedback**: Feedback is on for full discoveries and partial discoveries. This option provides a complete topology in all situations but takes the longest time.
- **Feedback Only on Full**: Feedback is on for full discoveries, ensuring a complete topology. For partial discoveries there is no feedback. This ensures that the partial discovery runs in the quickest time possible. This is the default setting.

Configuring rebuilding of layer settings

You can allow the rebuilding of the topology layers to display an accurate topology when you configure a partial discovery.

About this task

To rebuild the topology layers following a partial discovery, select the **Enable Rediscovery Rebuild Layers** setting on the **Advanced** tab within the Configuration option in the Discovery Configuration GUI. If you specify that topology layers *should* be rebuilt following partial discovery, the result is an accurate topology showing all connectivity. However, the process of adding new devices takes longer.

Enabling discovery of remote neighbors for partial discovery

You can improve the accuracy of connections found during a partial discovery by enabling the discovery of remote neighbors.

About this task

By default, remote neighbor discovery is off. Enabling remote neighbor discovery makes the discovery take longer.

With remote neighbour discovery on, Network Manager checks, during a partial discovery, whether any connections to remote neighbors have changed. (Remote neighbors in this context are connected devices that were in scope of the last full discovery but are out of scope of the current partial discovery.)

If the connections have changed, the connected devices are included in the partial discovery, resulting in a more accurate topology.

Restriction: If a connection between devices has changed, but the information about the connection is stored only on the device that is out of scope, then the change is not registered and the connected devices are not included in the partial discovery. Enabling remote neighbor discovery increases the accuracy of the topology, if it has changed, but does not ensure that all changes are discovered. For the most accurate topology, run a full discovery.

To enable remote neighbor discovery, select **Enable Rediscovery of Related Devices** on the **Advanced** tab within the **Configuration** option in the Discovery Configuration GUI.

Starting partial discovery from the GUI

Starting a partial discovery involves defining a seed and scopes.

About this task

If a full discovery has not been run since the last time that the discovery engine, **`ncp_disco`**, was started, you cannot start a partial discovery.

You can start a partial discovery of a device or subnet from the **Active Discovery Status** window. You can also discover specific devices by right-clicking them from within the Hop View and Network Views.

To start a partial discovery from the **Active Discovery Status** window, complete the following tasks.

Procedure

1. Select the domain in which you want to run a discovery from the **Domain** menu. You can start to type the name of the domain, and matching domains are listed below the **Domain** field.
2. Click the downward-facing arrow next to the **Start Discovery** button  and select **Start Partial Discovery** from the menu. The **Partial Discovery** window is displayed. Specify the IP addresses and subnets that contain the devices to be discovered
3. Under **Partial Discovery**, select the required nodes and subnets.
4. To add a new subnet or node, click **New.**
5. Complete the fields as follows and click **OK**:

   Discover
   Select one of the following options:

   Identifier
   Type the required IP address.

   Subnet
   Type the required subnet and specify the number of netmask bits. The **Netmask** field is automatically updated.

EMS Device Name
Type the name of a device that was discovered through an Element Management System (EMS). You can type the nativeID, entity name, sysName, or display name.

Note: When running a partial discovery of an EMS collector from the **Discovery Status GUI**, within the **New Partial Discovery Node/Subnet** window you must specify the EMS identifier value in the **EMS Device Name** field and not in the **Identifier** field. The **Identifier** field only accepts IP addresses.

6. To add new scope zones, click **Scope**.

   Note: If you add a scope zone that is not included within the scope of the last full discovery, connections between devices in the new scope and the old scope might not be accurate until the next full discovery. Enabling remote neighbor discovery can help improve the accuracy of these connections.

7. To add a new discovery scope zone click **New**  . To edit an existing scope zone, click the required entry in the list.
8. Complete the fields as follows and click **OK**:

   Action
   Define the subnet range as an inclusion zone or exclusion zone. If the subnet range is an inclusion zone that you intend to ping during the discovery, click **Add to Ping Seed List**. Clicking this option automatically adds the devices in the scope zone as a discovery seed devices.

   Restriction: The `Add to Ping Seed List` option is not available for IPv6 scope zones. This prevents ping sweeping of IPv6 subnets, which can potentially contain billions of devices to be pinged. Ping sweeping of IPv6 subnets can therefore result in a non-terminating discovery.

9. Click **OK** then click **Go**.

   When a partial discovery is running, the **Start Discovery** button is toggled off  .

Manually discovering a device or subnet from the command line

You can manually discover a device or subnet from the command line.

About this task

To manually discover a device or subnet from the command line, make inserts to the `finders.rediscovery` table using `ncp_oql`, specifying the IP address or subnet to be discovered, as described in the following example.

Manual discovery

To manually discover the device with IP address 192.168.1.2, first start the OQL Service Provider using the following command:

```
ncp_oql -domain NCOMS -service Disco
```

Having logged into the OQL provider, run the following query (note that the command is entered on one line):

```
insert into finders.rediscovery (m_Address, m_RequestType) values
("192.168.1.2", 1);
```

When discovery of a device is forced like this, `ncp_disco` immediately passes it to the Ping finder to confirm it exists, and, if it does, triggers the appropriate agents to reanalyze it. If the connections from the device have changed, neighboring devices might also be discovered.

Removing a device from the network

You can manually remove a device that is known to be scheduled for permanent removal from the network.

Procedure

1. Remove the device from the topology database using the **RemoveNode.pl** script.
2. Physically remove the device from the network.

Setting the linger time for a device

The value of the LingerTime field indicates how many discoveries a device can fail to be found in before it is assumed to have been removed from the network and its record is removed from the topology. Setting the LingerTime field to zero ensures that when the device is not found in the next discovery, its record is immediately removed from the topology.

About this task

Setting the LingerTime in the ncimCache.lingerTime table sets the LingerTime for only the chassis. Contained entities such as interfaces and cards are removed from the topology when a discovered device has different containment data showing that the contained entity is no longer present. LingerTime can be overridden for specific devices or contained entities by editing the discovery stitchers.

To set the LingerTime field for a device to zero, complete the following steps:

Procedure

1. Enter a command similar to the following to start the OQL Service Provider:

```
ncp_oql -domain NCOMS -service Model
```

2. Update the LingerTime field in the ncimCache.lingerTime table for all entities that represent the device. For example, if the device is called `core-router.abcd.com`, enter the following command, on one line:

```
update ncimCache.lingerTime set lingerTime->LINGERTIME = 0
where ENTITYNAME = 'core-router.abcd.com';
```

Chapter 17. Troubleshooting discovery

You can troubleshoot discovery by monitoring discovery events and by running discovery reports. You can also configure your own discovery events.

Related tasks

Troubleshooting Network Manager

Consult these troubleshooting notes to help determine the cause of the problem and what to do about it.

Troubleshooting discovery with reports

The troubleshooting reports provide easy visibility into the discovery results to help with verification and troubleshooting of both the discovery results and the network itself.

About this task

Network Manager uses the Cognos Analytics component to generate reports.

For more information, see the Cognos Analytics Knowledge Center at: https://www.ibm.com/support/knowledgecenter/SSEP7J.

To access the reports in the Network Manager GUI, complete the following steps:

Click the **Reporting** icon and select **Common Reporting**. Within the widget, select **Network Manager**. A list of folders display. These folders contain all Cognos reports for your access.

You can use reports for verifying and troubleshooting discovery results such as the examples in Table 73 on page 407.

For more information on the Network Manager reports, refer to the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Table 73. Report categories to use for discovery troubleshooting

Troubleshooting task	Consult this report category and report	Benefit of the report
Identifying all discovered nodes and interfaces	Utility reports: Discovered nodes and interfaces flat file list	This report lists all nodes and interfaces that were discovered. It also marks interfaces or ports connected to network devices. It allows you to check that specific devices and interfaces were actually discovered.
Resolving mismatches	Troubleshooting reports: Connected Interface Duplex Mismatch	This report provides a list of connections that have mismatches between half- and full-duplex devices, where one end of the connection is half-duplex and the other end is full-duplex. This mismatch is one of the key configuration problems that network managers have to find to resolve performance or availability issues.
Resolving inaccessible devices	Troubleshooting reports: Devices with no SNMP Access	This report identifies the devices that do not have SNMP access. You can then identify the reason for the SNMP access failure.
Resolving unconnected devices	Troubleshooting reports: Devices With No Connections	This report lists unconnected devices as a first step in determining why the discovery found no network connections for a device.

© Copyright IBM Corp. 2006, 2021

Table 73. Report categories to use for discovery troubleshooting (continued)		
Troubleshooting task	**Consult this report category and report**	**Benefit of the report**
Resolving unclassified devices	Troubleshooting reports: Devices with Unclassified SNMP Object IDs	Using these reports, you can create leaf-node AOC files for the new device class.
	Asset reports: • Interface Availability • Summary By Device Class • Vendor and Device Availability	
Resolving devices with unregistered SNMP object IDs	Troubleshooting reports: Devices with Unknown SNMP Object IDs	Using the information in this report, you can modify the AOC files associated with the unregistered devices.
Identifying devices pending deletion	Troubleshooting reports: Devices Pending Delete on Next Discovery	This report displays information on devices to be deleted from the topology if they are not found during the next discovery cycle. The report allows you to check that removal of devices from the topology is progressing, and to identify devices erroneously scheduled for removal.

Monitoring discovery status

You can view discovery status messages to understand the status and progress of the discovery. You can also configure your own discovery events.

Process flow for creating discovery events

Discovery events are created during the discovery process showing the progress of agents, stitchers, and finders. These events are sent to and stored in Tivoli Netcool/OMNIbus and can be viewed using the Web GUI.

Discovery events are created in the following stages:

- During the data collection phase of discovery, dedicated stitchers (AgentStatus and FinderStatus) detect whether finders and agents have started or stopped.
- During the data processing phase, a dedicated stitcher (CreateStchTimeEvent) detects key events; for example, discovery has started building the working entities table, or the containment table.
- Whenever one of the above stitchers detects an event, it writes that event to the `disco.events` table.
- The DiscoEventProcessing stitcher responds to an insert into the `disco.events` table and creates and sends the appropriate event to the probe for Tivoli Netcool/OMNIbus, nco_p_ncpmonitor, which then forwards the event to the ObjectServer.
- You can switch the generation of discovery events on or off by setting the value of the `m_CreateStchrEvents` field in the `disco.config` table.

You can configure your own discovery events by writing a stitcher to detect the desired event and write that event data to the `disco.events` table.

Monitoring discovery status messages

You can view discovery status messages to understand the status and progress of the discovery.

About this task

The discovery processes, including agents, stitchers, and finders, send messages to IBM Tivoli Netcool/OMNIbus when they start and stop. You can view these messages to see if the discovery processes are running as expected, and to gauge the overall progress of discovery.

To view discovery process status messages, complete the following tasks.

Procedure

1. Click the **Incident** icon and select **Events** > **Event Viewer**.
2. Apply a filter to the **Event Viewer** so that only events with an `Agent` of `ncp_disco` are displayed.
3. Optional: Refine the filter or sort on **EventId** to view only specific kinds of discovery events.
4. Ensure that the **LocalPriObj** and **LocalSecObj** columns are displayed in the **Event Viewer**. These columns contain information for discovery events. (Not all columns are used by all events.)

NCIM entity ID retrieval failure

If a discovery event appears in the **Event Viewer** with the description "The retrieval of an entity ID from NCIM for a certain entity failed", then the NCIM database might be down. This will cause a mismatch between the entity IDs in NCIM and in the embedded discovery database DNCIM; that is, NCIM and DNCIM will have a different set of entity IDs for the same network entities.

If the NCIM database is down for any reason while the RetrieveNCIMEntityId stitcher is running, then there will be a mismatch between the entity IDs in NCIM and DNCIM. It is not essential for the entity IDs in DNCIM and NCIM to match; however, in the event of discovery fault resolution, it is helpful if the entity IDs are consistent across the databases.

If there is a mismatch between the entity IDs inNCIM and DNCIM then you can make sure that the databases are aligned for the next discovery by doing one of the following:

- Refresh the DNCIM database prior to the next discovery
- Log into your running discovery DNCIM database and delete all the records in entityNameCache for your domain

Note: If your NCIM topology database is shared between multiple domains then the chance of the DNCIM and NCIM entity IDs becoming mismatched is much higher.

Refresh the DNCIM database prior to the next discovery

Refresh the DNCIM database only if the Discovery engine, ncp_disco, is not running.

Refresh the DNCIM database by deleting the following directory and everything in it. The DNCIM database stores all the database files in this directory:

```
$NCHOME/precision/embeddedDb/sqlite/processName.domain
```

Where:

- *processName* is the name of the Network Manager process that is using DNCIM. In this case, this is the name of the Discovery engine, ncp_disco.
- *domain* is the name of the current domain.

For example, if the name of the current domain is NCOMS, then the database files are located here:

```
$NCHOME/precision/embeddedDb/sqlite/ncp_disco.NCOMS
```

Note: This will result in a small delay to the next discovery because the DNCIM database must be rebuilt.

Log into the DNCIM database and delete all records in entityNameCache for your domain

Provided that the discovery is in the static state (phase 0) it is safe to alter the discovery DNCIM database. Proceed as follows:

1. Check that the Discovery engine, ncp_disco, is running and that discovery is in phase 0 by running the following query. If this query returns the value 0, then the discovery is in phase 0.

   ```
   select m_Phase from disco.status
   ```

2. Issue the following OQL command to access the DNCIM database:

   ```
   ncp_oql -domain domain_name -service DNCIM
   ```

   Where *domain_name* is the name of the current network domain.

3. Issue the following command to delete all the records in entityNameCache:

   ```
   DELETE FROM entityNameCache
   ```

   Note: For a large database, the delete operation can take a long time.

Troubleshooting discovery agents

You can use the Discovery Status GUI to troubleshoot discovery problems associated with discovery agents.

Troubleshooting an unusually long discovery

A discovery might be taking a long time to complete because an agent is unable to complete processing on a specific device. Use the **Agents Status** section to determine which agent is taking a long time to complete and which device it is working on.

About this task

To use the **Agents Status** section to determine if the cause of the problem is an agent that is blocked on a device, complete the following steps:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Status**.
2. Within **Network Discovery Status**, click the **Agents Status** tab.
3. Set the Phases drop-down list above the upper Agents table to Interrogating Devices.

   The upper Agents table now displays only agents that are scheduled to complete in the first discovery phase, Interrogating Devices.

   Note: This problem usually occurs during the first discovery phase, Interrogating Devices.

4. Ensure that the **State** column is sorted in descending order.

   The agents appear by default in descending order of agent state, as listed in the following table.

Table 74. Agent states

State	Value	Icon	Description
Died	5		The agent has terminated unexpectedly. This is a potential discovery problem.
Finished	4		The agent is still running but has finished processing of all the entities in its queue. The agent is still available to process any further agents placed in the queue.

Table 74. Agent states (continued)			
State	**Value**	**Icon**	**Description**
Running	3		The agent is currently processing entities.
Starting	2		The agent is starting up.
Not running	1		The Agent is not running.

5. Scroll down the table to find the agents that have the status Running .
   These are the agents that are still processing devices. If the discovery has been running for an unusually long time, then there might be just one agent that still has Running status . This is the blocked agent.
6. Select one of the agents with Running status .
   By default, the lower table now displays all the entities that are still queued for this agent.
7. Click the **All** radio button above the lower table.
   The lower table now shows all entities that have been processed by this agent, that are still being processed by this agents, or that are in the agent queue.
8. Ensure that the **State** column is sorted in descending order.
   The entities appear by default in descending order of agent state, as listed in the following table.

Table 75. Entity states			
State	**Value**	**Icon**	**Description**
Died	5		Processing of the entity terminated unexpectedly. Either the agent was stopped manually, or there is a discovery problem.
Finished	4		An agent has completed processing this entity.
Running	3		An agent is currently processing this entity.
Starting	2		An agent is beginning to process this entity.
Not running	1		This entity is not currently being processed.

9. Scroll down the table to find the entities that have the status Running .
   These entities are still being processed by this agent. If the agent is stuck on a single device, then there will only be one entity with Running status .
10. Look at the other information in the table to find out more about this entity.
    The Elapsed Time column indicates how long the agent has been processing this device. The SNMP Access column indicates whether the agent was able to gain SNMP access to this device. If the agent was unable to gain SNMP access to the device, there might be a problem with SNMP community string settings. Further investigation of this device is required.
11. If the device has a large number of interfaces, the discovery might appear to be stuck on this device because it takes a long time to download all the information. Configure an interface filter for this device to filter out information that you are not interested in. When you run discovery again, the SNMP helper retrieves less information from the device, and this might solve the problem.

Identifying failed agents

A source of discovery failure can be agents that terminate unexpectedly during discovery. Use the **Agents Status** section to determine whether any agents terminated unexpectedly.

About this task

To use the **Agents Status** section to determine whether any discovery agents are not running correctly, complete the following steps:

Procedure

1. Click the **Discovery** icon and select **Network Discovery Status**.
2. Within **Network Discovery Status**, click the **Agents Status** tab.
3. Ensure that the Phases drop-down list above the upper Agents table is set to All Phases.

   The upper Agents table now displays all agents that started so far in this discovery.
4. Click the **State** column in the upper Agents table so that the agents are ordered in descending order of **State**.

   The agents now appear in the table in alphabetical order of status.
5. Any agents that terminated unexpectedly are at the top of the table and have the status Died.

What to do next

Further investigation is required to determine why this agent terminated unexpectedly.

Troubleshooting missing devices

If a device that you expect to find in your network topology is not present, follow these steps to troubleshoot the problem.

Before you begin

Before following these steps, run a full discovery with feedback enabled.

About this task

To check some common causes for a device not being found in the network maps, complete the following steps.

Procedure

1. Verify that the device you are looking for is running and connected to the network.
2. Search for the device.
   a) Search for the device in the network maps by hostname and then by IP address.
   b) If you know which devices it is connected to, try finding one of the connected devices in the **Network Hop View**. Then set the number of hops to 1 and see if the device is shown as connected.
3. Check whether the device is in scope. Review the discovery scope, including exclusion zones, in the **Scope** tab of the **Network Discovery Configuration** GUI.
4. Check if the device is being filtered out of the discovery.
   a) Click **Filters**.
   b) Review the prediscovery and postdiscovery filters to ensure that the device is not being prevented from being discovered or instantiated.
   c) Search the `ncp_disco.`*`domain`*`.log` file for the device name.

If the device has been filtered out by a postdiscovery filter, messages are present such as:

```
Filtering entity name and not sending to model
```

```
name matches the instantiateFilter
```

If a device relationship has been filtered out by a postdiscovery filter, messages are present such as:

```
Filter relationship involving name
```

Troubleshooting an idle discovery

If you start the discovery, and after some minutes no devices have been discovered, follow these troubleshooting steps.

About this task

If the discovery status stays in phase zero (idle) after you start it, and no devices have been discovered, try the following troubleshooting steps.

Procedure

1. If you are using the file finder, check that you have correctly specified which field in the seed file contains the IP address and which contains the host name. You can verify these settings in the Discovery Configuration GUI.
2. If you are using the ping finder and pinging individual IP addresses, check that these IP addresses are reachable. If not, you might have a network outage or a firewall issue.
3. Verify that the seed IP addresses are in scope. Even if you add an address to the ping finder or file finder, the device is not pinged or instantiated if it is not included in the scope. For example, if your discovery scope is 172.16.1.0 /24 and your seeds are in the 192.168.1.0 /24 network, then the finders cannot find them.
4. If you are pinging a large, sparsely populated subnet, for example, a class B subnet containing only 10 devices, the ping finder might take a long time to find the first device.

What to do next

If you need to review the discovery logs, refer to the information about locating log files and changing logging levels in the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Repairing a corrupted discovery

If the discovery stops in an unusual way, for example, if the ncp_disco process is stopped forcibly from the command line, or exits unexpectedly, you might need to repair the discovery before running another discovery.

About this task

If the discovery is shut down normally, all the dependent processes such as discovery agents are also shut down, and the discovery database is ready for another discovery. However, if the ncp_disco process is stopped forcibly or unexpectedly, then dependent processes can be left running, and the discovery cache files are left in a state such that they can interfere with the next discovery. In this case the discovery can be said to be *corrupted*.

To repair a corrupted discovery, complete the following steps:

Procedure

1. Remove the entry for the ncp_disco process from the services.inTray database in ncp_ctrl, if it is present. Removing the entry prevents the ncp_ctrl process from restarting ncp_disco.
2. Stop the ncp_disco process, if it is still running, using an appropriate command for your operating system.
   For example, on Unix, run the `kill -9` command on the process ID for ncp_disco.
3. Remove the cache files for the dNCIM database. Navigate to the `$PRECISION_HOME/embeddedDb/sqlite` directory and delete the `ncp_disco.`*`domain`* directory.
4. Optional: Archive or remove existing log files to start the next discovery with fresh log files. The log files for the following processes are relevant:
   - **ncp_disco**
   - **ncp_df_***
   - **ncp_agent***
   - **ncp_disco_perl_agent***
5. Search for any discovery finders or agents that are still running in the corrupted domain. The discover finders have process names that start with ncp_df. The discovery agents start with ncp_agent, and any Perl discovery agents start with ncp_disco_perl_agent. Stop any agents or finders that are still running in the corrupted domain.
6. If you want to run the ncp_disco process as a process managed by the ncp_ctrl process, which is the default, then start ncp_disco as a managed process. See the topic on starting managed processes for more information.
   Starting ncp_disco as a managed process starts a new discovery.
7. Start or monitor your discovery using the Discovery Configuration GUI.

Removing discovery cache files

Remove discovery cache files to perform a new, clean discovery.

About this task

To remove the current network discovery for a domain, you must remove all discovery cache files. You might want to do this when you need to remove all data from a previous discovery, or when directed to do so by IBM Support.

This procedure deletes all current discovery cache files and clears the discovery database, effectively resetting discovery. After performing this procedure, you must run a new full discovery of your network.

Note: Because the network topology is stored separately in the NCIM database, this procedure does not remove your network maps. However, any changes made to your network since the last discovery are reflected in the next discovery.

Perform the following procedure to remove all discovery cache files:

Procedure

1. Stop all Network Manager processes using the `itnm_stop` script.
2. Navigate to the `$NCHOME/var/precision` directory and remove all files that belong to the domain you wish to remove. Files that belong to a particular domain have the domain in the file name. For example, a configuration file belonging to the domain NCOMS would be called `file_name.NCOMS.cfg`.
3. Navigate to the `$PRECISION_HOME/embeddedDb/sqlite` directory and delete the `ncp_disco.`*`domain`* directory.
4. Optional: Archive or remove existing log files to start the next discovery with fresh log files. The log files for the following processes are relevant:

- **ncp_disco**
- **ncp_df_***
- **ncp_agent***
- **ncp_disco_perl_agent***

5. Restart the Network Manager processes using the `itnm_start` script.

   New, empty log files are automatically created when the Network Manager processes are restarted using the `itnm_start` scripts.
6. Perform a new network discovery.

Troubleshooting illegal characters

If you see an error message about illegal characters in insert statements into the topology database, follow these steps to troubleshoot the problem.

About this task

If you have network devices with characters in their descriptions that are not allowed in the locale set in the database, you might see an error message similar to the following:

```
Warning: W-RIV-002-206: [4115626896t] CMdlDbEntityMgr.cc(647) A database 'execute'
operation has failed : SQLRETURN = -1 CNcpODBCSth.cc line 233 : [<database>][<database> ODBC
Driver]
[<database>]An illegal character has been found in the statement.
```

Procedure

1. Back up and edit the `SnmpStackSchema.cfg` file.
2. Locate the line that configures an insert into the `snmpStack.conversionCfg` table and edit it to the following:

   ```
   insert into snmpStack.conversionCfg values (1);
   ```
3. Save and close the file.

Results

The SNMP Helper substitutes characters returned from devices that are not allowed in the database locale with the question mark character: '?'.

The SNMP Helper substitutes characters on only those objects that are configured in the snmpStack.multibyteObjects table.

Chapter 18. Discovering and using custom data

You can enrich the topology by adding custom data to the information discovered by the discovery process. For example, you can add custom data about discovered entities from third-party sources using agents or collectors. This custom data can subsequently be used for event enrichment, network visualization, polling and reporting.

About this task

Adding custom data to the network topology involves discovering the data and then storing it in the topology database. There are several methods for discovering data. Choose the appropriate method based on the source and complexity of the data.

There are two methods you can use to store custom data. Choose the appropriate method based on what you want to do with the custom data:

Storing custom data using name-value pairs

If you want to use the extra data only for use in event enrichment or viewing in the Structure Browser, then you can add custom data using name-value pairs. You do not need to create custom database tables.

This method can be easier to implement and upgrade because you do not need to change the default database schemas.

Storing custom data in new database tables

If you want to use the data not only for event enrichment or viewing in the Structure Browser, but also for custom polling, finding devices in the Hop View, or to use in defining Network Views, you must create new database tables.

Reasons for adding custom data

The reasons for adding custom data determine the appropriate method of storing the data. Common reasons for adding custom data to your topology include enriching the data that is associated with a device or interface, and modeling custom connectivity between devices.

Storing custom data in new database tables

If you want to filter or search on the data, store the data in custom tables. This method of storing custom data is suitable for the following example uses.

- Creating different reports for different customers
- Implementing different poll policies for different providers based on Service Level Agreements
- Creating a network view based on location

All the previous uses involve identifying the relevant entities based on the custom data fields. Storing the data in new database tables is more performant for these uses than storing the data as name-value pairs.

Storing custom data as name-value pairs in the entityDetails database table

If you want to include the custom data as part of another operation, store the data as name-value pairs in the entityDetails database table. This method of storing custom data is suitable for the following example uses.

- Adding customer names to reports that filter on other data
- Tagging network events with location or customer name

© Copyright IBM Corp. 2006, 2021

In these cases, the relevant entities are selected based on other database fields. Entities are selected by SQL queries or Event Gateway lookups. The custom data is extracted from the selected entities if it is present. For these uses, you can alternatively store the data in new database tables, but it is quicker to store data as name-value pairs in the entityDetails database table.

Modeling custom connectivity between entities

Custom connectivity is modeled in Network Manager by using layers. Examples of connectivity modeled by default by using layers include the following types:

- Cisco Discovery Protocol, the protocol that is used between Cisco communications devices.
- SynOptics Network Management Protocol, the protocol that is used between Nortel communications devices.

If you want to model custom connectivity between entities, then you must store the data in new database tables. You must write a custom agent to retrieve the relevant custom data from your network entities. You must also write a custom layer stitcher to build the custom connectivity from the agent data. Examples of default layer stitchers that are provided with Network Manager are `CDPLayer.stch`, `OSPFLayer.stch`, `SONMPLayer.stch`, and `SRPLayer.stch`. You can inspect these stitchers to help you write your custom layer stitcher.

Discovering custom data

To use custom data in the product, you must first add the information to the discovery.

About this task

You can use different methods to discover custom data.

Adding name-value pairs to entities using the File finder

If you enable the File finder to seed your discovery, then you can add name-value pairs to entities by adding extra columns to the seed file read by the File finder.

About this task

The example procedure below assumes that you are adding the following extra columns to your File finder seed file:

- customer
- location

The following example text file fragment shows what the seed file might look like:

```
vi /var/tmp/logged_hosts

172.16.1.21       lnd-dharma-acme       acme        london
172.16.1.201      lnd-phoenix-acme      acme        london
172.16.1.25       prs-sun-acme          acme        paris
172.16.2.33       ranger1               telecorp    newyork
172.16.2.34       ranger2               telecorp    newyork
~
"/var/tmp/logged_hosts" [Read only] 4 lines, 190 characters
```

In this example text file fragment, the third column holds customer information, and the fourth column holds location information.

Procedure

1. Edit the DiscoFileFinderParseRules.cfg configuration file.
2. In this configuration file, configure the File finder to parse the seed file using an insert similar to the example. Ensure that you configure the `m_ColDefs` field to match the new custom tag columns.

```
insert into fileFinder.parseRules
(
            m_FileName,
            m_Delimiter,
            m_ColDefs
)
values
(
            "/var/tmp/logged_hosts",
            "[     ]",
            [
                {
                    m_VarName="m_UniqueAddress",
                    m_ColNum=1
                },
                {
                    m_VarName="m_Name",
                    m_ColNum=2
                },
                {
                    m_VarName="m_CustomTags->customer",
                    m_ColNum=3
                },
                {
                    m_VarName="m_CustomTags->location",
                    m_ColNum=4
                }
            ]
);
```

This insert instructs the File finder to do the following:

- Parse `/var/tmp/logged_hosts`.
- Treat white space as the data separator.
- Use the following column definitions:
  - `m_UniqueAddress` for the first column
  - `m_Name` for the second column
  - `m_CustomTags->customer` for the third column
  - `m_CustomTags->location` for the fourth column

3. Edit the DbEntityDetails.cfg file and configure an insert similar to the following:

```
insert into dbModel.entityDetails
(
    EntityType,
    EntityDetails
)
values
(
    1,  -- chassis
    {
        Customer = "eval(text, '&m_ExtraInfo->m_CustomTags->customer')",
        Location = "eval(text, '&m_ExtraInfo->m_CustomTags->location')"
    }
);
insert into dbModel.entityDetails
(
     EntityType,
     EntityDetails
)
values
(
     2,  -- port/interface
     {
         Customer = "eval(text, '&m_ExtraInfo->m_CustomTags->customer')",
         Location = "eval(text, '&m_ExtraInfo->m_CustomTags->location')"
     }
);
```

4. Restart Network Manager to propagate the configuration file changes:

```
itnm_start ncp -domain domain
```

5. Run a full discovery using the File finder to discover your network with the name-value pairs added.

What to do next

Ensure that the new data is propagated between discoveries.

Developing agents or collectors to retrieve custom data

You can develop custom discovery agents or collectors to retrieve custom data and add it to the discovery.

About this task

You can retrieve custom data from a third-party source, such as a flat file, database, or third-party system. Do this by developing a Perl agent or an EMS collector.

Important:

Writing custom discovery agents or collectors is an advanced procedure. Incorrect inserts into discovery databases can corrupt discovery and cause unpredictable results. Consider engaging IBM Services or an IBM Business Partner.

Use the following information to determine the best method to retrieve data:

Table 76. Methods for retrieving data

Source system	Method	Link
EMS	EMS collector	"Configuring EMS discoveries" on page 245
Flat file or database	Custom agent	For information about writing discovery agents, see the *IBM Tivoli Network Manager Reference*.For information about writing discovery agents, see the *IBM Tivoli Network Manager Reference*.

Adding data manually using custom tag tables

You can add name-value pairs to entities by creating inserts containing the name-value pair data into the disco.ipCustomTags table or into the disco.filterCustomTags table.

Adding name-value pairs to entities using the disco.ipCustomTags table

You can associate name-value pair tags to unique IP addresses using the disco.ipCustomTags table. Use this method if you know the individual IP addresses that you want to tag.

About this task

The example procedure below assumes that you are adding the following two custom name-value pair tags to entities in your discovery:

- customer
- location

This example uses the disco.ipCustomTags table to configure the following name-value pair tags:

Table 77. Example name-value pairs

IP Address	Name	Value
172.16.1.21	customer	acme

Table 77. Example name-value pairs (continued)		
IP Address	**Name**	**Value**
172.16.1.21	location	london
172.16.1.201	customer	acme
172.16.1.201	location	london
172.16.1.25	customer	acme
172.16.1.25	location	paris
172.16.2.33	customer	telecorp
172.16.2.33	location	newyork
172.16.2.34	customer	telecorp
172.16.2.34	location	newyork

Procedure

1. Edit the `DiscoConfig.cfg` configuration file.
2. In this configuration file, add an insert similar to the following.

```
insert into disco.ipCustomTags
(
            m_UniqueAddress,
            m_CustomTags
 )
values
(
            '172.16.1.21',
            {
                customer="acme",
                location="london"
            }
);

insert into disco.ipCustomTags
(
            m_UniqueAddress,
            m_CustomTags
 )
values
(
            '172.16.1.201',
            {
                customer="acme",
                location="london"
            }
);

insert into disco.ipCustomTags
(
            m_UniqueAddress,
            m_CustomTags
 )
values
(
            '172.16.1.25',
            {
                customer="acme",
                location="paris"
            }
);

insert into disco.ipCustomTags
(
            m_UniqueAddress,
            m_CustomTags
 )
values
```

```
(
            '172.16.2.33',
            {
                customer="telecorp",
                location="newyork"
            }
);

insert into disco.ipCustomTags
(
            m_UniqueAddress,
            m_CustomTags
 )
values
(
            '172.16.2.34',
            {
                customer="telecorp",
                location="newyork"
            }
);
```

3. Save the `DiscoConfig.cfg` configuration file.

What to do next

You can now run a full discovery to discover your network with the custom tags.

Adding name-value pairs to entities using the disco.filterCustomTags table

You can associate name-value pair tags to a filtered set of IP addresses using the disco.filterCustomTags table. Use this method if you want to tag multiple addresses using a shared attribute of those devices.

About this task

You can filter IP addresses based on a wide variety of criteria. For example, you can filter based on device name, based on IP address, or based on VLAN identifier. The example procedure below applies a filter based on IP address, and uses the disco.filterCustomTags table to configure the following name-value pair tags to all IP addresses in the subnet 172.20.3.0/24:

Table 78. Example name-value pairs

IP Address	Name	Value
172.20.3.0/24	customer	acme
172.20.3.0/24	location	london

Procedure

1. Edit the DiscoConfig.cfg configuration file.
2. In this configuration file, add the following insert:

```
insert into disco.filterCustomTags
(
            m_Filter,
            m_CustomTags
 )
values
(
            "m_UniqueAddress LIKE '172.20.3'",
            {
                customer="acme",
                location="london"
            }
);
```

3. Save the DiscoConfig.cfg configuration file.

Other examples of filters

The procedure above applies a filter based on IP address: `"m_UniqueAddress LIKE '172.20.3'"`.

You can create a filter based on any attributes associated with discovered entities. For example, you could apply the following filters:

- Filter based on entity name: `"m_Name LIKE 'lon'"`
- Filter based on VLAN identifier of a VLAN entity: `"m_LocalNbr->m_VlanID = 102"`

What to do next

You can now run a full discovery to discover your network with the custom tags.

Enriching the topology using the GetCustomTag stitcher

You can use the GetCustomTag stitcher to add data to topology entities. You might choose to use this method if you have complex customization needs.

Before you begin

If you want to apply a fixed tag to matched entities, you can use the m_CustomTags field, and you do not need to use the GetCustomTag stitcher.

About this task

If you want to add more complex data, then customize the GetCustomTag.stch stitcher. The tag configured in the `m_StitcherTagName` field is used as the field name, and the value returned by the GetCustomTag stitcher is used as the field value.

Note: Editing stitchers is an advanced task, which requires knowledge of the Stitcher Language and the discovery data flow.

To customize the `GetCustomTag.stch` stitcher, complete the following steps:

Procedure

1. Add data to the disco.filterCustomTags or disco.ipCustomTags table to specify which entities to apply custom tags to.
2. Edit the `DiscoConfig.cfg` configuration file to configure the GetCustomTag stitcher.
3. In this configuration file, add the following insert:

```
insert into disco.filterCustomTags
            (
                        m_Filter,
                        m_StitcherTagName,
             )
            values
            (
                        "m_UniqueAddress LIKE '172.20.3.'",
                        'customer'
            );
```

   This insert configures the system to use the `GetCustomTag.stch` stitcher to look up the value of the `customer` field for each entity in the subnet 172.20.3.0/24. The `GetCustomTag.stch` stitcher must be modified to provide this information, otherwise the name/value pair is not added to the entity.
4. Save the `DiscoConfig.cfg` configuration file.
5. Modify the `GetCustomTag.stch` stitcher to support the custom tag.

   The `GetCustomTag.stch` stitcher receives the tag and entity name as stitcher arguments 1 and 2 respectively and returns a text value. Customize the `GetCustomTag.stch` stitcher to determine the appropriate return values, depending on what you want to use the data for. In the following example, the stitcher is customized to check for the string `lon` in the supplied entity name. If the entity name

contains the string, the value `A-Z Inc., London` is returned as the value for `customer`. If the entity name does not contain the string, none is returned.

```
UserDefinedStitcher
{
    StitcherTrigger
    {
        // Called from another stitcher when tag criteria is met
    }

    StitcherRules
    {
        text tagName = eval(text,'$ARG_1');
        text entityName = eval(text,'$ARG_2');

        text value = NULL;

        if(tagName == "customer")
        {
            // insert logic to retrieve custom tag
            //
            // As an example, here we use pattern matching to
            // assign all entities with 'lon' in their name
            // to the A-Z Inc. customer.
            //
            int count = MatchPattern(entityName, '(lon)');
            if (count == 1)
            {
                value = "A-Z Inc., London";
            }
            else
            {
                // Not an A-Z Inc. device.
                // If we leave value as NULL then no
                // 'customer' custom tag will be
                // created, however in this example we
                // want such entities tagged with
                // 'none'..
                value = "none";
            }
        }

        SetReturnValue(value);
    }
}
```

Note: Only entities that pass the criteria configured in the `disco.filterCustomTags` or `disco.ipCustomTags` table are passed to the `GetCustomTag.stch` stitcher.

Results

The following custom name-value pair tag is added to all IP addresses in the subnet 172.20.3.0/24:

Table 79. Data added to entities

IP Address	Name	Value
172.20.3.0/24	Customer	A-Z Inc., London

What to do next

You must now configure the `DbEntityDetails.cfg` configuration file to ensure that, following discovery, the NCIM topology database entityDetails table is updated with the custom tags.

The example GetCustomTag stitcher

This example explains how an example GetCustomTag stitcher retrieves the customer name associated with a device.

How the GetCustomTag stitcher is called

The `GetCustomTag.stch` stitcher is called by the `AddCustomTags.stch` stitcher. You do not need to modify the AddCustomTags stitcher. You do need to modify the GetCustomTag stitcher.

The AddCustomTags stitcher loops through the tags and the entities in the disco.ipCustomTags and disco.filterCustomTags tables. If, in either of these tables, the `m_StitcherTagName` field is set, then the AddCustomTags stitcher calls the GetCustomTag stitcher and passes it the relevant entity name and the `m_StitcherTagName` field as parameters. The `m_StitcherTagName` field holds the name part of a name-value pair tag, for example, `Customer`.

If the GetCustomTags stitcher returns a non-NULL value for a tag or entity then the AddCustomTags stitcher updates the `m_ExtraInfo->m_CustomTags` object in the workingEntities.finalEntity table record for the entity. The tag is used as the field name and the return value is used as the field value. From the workingEntities.finalEntity database table, the data passes to the DNCIM database, the ncimCache database, and the NCIM database.

Note: The AddCustomTags stitcher retrieves the entity name by performing a lookup in the workingEntities.finalEntity table. The stitcher uses the IP address in the disco.ipCustomTags table or the m_Filter as the WHERE clause provided in the disco.filterCustomTags table.

Description of an example GetCustomTag stitcher

The GetCustomTag stitcher takes as input a single entity name and a tag (the `m_StitcherTagName` field). You write custom logic in the stitcher to evaluate the value part of the name-value pair. By default the stitcher contains example code similar to that described here. You can customize this stitcher to work with different name-value pairs and you can change the logic defining how the value is calculated.

Table 80. Line-by-line description of an example GetCustomTag stitcher

Line numbers	Description
10	Set the value of the tagName variable from the first argument received from the AddCustomTags stitcher. This value is the name of the tag for which the value is to be evaluated.
11	Set the value of the entityName variable from the first argument received from the AddCustomTags stitcher. The entityName variable holds the entity name associated with the IP address for which the stitcher is evaluating the value of a name-value pair tag.
13	Set the value variable to NULL. The value variable is returned by the stitcher and holds the evaluated value of the name-value pair tag.
15	Test for a supported tag. In this case the tag is `customer`. You can add support for further custom tags by adding more `else if` blocks.
16-35	The `customer` tag lookup code. If the entity name contains the text pattern `lon` then set the value variable to the customer name `A-Z Inc. London`, otherwise set it to `none`.
37	Return the value of the tag.

```
UserDefinedStitcher
     {
         StitcherTrigger
         {
             // Called from another stitcher when tag criteria
             // is met
         }
```

```
        StitcherRules
        {
           text tagName = eval(text,'$ARG_1');
           text entityName = eval(text,'$ARG_2');

           text value = NULL;

           if(tagName == "customer")
           {
               // insert logic to retrieve custom tag
               //
               // As an example, here we use pattern matching
               // to assign all entities with 'lon'
               //  in their name to the A-Z Inc. customer.
               //
              int count = MatchPattern(entityName, '(lon)');
               if (count == 1)
               {
                   value = "A-Z Inc., London";
             }
               else
               {
                   // Not an A-Z Inc. device.
                   // If we leave value as NULL then no 'customer'
                   // custom tag will be created,
                   // however in this example we want
                   // such entities tagged with 'none'..
                   value = "none";
               }
           }

           SetReturnValue(value);
       }
   }
```

Storing custom data as name-value pairs in the entityDetails table

If you want to use the extra data for uses such as adding to reports, enriching network events, or viewing in the **Structure Browser**, then you can store custom data as name-value pairs in the existing entityDetails database table. You do not need to create new database tables.

About this task

The entityDetails database table is a default topology database table. You can add custom data to this table by inserting it as name-value pairs. This method is easier than creating new database tables, but less performant if you need to select entities based on the data.

Tip: If you want to use the data for custom polling, finding devices in the Hop View, or to use in defining Network Views, create new database tables instead. Use the procedure at: “Storing custom data in new database tables” on page 428.

To discover custom data and store it as name-value pairs, you need to discover the data, ensure that the data persists over multiple discoveries, and configure how the data is added to the DNCIM discovery database.

The data flow for discovering custom data and storing it as name-value pairs is as follows:

1. The data is retrieved from the source (flat file, EMS, or other source) by a discovery agent, collector, or Finder.
2. The name-value pairs are inserted into the workingEntities.finalEntity database table.
3. The **ncp_disco** process uses the dbModel.entityDetails discovery database table to populate the entityDetails table in the DNCIM discovery database with the information from the workingEntities.finalEntity table.
4. The data in the entityDetails table in DNCIM is automatically copied to the entityDetails table in NCIM and the ncimCache.entityData table.
5. From the ncimCache.entityData table, the data is available to the Event Gateway for event enrichment. If events are enriched with the data, the information is available in the **Event Viewer**.

6. From the NCIM topology database, the data is available to the **Structure Browser**.

The following table shows an example of the kind of data you might add to a device as name-value pairs.

Table 81. Example of name-value pair tags

IP address	Name	Value
172.20.3.20	customer	acme
172.20.3.20	location	london

Importing name-value pairs into the DNCIM discovery database

After the name-value pairs have been discovered, they must be inserted into the DNCIM discovery database. Configure the DbEntityDetails.cfg configuration file so that the name-value pairs are inserted into the DNCIM discovery database.

About this task

The **ncp_disco** process uses the dbModel.entityDetails discovery database table to populate the entityDetails table in the DNCIM discovery database with the information from the workingEntities.finalEntity table. To configure an insert into the dbModel.entityDetails table, complete the following steps:

Procedure

1. Edit the DbEntityDetails.cfg configuration file.

   Note: Editing the ModelNcimDb.cfg configuration file makes it more difficult to migrate your customizations. Edit the DbEntityDetails.cfg configuration file instead.
2. Add an insert into the dbModel.entityDetails database table to extend the default EntityType filter.

   You can have only one entityDetails mapping per EntityType filter. For example, append a mapping similar to the following code:

```
insert into dbModel.entityDetails
(
    EntityType,
    EntityDetails
)
values
(
    1,  -- chassis
    {
        NetworkEdge = "eval(text, '&m_ExtraInfo->m_NetworkEdge')",
        CustomerName = "eval(text, '&m_ExtraInfo->m_CustomerName')",
        CustomerType = "eval(text, '&m_ExtraInfo->m_CustomerType')"
    }
);
```

3. Save and close the DbEntityDetails.cfg configuration file.

Results

The next time that a full discovery is run, the name-value pairs are inserted into the entityDetails table in the DNCIM discovery database. The data in the entityDetails table in DNCIM is automatically copied to the entityDetails table in NCIM and the ncimCache.entityData table.

What to do next

Ensure that custom name-value pairs are preserved between discoveries.

Preserving custom name-value pairs between discoveries

By default, custom data added to the dbModel.entityDetails table using inserts into the DbEntityDetails.cfg file is deleted if it is not present in the next discovery. You can configure discovery to preserve this custom data.

About this task

To configure the discovery to keep custom name-value pairs that were added to the dbModel.entityDetails table, enable the KeepOldEntityDetails setting.

If the KeepOldEntityDetails setting is enabled, the data is preserved in future discoveries. The values are updated if new information is discovered.

To enable the KeepOldEntityDetails setting, complete the following steps:

Procedure

1. Back up and edit the `NCHOME/etc/precision/ModelSchema.cfg` file.
2. Locate the insert into the model.config table and set the KeepOldEntityDetails field to 1 (enabled).

   For example, the insert might look like this:

```
insert into model.config
(
    LingerTime,
    ChassisCreationEvents,
    IpInterfaceCreationEvents,
    MaintenanceStateEvents,
    ManagedStatusUpdateInterval,
    DeleteRenamedDevices,
    KeepOldEntityDetails
)
values
(
    3,
    0,
    0,
    0,
    30,
    1,
    1,
);
```

Storing custom data in new database tables

If you want to use the data for custom polling, finding devices in the Hop View, or to use in defining Network Views, you must create new database tables.

About this task

If you want to use custom data for applications that involve selecting entities based on the custom data, it is more performant to use custom database tables.

Tip: If you want to use the extra data only for applications where entities are selected based on other database fields, for example in event enrichment or viewing in the Structure Browser, you do not need to create custom database tables. You can store custom data more easily as name-value pairs in the entityDetails database table, following the procedure at: “Storing custom data as name-value pairs in the entityDetails table” on page 426.

To store custom data in new database tables, you need to do the following tasks:

- Create new tables in the NCIM topology database.
- Create the same tables in the DNCIM discovery database.
- Discover the data.

- Configure how the data is added to the DNCIM discovery database.

The data flow for discovering custom data by creating custom database tables is as follows:

1. The data is retrieved from the source (flat file, EMS, or other source) by a discovery agent or collector.
2. The data is inserted into the workingEntities.finalEntity database table.
3. The **ncp_disco** process uses the dbModel.entityMap discovery database table to populate the new custom table in the DNCIM discovery database with the information from the workingEntities.finalEntity table.
4. The data in the new custom table in DNCIM is automatically copied to the equivalent table in NCIM and the ncimCache.entityData table.
5. From the ncimCache.entityData table, the data is available to the Event Gateway for event enrichment. If events were enriched with the data, the information is available in the **Event Viewer**.
6. From the NCIM topology database, the data is available to the **Network Hop View**, **Network Views**, and **Structure Browser**.

Creating new tables in the NCIM topology database

Create new tables in the NCIM database to hold the custom data that you want to discover.

About this task

If you create the NCIM database tables first, you can plan what you want to use the data for, before you configure how it is discovered.

Procedure

1. Define the new table in NCIM by using the command line tool for your database, or **ncp_oql**.
2. Call the first field of the new table `entityID` and define the field as a foreign key to the entityData table.

   Defining the `entityID` field in this way links the extra information about the entity to the main NCIM entity register. If the entity is removed from the entityData table, the data is also deleted from the custom table.
3. Restart Network Manager.
4. Optional: Insert some example data into the new table and test if it is suitable for your purpose.

Creating new tables in different databases

The following example SQL statement shows how to create a custom table `exampleCustomData` in NCIM for Db2. The table used in the example is based on the example dNCIM customisation defined in the `createCustomization.sql` file supplied with Network Manager.

```
CREATE TABLE exampleCustomData
(
    entityId INTEGER NOT NULL,
    slaId   VARCHAR(255),
    customerId              VARCHAR(255),
    customerContact         VARCHAR(255),

    CONSTRAINT example_cd_pk PRIMARY KEY (entityId),

    CONSTRAINT excd_entData_fk FOREIGN KEY (entityId)
    REFERENCES entityData(entityId) ON DELETE CASCADE
);
```

The following example SQL statement shows how to create a custom table `exampleCustomData` in NCIM for Oracle. The table used in the example is based on the example dNCIM customisation defined in the `createCustomization.sql` file supplied with Network Manager.

```
CREATE TABLE exampleCustomData
(
```

```
    entityId NUMBER(10) NOT NULL,
    slaId   VARCHAR2(255),
    customerId              VARCHAR2(255),
    customerContact         VARCHAR2(255),
--
    CONSTRAINT              example_cd_pk PRIMARY KEY (entityId),
--
    CONSTRAINT excd_entData_fk FOREIGN KEY (entityId)
    REFERENCES entityData(entityId) ON DELETE CASCADE
);
```

What to do next

Now create the same table in the DNCIM discovery database.

Updating dNCIM to store custom data

Update dNCIM to store custom data by first writing an SQL script to define the tables to store the custom data.

About this task

To support the creation of the new SQL script and to ensure that the custom tables defined in the script are automatically recreated if the dNCIM database is dropped, the following pieces of sample code are provided. You can use these pieces of code as a starting point

Code to define the custom dNCIM tables

A sample SQL script is provided called `createCustomization.sql`, located at `$NCHOME/precision/scripts/sql/sqlite/createCustomization.sql`. This script contains an example of the SQL needed to create a custom table to store custom data:

```
CREATE TABLE dncim.exampleCustomData
(
    entityId                    INTEGER NOT NULL,
    slaId                       VARCHAR(255),
    customerId                  VARCHAR(255),
    customerContact             VARCHAR(255),

    CONSTRAINT example_cd_pk PRIMARY KEY (entityId),

    CONSTRAINT exCd_entData_fk FOREIGN KEY (entityId)
    REFERENCES entityData(entityId) ON DELETE CASCADE
);
```

To update dNCIM to store custom data, perform the following steps:

Procedure

1. Modify the sample SQL script `createCustomization.sql` script to define the tables to store your custom data.
2. Save the file `$NCHOME/precision/scripts/sql/sqlite/createCustomization.sql`.
3. Delete the old dNCIM database directory `$NCHOME/precision/embeddedDb/sqlite/ncp_disco.`*`domain_name`*`/`.

   Where *domain_name* is the name of the relevant domain.

   The dNCIM database directory `$NCHOME/precision/embeddedDb/sqlite/ncp_disco.`*`domain_name`*`/` will be recreated when you next restart the Discovery engine, ncp_disco. The new dNCIM database directory will pick up the changes that you specified in the `createCustomization.sql` SQL script.

What to do next

You must also configure NCIM to store the custom data.

Mapping retrieved data to DNCIM custom data tables

Specify how custom data from the discovery should be stored in the topology database by mapping the custom data to the custom tables added to dNCIM and NCIM.

About this task

Define inserts within the `$NCHOME/etc/precision/DbEntityDetails.cfg` configuration file to specify how to map the custom data to the dNCIM custom tables.

Example: add single row of custom data about existing entity

To add a single row of custom data, add the appropriate insert to the `$NCHOME/etc/precision/DbEntityDetails.cfg` configuration file.

Inserting data discovered using the File finder, a discovery agent, or collector

The following insert adds a single row of custom data about an existing entity. In the mapping, the values on the left are the field names of a new table called `exampleCustomData`. The `exampleCustomData` table must be created in dNCIM and NCIM. The eval statements on the right take the values from the `workingEntities.finalEntity` database table. This data must already be discovered by using one of the methods for discovering custom data and added to the `workingEntities.finalEntity` database table.

In this example, service-level agreement data for a an existing entity is mapped to the custom dNCIM table `exampleCustomData`.

In this example, the data was discovered using the File finder, a discovery agent, or collector. The data was added to custom fields within the `m_ExtraInfo` field in the `workingEntities.finalEntity` database table.

```
insert into dbModel.entityMap
(
   EntityFilter,
   TableName,
   FieldMap
)
values
(
   "m_ExtraInfo->m_CustomerSLA IS NOT NULL",
   "exampleCustomData",
   {
       entityId           = "eval(int, '&m_EntityId')",
       slaId  = "eval(text, '&m_ExtraInfo->m_CustomerSLA')",
       customerId         = "eval(text, '&m_ExtraInfo->m_CustomerId')",
       customerContact    = "eval(text, '&m_ExtraInfo->m_CustomerContact')"
    }
 );
```

The table below describes the insert.

Table 82. Description of the insert

Line	Description
9	Only add rows to the custom table where the `m_ExtraInfo->m_CustomerSLA` custom field for the entity has a value. This filters out entities that are not associated with a service-level agreement.
10	Add the rows to the custom dNCIM table `exampleCustomData`.

Table 82. Description of the insert (continued)	
Line	**Description**
11-16	Add the following data to each row: • Entity identifier. • Service level agreement identifier. • Customer identifier. • Customer contact.

Inserting data discovered using custom tag tables

The following insert adds a single row of custom data about an existing entity. In the mapping, the values on the left are the field names of a new table called `exampleCustomData`. The `exampleCustomData` table must be created in dNCIM and NCIM. The eval statements on the right take the values from the `workingEntities.finalEntity` database table. This data must already be discovered by using one of the methods for discovering custom data and added to the `workingEntities.finalEntity` database table.

In this example, service-level agreement data for a an existing entity is mapped to the custom dNCIM table `exampleCustomData`.

In this example, the data was discovered using custom tag tables. Data discovered using custom tag tables is added to custom fields within the `m_ExtraInfo->m_CustomTags` field in the `workingEntities.finalEntity` database table.

```
insert into dbModel.entityMap
(
   EntityFilter,
   TableName,
   FieldMap
)
values
(
   "m_ExtraInfo->m_CustomTags->m_CustomerSLA IS NOT NULL",
   "exampleCustomData",
   {
       entityId           = "eval(int, '&m_EntityId')",
       slaId              = "eval(text, '&m_ExtraInfo
                             ->m_CustomTags->m_CustomerSLA')",
       customerId         = "eval(text, '&m_ExtraInfo
                              ->m_CustomTags->m_CustomerId')",
       customerContact    = "eval(text, '&m_ExtraInfo
                                 ->m_CustomTags->m_CustomerContact')"
    }
 );
```

The table below describes the insert.

Table 83. Description of the insert	
Line	**Description**
9	Only add rows to the custom table where the `m_ExtraInfo->m_CustomTags->m_CustomerSLA` custom field for the entity has a value. This filters out entities that are not associated with a service-level agreement.
10	Add the rows to the custom dNCIM table `exampleCustomData`.

Table 83. Description of the insert (continued)	
Line	**Description**
11-16	Add the following data to each row: • Entity identifier. • Service level agreement identifier. • Customer identifier. • Customer contact.

Using custom data to enrich events

You can use custom data in event enrichment in a similar way to normal data.

Before you begin

Ensure that you are familiar with how to do event enrichment with normal data. If you want to use a new field in the ObjectServer alerts.status table to store the enriched data, you must create the field in the ObjectServer first before following the steps below.

For information on how to add a custom field to an ObjectServer table, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

About this task

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 Reference*.

The following steps explain how to configure this extra event enrichment.

Procedure

1. Edit the Event Gateway schema file, `$NCHOME/etc/precision/EventGatewaySchema.cfg`, to allow the Event Gateway to update the new field. Remember to add a comma at the end of the line containing the `NmosSerial` field, before the line containing the new `InterfaceName` field.
   For example, to configure the Event Gateway to update a new field `customer`, add the text in bold to the outgoing event filter.

```
insert into config.ncp2nco
(
    FieldFilter
)
values
(
    [
        "NmosCauseType",
        "NmosDomainName",
        "NmosEntityId",
        "NmosManagedStatus",
        "NmosObjInst",
        "NmosSerial",
        "Customer"
    ]
);
```

   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 StandardEventEnrichment stitcher 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 customer information from the topology database and to populate the Customer field.

   One way to do this is by adding code to the StandardEventEnrichment stitcher. The code is different depending whether you added custom data as name-value pairs to the entityDetails table or created a custom database table. Adding this code to the stitcher 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 Reference*.

   The following example code takes the value of `customerId` from the entityDetails table in the ncimCache files. Use this method if you added custom data without creating new NCIM database tables.

```
if ( entityType == 1 )
{
     text customerName = @entity.entityDetails.customerId;

    if ( customerName <> eval(text, '&Customer') )
    {
          @enrichedFields.Customer = customerName;
    }
}
```

Table 84. Lines of code relevant to the interface name example

Line numbers	Description
1	This event enrichment is only relevant for chassis events. Check that this event relates to a chassis by ensuring that the entityType value is 1, and if so, continue processing.
3	Retrieve the customer data from the entityDetails table. **Note:** Fields within the NCIM cache are usually in all uppercase, for example, `@mainNode.chassis.SYSLOCATION`. In this example, `customerName` is in mixed case because it is in the entityDetails table.
5 - 8	Only populate the Customer field if the value is not already present in the in-scope event.

   The following example code takes the value of `customerName` from the `customer` table in the ncimCache files. Use this method if you added custom data to a new NCIM database table.

```
if ( entityType == 1 )
{
     text customerName = @entity.customer.CUSTOMERNAME;

    if ( customerName <> eval(text, '&CustomerName) )
    {
          @enrichedFields.CustomerName = customerName;
    }
}
```

Visualizing custom data in the Structure Browser

If you add custom data using name-value pairs in the `entityDetails` table or by creating new database tables, you can see the data in the **Structure Browser**.

About this task

You do not need to perform any extra configuration steps to see custom data in the `entityDetails` table in the **Structure Browser**.

To see new database tables in the **Structure Browser**, complete the following steps.

Procedure

1. Go to the $NMGUI_HOME/profile/etc/tnm/ directory.
2. Edit the ncimMetaData.xml file and locate the section that defines the entityType with which you want to associate the custom data. This entityType is most commonly 1, that is, a chassis.
3. Add the custom data by adding lines similar to the following lines in bold:

```
    <entityMetaData entityType="1" table="physicalChassis" manager="PrecisionIP"
              entitySearch="true">

        <dataField tableAlias="e" dataType="int" column="entityId"
         entitySearch="false"/>
        <dataField tableAlias="e" dataType="str" column="entityName"
         entitySearch="false"/>
        <dataField tableAlias="e" dataType="str" column="displayLabel"
         entitySearch="false"/>
        <dataField tableAlias="e" dataType="bool" column="manual"
         entitySearch="true"/>

...

        <dataField tableAlias="x" dataType="str" column="customerId"/>
        <dataField tableAlias="x" dataType="str" column="slaId"/>
        <dataField tableAlias="x" dataType="str" column="customerContact"/>

        <fromTables>
            FROM _ncim_.entityData e

...

            LEFT OUTER JOIN _ncim_.exampleCustomData x ON x.entityId =
            e.entityId
```

In the preceding example, a left join against the custom table is used. This join ensures that data is displayed for devices that do not have an entry in the custom database table. The fields to be displayed are listed, and the tableAlias matches the alias given in the left join statement.

Using custom data in polling

You can poll a set of network devices based on custom data.

About this task

To define a polling scope based on custom data, complete the following steps.

Procedure

1. Create a network view based on the custom data that you want to use. Refer to the task “Visualizing custom data in the topology views” on page 435 for more information.
2. Create or edit a poll policy. Specify the new network view as the scope for the poll policy.

Visualizing custom data in the topology views

If you added custom data to a new NCIM database table, you can search on the custom data, and you can create network views with the custom data.

About this task

It is possible to create network views based on custom data that was added as name-value pairs, but these network views are not efficient. For better performance, create network views based on only custom data that was added in new database tables.

To enable visualization of custom attributes in the topology views, complete the following tasks:

Procedure

1. Go to the $NMGUI_HOME/profile/etc/tnm/ directory.
2. Edit the ncimMetaData.xml file by appending lines similar to the following example:

```
<entityMetaData table="customer" manager="AllManagers" entitySearch="true">
     <dataField dataType="str" column="customerName"/>
     <dataField dataType="str" column="customerType"/>
</entityMetaData>
```

   In the preceding example, the custom table customer is added. The manager attribute must always be set to AllManagers. Set the attribute entitySearch to true to use the data in searches. Each dataField is a single column of the table to be displayed and must have the correct dataType.

Results

Operators can now select the table when they search in the **Hop View** or when they create a new dynamic network view.

Part 3. Polling the network

Poll the network to retrieve information from network devices that you can use to monitor the behavior of the devices.

About this task

For more information about administrator tasks related to network polling, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

© Copyright IBM Corp. 2006, 2021

Chapter 19. 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.

Note: Network Manager does not poll non-IP entities, such as layer 1 optical devices and radio access network devices. These devices are automatically set to unmanaged status.

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 pollers cannot handle the polling demands for your network, you might need to confgure additional pollers.

Related tasks

Administering multiple pollers
If additional pollers are needed to poll your network, you can set up new pollers. You can add pollers or remove pollers, or use a poller ID to associate a specific poller with a policy.

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

Default poll policies
Network Manager provides a set of default poll policies. Use this information to familiarize yourself with these policies.

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: 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.

  Note: A poll policy can have one or more poll definitions associated with it. This can be useful, for example, when you want to poll information that is specific to the device vendor. In such cases you need to set up a poll definition for each vendor (because each vendor might have different MIBs), but have only one policy with all poll definitions added to get the data from across your network.

© Copyright IBM Corp. 2006, 2021

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 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.

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.

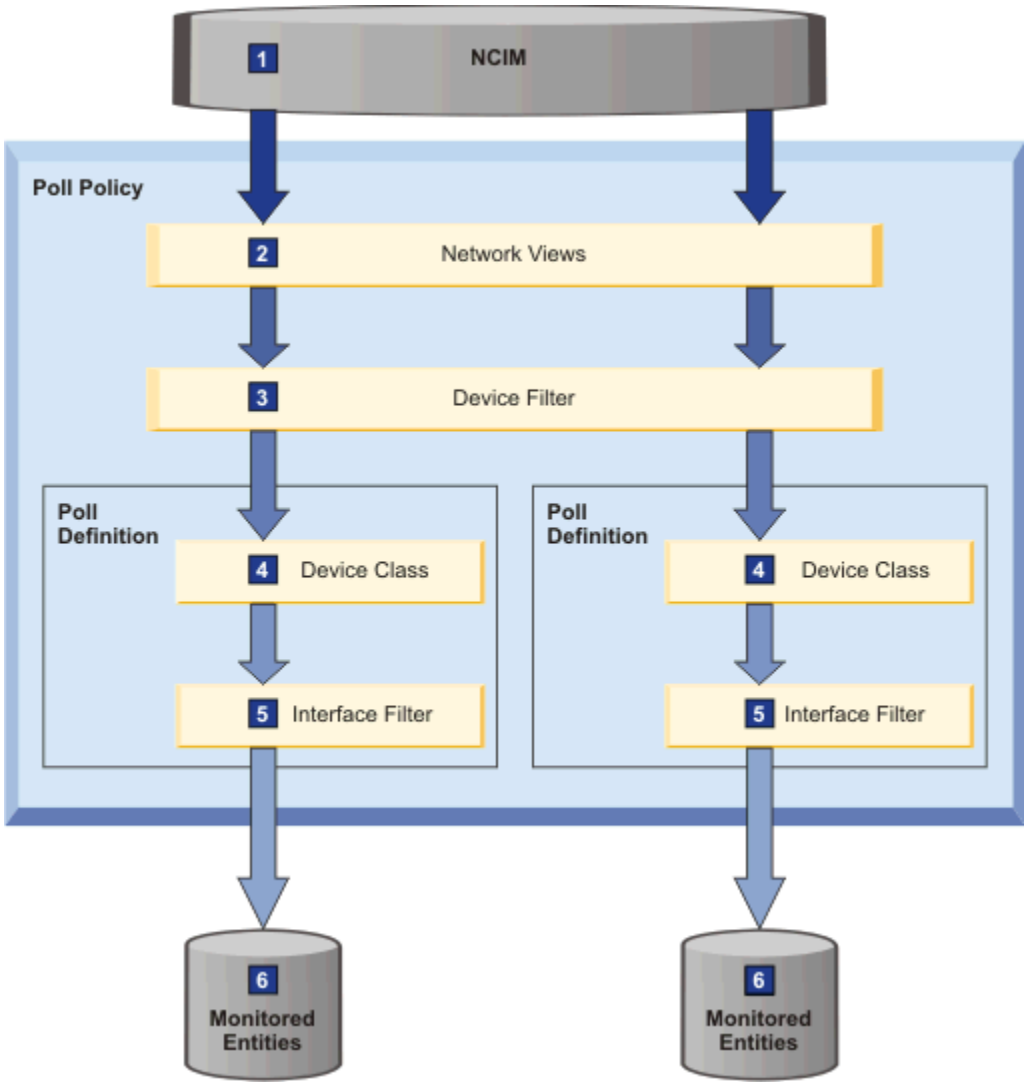


Figure 7. Poll policy scope

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

Default poll definitions

Network Manager 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 User 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*.
- Basic threshold poll definition type only: data units. Specify the data units for this poll definition. The appropriate data unit varies depending on the type of data in the poll definition. Here are some typical

data types and the data units that correspond to those data types, together with examples from the default Network Manager poll definitions:

Counts
Poll definitions where the value is a count of some item. Examples include:

- dot3StatsAlignmentErrors
- ifInDiscards
- ifInErrors

For this type of poll definition specify a data unit of #.

Percentages
Poll definitions where the value is a percentage. Examples include:

- cpuBusyPoll
- ciscoCPUTotal5min

For this type of poll definition specify a data unit of %.

Specific units of measure
Poll definitions where the value is a specify unit of measure. Examples include:

- memoryPoll

For this type of poll definition specify the appropriate unit of measure. For example, in the case of memoryPoll the appropriate unit of measure is `bytes`.

- 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.

You can choose to store ping poll results together with the following two metrics:

- round trip time (RTT)
- Packet loss

The Ping result is always stored if storage option is set. The other two metrics are optional. The RTT is the milliseconds between when ping sent out and response was processed, or -1 if no response received. The packet loss is a percentage of packets loss, this is determined by sending multiple ping requests and tallying up the lost packets. The actual Ping result is 0 if ping fail or 100 if ping success.

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.

Link state polling

Link state polling monitors changes to the status of the following interface MIB variables: ifOperStatus and ifAdminStatus.

If the value of one of these MIB variables changes between poll intervals, an event is raised.

The initial state of link state polls is determined by any existing link state events on the interface. If there are no existing link state events, then the initial state is set to Clear. If many interfaces in your network do not have their administrative status set, you might want to determine the initial interface state by polling the interface instead. To set the initial state of the interface using a poll, set the `UseFirstPollForInitialState` configuration option in the `NcPollerSchema.cfg` configuration file.

When storing SNMP Link State data in the NCPOLLDATA polling database the data is not stored every poll cycle. The data is stored either upon the detection of a state change, or after 15 minutes, whichever happens first. If a state change occurs then the previous and current values are stored. If there is no state change then the current value is stored.

The interface can be in one of the following states:

- 3: available
- 2: administrative interface is down
- 1: unavailable

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. Additionally, an event is generated when a poll fails to return any data, and an event with a clear severity is generated when a poll to the same device subsequently succeeds.

Table 85. 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

Table 85. Events generated by SNMP link state polling (continued)			
Status of the ifAdminStatus MIB variable between poll intervals	**Status of the ifOperStatus MIB variable between poll intervals**	**Event generated**	**Event Severity**
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 hasn't.`	Minor

Related tasks

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.

Configuring Link State polling
You can specify how the ncp_poller process determines the initial state for Link State polls when there is no existing event.

Remote ping polling

During remote ping polling, enterprise-specific device MIBs are used to verify the status of the path between devices. In Multi Protocol Label Switching (MPLS) networks, remote ping polling can be useful on the edge of the network, where automatic MPLS rerouting is less likely to occur. 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 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 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 Administration 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 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:

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.

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 trigger 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 86. Default report to data label mapping

Report	Data label
Router Health Summary	memoryPercentageUsage
	cpuBusy
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 use 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 20. Enabling and disabling polls

To activate Network Manager polling, you must enable the poll policies.

Before you begin

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.

About this task

By default, only the chassis ping poll is enabled. You must enable other poll policies that you want to use.

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 a poll policy cause the Polling engine, `ncp_poller`, to restart the poll policy, 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:

Procedure

1. Click the **Administration** icon and select **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, 2021

Chapter 21. 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.

About this task

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 a poll policy cause the Polling engine, `ncp_poller`, to restart the poll policy, 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
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

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

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*.

Creating adaptive polls
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.

About this task

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.

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.

Procedure

1. Click the **Administration** icon and select **Network** > **Network Polling**.
2. Create a new policy by doing one of the following:

© Copyright IBM Corp. 2006, 2021

- 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. Ensure that you have specified at least one poll definition for the policy before enabling it.

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 highest 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 87. 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
Select the poller on which to run the poll policy.

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

Related tasks

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.

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.

About this task

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**.

Procedure

1. Click the **Administration** icon and select **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.

   Assign to Poller Instance
   Select the poller on which to run the poll policy.

   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 **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

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

Default poll policies

Network Manager provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Default ping policies

Network Manager provides default poll policies for ping operations.

The following table provides information on the default ping poll policies.

Table 88. 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. The managed status can be changed by using the Topoviz or the Structure Browser GUIs, or by using the `ManagedNode.pl`, `UnManagedNode.pl`, or `RemoveNode.pl` scripts.
End Node Ping	Uses the End Node Ping poll definition to perform ping operations on all end nodes, as defined by the class settings.

Table 88. Default ping poll policies (continued)	
Poll Policy Name	**Description**
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 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.

Table 89. 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 90. Basic SNMP threshold poll policies	
Name	**Description**
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 90. 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.
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.
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.
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.
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.
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.

Table 90. Basic SNMP threshold poll policies (continued)	
Name	**Description**
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.
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.

Foundry SNMP threshold poll policies

The following table describes the SNMP threshold policies that are supplied for Foundry devices.

Table 91. 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 92. 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.

Table 92. SNMP threshold poll policies for Cisco devices (continued)	
Name	**Description**
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.

Chapter 22. Creating new poll definitions

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

Before you begin

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.

About this task

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
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 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.

Related reference

Default poll definitions
Network Manager 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 begin

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.

About this task

To create a basic threshold poll definition:

Procedure

1. Click the **Administration** icon and select **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.

© Copyright IBM Corp. 2006, 2021

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.

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 User 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.

Data Units
Specify the data units for this poll definition. The appropriate data unit varies depending on the type of data in the poll definition. Here are some typical data types and the data units that correspond to those data types, together with examples from the default Network Manager poll definitions:

Counts
Poll definitions where the value is a count of some item. Examples include:

- dot3StatsAlignmentErrors
- ifInDiscards
- ifInErrors

For this type of poll definition specify a data unit of #.

Percentages
Poll definitions where the value is a percentage. Examples include:

- cpuBusyPoll
- ciscoCPUTotal5min

For this type of poll definition specify a data unit of %.

Specific units of measure
Poll definitions where the value is a specify unit of measure. Examples include:

- memoryPoll

For this type of poll definition specify the appropriate unit of measure. For example, in the case of memoryPoll the appropriate unit of measure is bytes.

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.

Note: When polling for interface data (not ping polling or remote ping polling), by default, all interfaces in the SNMP interfaces table of the device are polled, whether they were discovered or not. Interfaces might not be discovered if you configured interface filtering for discovery, or for some other reason, for example that they were inaccessible at discovery time. Undiscovered interfaces are still polled unless you configure a filter on the interface records in the NCIM database for the poll. If you add any interface filter to this poll, the filter is applied to the interface records in the NCIM topology database, and only those interfaces are polled. Only the subset of the discovered interfaces that also matches the filter is polled.

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 **Event Viewer** when an 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 6a to 6c for the **Clear Threshold** area.
9. Click **Save**.

Related concepts

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 use multibyte characters before you configure basic or generic threshold poll definitions.

Poll policy scope

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

Related reference

Example basic threshold expression

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

Creating generic threshold poll definitions

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

About this task

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

To create a generic threshold poll definition:

Procedure

1. Click the **Administration** icon and select **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.

   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 User 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.

   Note: When polling for interface data (not ping polling or remote ping polling), by default, all interfaces in the SNMP interfaces table of the device are polled, whether they were discovered or not. Interfaces might not be discovered if you configured interface filtering for discovery, or for some other reason, for example that they were inaccessible at discovery time. Undiscovered interfaces are still

polled unless you configure a filter on the interface records in the NCIM database for the poll. If you add any interface filter to this poll, the filter is applied to the interface records in the NCIM topology database, and only those interfaces are polled. Only the subset of the discovered interfaces that also matches the filter is polled.

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 **Event Viewer** 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 **Event Viewer** 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**.

What to do next

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

Related concepts

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 use multibyte characters before you configure basic or generic threshold poll definitions.

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

Related reference

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

Creating chassis and interface ping poll definitions

Use the **Poll Definition Editor** to create chassis and interface ping poll definition types. The results of a ping poll are stored as 0 if the ping operation fails or 100 if the ping operation succeeds.

About this task

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:

Procedure

1. Click the **Administration** icon and select **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.

   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 User 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.

   Note: When polling for interface data (not ping polling or remote ping polling), by default, all interfaces in the SNMP interfaces table of the device are polled, whether they were discovered or not. Interfaces might not be discovered if you configured interface filtering for discovery, or for some other reason, for example that they were inaccessible at discovery time. Undiscovered interfaces are still polled unless you configure a filter on the interface records in the NCIM database for the poll. If you add any interface filter to this poll, the filter is applied to the interface records in the NCIM topology database, and only those interfaces are polled. Only the subset of the discovered interfaces that also matches the filter is polled.
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. Response time is stored as the time in milliseconds between when the ping request was sent out and when the response was processed. If no response is received then the value of -1 is stored.

Packet Loss
Check the box to collect data about lost packets. Packet loss is stored as the percentage of packets lost, which is in turn determined by sending multiple ping requests and calculating the percentage of 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
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.

About this task

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:

Procedure

1. Click the **Administration** icon and select **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.

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 User 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.

   Note: When polling for interface data (not ping polling or remote ping polling), by default, all interfaces in the SNMP interfaces table of the device are polled, whether they were discovered or not. Interfaces might not be discovered if you configured interface filtering for discovery, or for some other reason, for example that they were inaccessible at discovery time. Undiscovered interfaces are still polled unless you configure a filter on the interface records in the NCIM database for the poll. If you add any interface filter to this poll, the filter is applied to the interface records in the NCIM topology database, and only those interfaces are polled. Only the subset of the discovered interfaces that also matches the filter is polled.

8. Click **Save**, then click **OK**.

Related concepts

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

Default poll definitions

Network Manager 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 provides.

Note: Basic threshold poll definitions require data unit settings. The table includes the default data unit settings for each basic threshold poll definition.

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.

Data units: #

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.

Data units: %

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

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.

Data units: #

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.

dot3StatsAlignmentErrors

Poll definition type: Basic threshold.

Data units: #

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.

Data units: #

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.

Data units: #

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

HighDiscardRate

Poll definition type: Basic threshold.

Data units: #

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

ifInDiscards

Poll definition type: Basic threshold.

Data units: #

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.

Data units: #

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.

Data units: #

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.

Data units: #

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.

Data units: #

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.

Data units: Bytes

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

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.

Data units: %

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.

Data units: %

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.

Data units: Time, milliseconds

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.

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 **Event Viewer** 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 **Event Viewer** should now read as follows:

   ```
   CPU usage high (avgBusy5=eval(text,"SNMP.VALUE.avgBusy5")>=eval
   (text,"SNMP.VALUE.avgBusy6"))
   ```

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 **Event Viewer** 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 **Event Viewer** should now read as follows:

```
CPU usage high (avgBusy5=eval(text,"SNMP.VALUE.avgBusy5")<=80)
```

Chapter 23. 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

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

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

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.

Related reference

Default poll policies

Network Manager 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 begin

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 a poll policy cause the Polling engine, `ncp_poller`, to restart the poll policy, 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.

About this task

To change a poll policy:

Procedure

1. Click the **Administration** icon and select **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. Ensure that you have specified at least one poll definition for the policy before enabling it.

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

© Copyright IBM Corp. 2006, 2021

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 highest 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 93. 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
Select the poller on which to run the poll policy.

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

Default poll policies

Network Manager 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 a poll policy, for example, the ciscoMemoryPctgUsage 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

Default poll policies

Network Manager 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 begin

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

Creating new poll definitions

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

Related reference

Default poll policies

Network Manager 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.

About this task

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:

Procedure

1. Click the **Administration** icon and select **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.

   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 User 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.

   Data Units
   Specify the data units for this poll definition. The appropriate data unit varies depending on the type of data in the poll definition. Here are some typical data types and the data units that correspond to those data types, together with examples from the default Network Manager poll definitions:

   Counts
   Poll definitions where the value is a count of some item. Examples include:

   - dot3StatsAlignmentErrors
   - ifInDiscards
   - ifInErrors

   For this type of poll definition specify a data unit of #.

   Percentages
   Poll definitions where the value is a percentage. Examples include:

   - cpuBusyPoll
   - ciscoCPUTotal5min

   For this type of poll definition specify a data unit of %.

   Specific units of measure
   Poll definitions where the value is a specify unit of measure. Examples include:

   - memoryPoll

For this type of poll definition specify the appropriate unit of measure. For example, in the case of memoryPoll the appropriate unit of measure is `bytes`.

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.

   Note: When polling for interface data (not ping polling or remote ping polling), by default, all interfaces in the SNMP interfaces table of the device are polled, whether they were discovered or not. Interfaces might not be discovered if you configured interface filtering for discovery, or for some other reason, for example that they were inaccessible at discovery time. Undiscovered interfaces are still polled unless you configure a filter on the interface records in the NCIM database for the poll. If you add any interface filter to this poll, the filter is applied to the interface records in the NCIM topology database, and only those interfaces are polled. Only the subset of the discovered interfaces that also matches the filter is polled.

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 **Event Viewer** when an 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 6a to 6c for the **Clear Threshold** area.

8. Click **Save**.

Related concepts

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 use multibyte characters before you configure basic or generic threshold poll definitions.

Poll policy scope

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

Related reference

Example basic threshold expression

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

Changing generic threshold poll definitions

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

About this task

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:

Procedure

1. Click the **Administration** icon and select **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.

   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 User 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.

Note: When polling for interface data (not ping polling or remote ping polling), by default, all interfaces in the SNMP interfaces table of the device are polled, whether they were discovered or not. Interfaces might not be discovered if you configured interface filtering for discovery, or for some other reason, for example that they were inaccessible at discovery time. Undiscovered interfaces are still polled unless you configure a filter on the interface records in the NCIM database for the poll. If you add any interface filter to this poll, the filter is applied to the interface records in the NCIM topology database, and only those interfaces are polled. Only the subset of the discovered interfaces that also matches the filter is polled.

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 **Event Viewer** 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 **Event Viewer** 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
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 use multibyte characters before you configure basic or generic threshold poll definitions.

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

Related reference

Default poll policies
Network Manager provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Example generic threshold expression

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

Changing chassis and interface ping poll definitions

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

About this task

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:

Procedure

1. Click the **Administration** icon and select **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.

   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 User 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.

   Note: When polling for interface data (not ping polling or remote ping polling), by default, all interfaces in the SNMP interfaces table of the device are polled, whether they were discovered or not. Interfaces might not be discovered if you configured interface filtering for discovery, or for some other reason, for example that they were inaccessible at discovery time. Undiscovered interfaces are still polled unless you configure a filter on the interface records in the NCIM database for the poll. If you add any

interface filter to this poll, the filter is applied to the interface records in the NCIM topology database, and only those interfaces are polled. Only the subset of the discovered interfaces that also matches the filter is polled.

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. Response time is stored as the time in milliseconds between when the ping request was sent out and when the response was processed. If no response is received then the value of -1 is stored.

   Packet Loss
   Check the box to collect data about lost packets. Packet loss is stored as the percentage of packets lost, which is in turn determined by sending multiple ping requests and calculating the percentage of 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
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.

About this task

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:

Procedure

1. Click the **Administration** icon and select **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.

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 User 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.

   Note: When polling for interface data (not ping polling or remote ping polling), by default, all interfaces in the SNMP interfaces table of the device are polled, whether they were discovered or not. Interfaces might not be discovered if you configured interface filtering for discovery, or for some other reason, for example that they were inaccessible at discovery time. Undiscovered interfaces are still polled unless you configure a filter on the interface records in the NCIM database for the poll. If you add any interface filter to this poll, the filter is applied to the interface records in the NCIM topology database, and only those interfaces are polled. Only the subset of the discovered interfaces that also matches the filter is polled.

6. Click **Save**, then click **OK**.

Related concepts

Link state polling
Link state polling monitors changes to the status of the following interface MIB variables: ifOperStatus and ifAdminStatus.

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

Related tasks

Configuring Link State polling
You can specify how the ncp_poller process determines the initial state for Link State polls when there is no existing event.

Related reference

Default poll policies

Network Manager 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 70%.

Required settings

To create a poll definition that responds to the requirements described in "Scenario" on page 491, 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          <          70
```

  - 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
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 use 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
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 use multibyte characters before you configure basic or generic threshold poll definitions.

Related tasks

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

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..

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

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 use multibyte characters before you configure basic or generic threshold poll definitions.

Related tasks

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

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

Chapter 24. Deleting poll policies

Delete poll policies when they are no longer required.

About this task

To delete a poll policy:

Procedure

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

What to do next

The selected poll policies are deleted.

© Copyright IBM Corp. 2006, 2021

Chapter 25. Deleting poll definitions

Delete poll definitions when they are no longer required.

About this task

To delete a poll definition:

Procedure

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

What to do next

The selected poll definitions are deleted.

© Copyright IBM Corp. 2006, 2021

Chapter 26. 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

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.

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 **Event Viewer**. 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 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.

© Copyright IBM Corp. 2006, 2021

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 User 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 User 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 **Event Viewer**. 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 **Event Viewer** now contains an actionable event, that is, an NmosPingFail event with a tally of 18 or greater.

Related tasks

Enabling and disabling polls
To activate Network Manager polling, you must enable the poll policies.

Changing poll policies
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

Default poll policies
Network Manager provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Default poll definitions
Network Manager 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 **Event Viewer**. 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 User 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 **Event Viewer**. 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 **Event Viewer** with a continually increasing Tally value. The **Event Viewer** now contains an actionable POLL-HighDiscardRate event.

Related tasks

Enabling and disabling polls
To activate Network Manager polling, you must enable the poll policies.

Changing poll policies
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

Default poll policies
Network Manager provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Default poll definitions
Network Manager 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 you begin

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.

About this task

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.

Procedure	Confirm device down	Confirm threshold violation
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 User 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

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

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

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.

Chapter 27. 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.

About this task

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*.

Related tasks

Administering Network Manager
To administer the product, you start and stop it, and administer processes, logs, ports, users, passwords, reports, and databases.

Administering polls

Use the command-line interface to administer poll policies.

Speeding up ncp_poller startup by not checking SNMP credentials

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

About this task

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.

Procedure

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.

About this task

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*.

Procedure

1. Change to the `$NCHOME/precision/scripts/perl/scripts` directory and locate the **itnm_poller.pl** script.

© Copyright IBM Corp. 2006, 2021

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
   ```

   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.

About this task

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*.

Procedure

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.

About this task

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*.

Procedure

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.

Before you begin

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.

Procedure

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 Administration 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 `PopulateDNCIM_ManagedStatus.stch` file and the NCIM database tables.

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

Configuring how many events the poller reads

Large increases in the number of network events can slow down the polling process, ncp_poller. To minimize the performance impact of sudden increases in events, you can define how many network events the poller reads.

About this task

The polling process, ncp_poller, reads events from the NCMONITOR.activeEvent table into memory at intervals in order 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.

To configure a limit to the number of events that the poller reads, complete the following steps:

Procedure

1. Back up and edit the following file: `NcPollerSchema.cfg`.
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 will read. 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.

Changing the interval for checking poll policy membership size

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

About this task

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.

Procedure

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.

Procedure

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.

Adjusting the size of poll data queues

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 **Event Viewer**. Then, the poller reduces the queue size by discarding the queued data. The discarded data is written to another file.

Before you begin

Set the poller debug level to 4.

Procedure

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 **Event Viewer** for messages and alerts.

Results

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

- In the **Event Viewer**, 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 for information only and cannot be inserted back into the polling database. For example:

  ```
  MONITOREDOBJECTID,5, MONITOREDINSTID,184, POLLTIME,1447960873, ERRORCODE,100,
  VALUE, 0
  MONITOREDOBJECTID,6, MONITOREDINSTID,184, POLLTIME,1447960873, ERRORCODE,100,
  VALUE, -1
  MONITOREDOBJECTID,5, MONITOREDINSTID,248, POLLTIME,1447960873, ERRORCODE,100,
  VALUE, 100
  ```

What to do next

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 **Event Viewer**.
- Clear the .polldata file. The file is not cleared or rotated automatically.

Related tasks

Changing the logging level for processes
Change the logging level of a process before you start the process or while the process is running.

Locating log files for a process
Locate log files for a process to obtain information that might be helpful in troubleshooting the process.

Setting up an additional poller
Set up an additional poller on the Network Manager server if the default pollers are not sufficient to handle your network load.

Monitoring poller capacity

Speeding up ncp_poller startup by not checking SNMP credentials

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

About this task

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.

Procedure

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.

Configuring Link State polling

You can specify how the ncp_poller process determines the initial state for Link State polls when there is no existing event.

About this task

The ncp_poller process can either use the first poll to determine the initial state of Link State polls, or assume a Clear state.

Procedure

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

Link state polling monitors changes to the status of the following interface MIB variables: ifOperStatus and ifAdminStatus.

Related tasks

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.

Administering multiple pollers

If additional pollers are needed to poll your network, you can set up new pollers. 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 additional 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 updating caches, partition dropping, and **SNMP MIB Grapher** support. By default, the `ncp_poller_admin` poller is started with the `-admin` option, and performs the administrative tasks. The `ncp_poller_default` poller is started with the `-noadmin` option, and is used for SNMP and Ping polling. If you have many devices to poll on a regular basis, consider using multiple pollers for polling. One poller must always be started with the `-admin` option to perform necessary administrative functions.

Note: 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.

If you are using failover, replicate the same arrangement of pollers on the backup server as you have on the primary server.

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

Restriction: For any given domain, in a distributed environment, all pollers must be started on the same server as the discovery engine.

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

For additional information, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Monitor Policy editor

The administrator controls which poller is used by a policy. By default, each policy uses the `ncp_poller_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. By default, the `ncp_poller_admin` poller performs this function.

Related information

Best Practices Guide for remote discovery on the Network Manager Best Practices pageFor more information on distributed architecture, see the Best Best Practices Guide for remote discovery on the Network Manager Best Practices page.

Setting up an additional poller

Set up an additional poller on the Network Manager server if the default pollers are not sufficient to handle your network load.

About this task

The default pollers are automatically installed when you install Network Manager. By default, the `ncp_poller_admin` poller is started with the `-admin` option, and performs the administrative tasks. The

`ncp_poller_default` poller is started with the `-noadmin` option, and is used for SNMP and Ping polling.

Complete the following steps to register a new instance of the poller and automatically start the poller.

Procedure

1. Edit the `CtrlServices.cfg` file.
   a) Locate the entry for the **`ncp_poller`** process.
   b) Copy the entry and change the serviceName setting in this new ncp_poller entry to match the name you used to register the poller. Alternatively, edit the commented-out example of an additional poller in the default version of the `CtrlServices.cfg` file and set the serviceName setting to be a unique name for the poller.

      Important: Ensure that only one of the pollers is started using the `-admin` option, which configures the poller to perform essential administrative functions. By default this is the `ncp_poller_admin` poller. Start all other pollers using the -noadmin option, which configures the pollers to not perform administrative functions. Refer to the command line options for ncp_poller for information on other options that you can use when starting the pollers.
   c) Save and close the file.
2. Restart the ncp_ctrl process with the specified domain.

   The **`ncp_ctrl`** process restarts all running **`ncp_`** processes, including the new poller. The poller automatically registers any new named pollers when it starts.

Example

The example below creates a poller, in the NCOMS domain, that you might use for certain ping polls, and another poller that you might use for certain SNMP queries.

1. In your `CtrlServices.cfg` file, leave the ncp_poller entry with the serviceName of ncp_poller_default as it is. This poller performs the administrative functions and is used for MIB graphing.
2. Uncomment the ncp_poller_ping entry. This insert should now look like this:

```
insert into services.inTray
(
    serviceName,
    binaryName,
    servicePath,
    domainName,
    argList,
    dependsOn,
    retryCount
)
values
(
    "ncp_poller_ping",
    "ncp_poller",
    "$PRECISION_HOME/platform/$PLATFORM/bin",
    "$PRECISION_DOMAIN",
    [ "-domain" , "$PRECISION_DOMAIN" , "-latency" ,  "200000", "-debug",  "0",
 "-messagelevel", "warn", "-name", "ncp_poller_ping", "-noadmin" ],
    [ "nco_p_ncpmonitor", "ncp_g_event" ],
    5
);
```

3. Add a similar entry for the new snmpPoller poller. This insert is exactly the same as the PingPoller insert, except for the serviceName setting, the -name option in the argList parameter.

```
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
);
```

What to do next

After you set up your pollers, complete these steps.

1. Restart Network Manager, and then log into Network Manager.
2. Click the **Administration** icon and select **Network** > **Network Polling** to edit the poll definitions.
3. Under **Poll Policy Properties**, choose **Assign to poller instance**.
4. Assign a poll to the existing poller and then assign a different poll to the new poller.

Assigning a poller for MIB graphing

If you have want to change the poller assigned to handle MIB graphing, edit the properties file

Before you begin

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

About this task

To assign a poller for MIB graphing:

Procedure

1. Open the following file: `$NMGUI_HOME/profile/etc/tnm/tnm.properties`.
2. Change the value of the **`tnm.graph.poller`** parameter from the default `ncp_poller_admin` 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.

About this task

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.

Procedure

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 poll data

Network Manager processes raw poll data into historical poll data and stores up a to full year of aggregated historical poll data.

About this task

Network Manager uses the Apache Storm realtime computation system to aggregate raw poll data into historical poll data, and stores raw and historical poll data in the NCPOLLDATA database.

Raw and historical poll data are presented in graphs and tables in the **Network Health Dashboard** and in performance reports.

You can also access this data on a per device or link basis from the topology map within the **Network Hop View** GUI, **Network Views** GUI, and **Path Views GUI**.

About poll data aggregation

Data aggregation is the process whereby raw polling data is aggregated into historical poll data and and stored in the database, from where it can be retrieved for presentation in graphs, reports, and dashboards. This is a fully automated system that begins processing raw data when it starts up.

In earlier releases aggregation of polling data for storage as historical data was performed by Tivoli Data Warehouse. From version 4.2 onwards, aggregation of polling data is performed entirely within Network Manager

Note: There is no migration path for historical polling data stored in a legacy Tivoli Data Warehouse integration.

Once set up, the complete system is maintenance free. There are mechanisms built in that protect the system if it starts to receive more data than it can handle, and metrics that make it easy to monitor during the planning phase.

Users can perform a very limited set of configuration tasks, such as defining whether to aggregate data for specific time periods. For example, you can switch off data aggregation for the annual data period.

For many customers, especially those with large networks, the rate at which poll data can be collected can be high. It is important that the mechanism used to perform the task of summarization be able to handle the data load without causing any disruption to the poller. Apache Storm is a distributed real-time computation system that can perform processing on live streams of data efficiently. Apache Storm uses a topology that defines the tasks of aggregating the poll data, and Storm then manages the tasks and the data load. The topology defines how raw data is processed to produce aggregated daily, weekly, and monthly, and annual polling data that can be stored in the NCPOLLDATA database. By default, this topology will take the name NMStormTopology.

Poll data aggregation

The historical poll data processing and storage system is designed to accept data from all instances of the Polling engine, ncp_poller across all domains. Within these ncp_poller instances, only data from poll policies that are configured to store data will be aggregated into historical data.

The system automatically aggregates data into four collections, each representing a different time period:

- Last day
- Last week
- Last month
- Last year

Each of these collections, including an additional collection that receives the raw data, is automatically purged as data ages out of its time period. The raw data is collected in a database table that maintains the last hour's worth of data.

Raw polling data, and the results of the data aggregation for the four time periods, is stored in the NCPOLLDATA database. All of the Apache Storm processing required to produce this aggregated data is set up by default. This enables Network Manager to provide a view of polling data going back a full year.

Raw polling data, and the results of the data aggregation for the four time periods, are stored in the following database tables within the NCPOLLDATA database.

- Raw polling data is stored in the pollData table.
- There are four summary data tables, one for each time period. In order to populate these tables with historical poll data, Apache Storm continuously reads data from the raw poll data table, pollData, and calculates aggregated poll data based on a separate default intervals for each time period using an exponential weighted moving average (EWMA) of the raw poll data. Storm then writes this average data to the relevant aggregated poll data table. For example, for the last day time period Storm calculates an average of the raw poll data every 15 minutes and writes this average value to the pdEwmaForDay table. The following NCPOLLDATA database tables store historical poll data.

  Note: Aggregated data is stored in the NCPOLLDATA database without regard to domain.

  pdEwmaForDay table
  Stores the aggregated last day of poll data with a 15 minute moving average.

  pdEwmaForWeek table
  Stores the aggregated last week of poll data with a two hour moving average.

  pdEwmaForMonth table
  Stores the aggregated last month of poll data with an 8 hour moving average.

  pdEwmaForYear table.
  Stores the aggregated last year of poll data with a 4 day moving average.

Poll data purging

Purging of the raw and historical poll data tables is handled automatically, making use of table partitioning for efficiency.

Each historical poll data table is time bound and the table is partitioned into time slots. For example, the pdEwmaForDay table, that stores historical poll data for the last day, holds 25 hours worth of data, broken up into 25 partitions of 1 hour each.

Inserts to the tables are made in the newest partition and the oldest partition is purged. This approach reduces conflicts between insert and delete operations, thereby improving purging and insertion efficiency. The automatic purging is based on an age factor, which is appropriate for each table. For instance in the forDay table, partitions are dropped as the data ages beyond 24 hours. This purging approach ensures that, once the total number of polled entities and polled KPIs is fixed, then the size of each historical poll data table remains stable and within the recommended bounds.

Note: All SQL queries to these tables can be made without reference to, or knowledge of, the partitions.

Capacity guidelines for historical data polling

Use this information to help calculate polling load and determine whether the polling load is within the recommended limits for a stable and well performing polling system.

Sizing considerations

The polling load on a server can be expressed by calculating the overall polling insert rate to the NCPOLLDATA database. The higher the insert rate, the larger the capacity needed within the raw and historical poll data storage tables to store the poll data. Once the historical poll data tables increase beyond a certain size, the performance of queries to the tables by dashboards and reports is likely to degrade; therefore both the insert rate to the POLLDATA raw poll data table, and the total number of rows in the historical poll data tables must not be increased beyond a recommended limits in order to ensure system stability.

It is recommended that the polling insert rate to the POLLDATA raw data table should not exceed 20 million inserts per hour, and the total number of rows for the aggregate data tables should not exceed 200 million rows. This ensures that the size of each of the historical polling data tables stays within acceptable limits.

Sizing tables

Use these tables to help determine the appropriate polling load for your system, in particular, how many entities and KPIs you should be polling and what polling interval to set for your poll policies.

Use the following table to determine your approximate polling insert rate and the number of rows this insert rate will generate in each of the historical poll data tables. Use these tables by reviewing the following columns and finding values that most closely match the settings in your network:

In each of the examples presented in the table, the number of KPIs (poll definitions) per poll policy and the poll interval for each poll policy are assumed to be the following:

Table 94. Number of KPIs and poll interval per poll policy

	Number of KPIs	**Poll interval in minutes**	**Number of polls per hour**
Chassis polls	3	5	36
Interface polls	5	5	60

The following examples show how the values for insert rate and number of rows in the historical poll data tables change as you vary the number of polled entities in your network.

Example that falls within the recommended limits

The following examples fall within the recommended limits for a stable polling system.

Number of polled entities

- Chassis devices: 15,000
- Interfaces: 300,000

Table 95. Insert rate and number of rows in millions per historical poll data table

	Insert rate [Inserts/hour]	**Last day [Millions of rows]**	**Last week [Millions of rows]**	**Last month [Millions of rows]**	**Last year [Millions of rows]**
Chassis polls	0.72	5.76	5.04	5.4	5.475

Table 95. Insert rate and number of rows in millions per historical poll data table (continued)					
	Insert rate [Inserts/hour]	**Last day [Millions of rows]**	**Last week [Millions of rows]**	**Last month [Millions of rows]**	**Last year [Millions of rows]**
Interface polls	18.56	153.6	134.4	144	146
Total	19.28	159.36	139.44	149.4	151.475

Example that falls outside of the recommended limits

The following examples fall outside of the recommended limits for a stable polling system.

Number of polled entities

- Chassis devices: 30,000
- Interfaces: 500,000

Table 96. Insert rate and number of rows in millions per historical poll data table					
	Insert rate [Inserts/hour]	**Last day [Millions of rows]**	**Last week [Millions of rows]**	**Last month [Millions of rows]**	**Last year [Millions of rows]**
Chassis polls	1,080,000	8.64	30.24	32.4	32.85
Interface polls	30,000,000	240	840	900	912.5
Total	31,080,000	248.64	870.24	932.4	945.35

Try it for yourself

Use the spreadsheet calculator available on SMC to estimate these values:

- Estimated number of polled entities in your network
- Average number of KPIs and poll interval across your poll policies

The spreadsheet calculates the insert rate and number of rows in the historical poll data tables for your network based on your estimated values.

Related information

Network Manager Calculator for system sizing and polled data storage On this page you can find the spreadsheet calculator that calculates insert rate and number of rows in the historical poll data tables based on estimated values.

System tolerance limits

The primary sources of potential disruption to the system are loss of connection to the NCPOLLDATA database, and the Apache Storm process going down. The system is tolerant to each of these eventualities and if all connections and process are restored within certain times, no data will be lost.

Apache Storm is down

The following sections describe system tolerance in more detail:

If Apache Storm goes down and stays down for less than one hour then there is no impact on raw poll data collection and storage in the pollData table. On restart, Storm will automatically continue to process the pollData rows from where it left off, ensuring that all raw poll data within that hour is successfully processed into historical poll data. Note that during the time that Apache Storm is down, no data will be added to the historical poll data tables.

If Apache Storm goes down and stays down for more than one hour then data from the pollData table begins to age out and will not be processed to the historical poll data tables.

NCPOLLDATA database is down

The following sections describe system tolerance in more detail:

If the NCPOLLDATA database is down for less than about 5 mins, then all instances of ncp_poller retain data in a buffer, and once the database connection is made again, the data in the buffer is automatically written to the pollData table.

If the NCPOLLDATA database is down for more than about 5 minutes, then at some point, the in-memory buffer is exhausted and oldest data is lost until the connection to the database resumes.

Starting and stopping Apache Storm

You can start, stop, and retrieve status on the Apache Storm system using the itnm_start, itnm_stop, and itnm_status commands.

About this task

For more information on these commands, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Configuring poll data aggregation

All of the Apache Storm processing required to produce this historical poll data is set up by default. You can, however, perform a limited set of configuration tasks.

Specifying which historical poll data periods to store

By default, Network Manager stores historical poll data in daily, weekly, monthly, and annual time periods. You can turn off storage of historical poll data for any of these periods.

About this task

To turn off storage of daily, weekly, monthly, or annual historical poll data, perform the following steps.

Procedure

1. Edit the properties file for the Storm topology using a text editor such as vi.

   By default, the Apache Storm topology takes the name `NMStormTopology` and uses a properties file named `NMStormTopology.properties`.

   ```
   vi $NCHOME/precision/storm/conf/NMStormTopology.properties
   ```

2. Find one of the following properties in the file, and set it to 0, to turn off storage of historical poll data for the corresponding time period.

   ewma.day.storage.enabled
   Set to false to switch off storage of daily historical poll data

   ewma.week.storage.enabled
   Set to false to switch off storage of weekly historical poll data

   ewma.month.storage.enabled
   Set to false to switch off storage of monthly historical poll data

   ewma.year.storage.enabled
   Set to false to switch off storage of yearly historical poll data

3. Save the file.
4. Restart the Apache Storm topology.

Other historical poll data configuration options

The `NMStormTopology.properties` configuration file also contains the following parameters. It is strongly recommended to leave these values at the default settings.

Aggregation periods

ewma.day.window.period.minutes
The window, in minutes, over which input raw poll data is aggregated for a single data point in the aggregate table for the day. A data point is generated only if new data was available within each window, so there is no gain having this lower than the smallest poller poll interval. The allowed range is 1 to 60 minutes. The default value is 15 minutes

ewma.week.window.period.minutes
The window, in minutes, over which input raw poll data is aggregated for a single data point in the aggregate table for the week. A data point is generated only if new data was available within each window. The allowed range is 1 to 300 minutes. The default value is 120 minutes.

ewma.month.window.period.hours
The window, in hours, over which input raw poll data is aggregated for a single data point in the aggregate table for the month. A data point is generated only if new data was available within each window. The allowed range is 1 to 24 hours. The default value is 8 hours.

ewma.year.window.period.days
The window, in days, over which input raw poll data is aggregated for a single data point in the aggregate table for the year. A data point is generated only if new data was available within each window. The allowed range is 1 to 30 days. The default value is one day.

Database access

nm.domain
The Apache Storm server uses this parameter to find credentials to enable it to access the NCPOLLDATA database. Storm looks for the domain-specific `DbLogins.`*`domain_name`*`.cfg` and `tnm.`*`domain_name`*`.properties` files.The `nm.domain` parameter is set at installation time. Storm will fall back to the generic versions of these files (`DbLogins.cfg` and `tnm.properties`) if no domain-specific versions of the files are found

Raw data table read interval

ncpolldata.read.interval.seconds

The interval at which the poll data table is queried The allowed range is 5 seconds to 3000 seconds. The default value is 60 seconds.

Number of JDBC connections

ewma.day.jdbc.connection.count
ewma.week.jdbc.connection.count
ewma.month.jdbc.connection.count
ewma.hour.jdbc.connection.count

The number of JDBC connections for inserting data to the various aggregate tables. The range allowed is 1 to 10.

Monitoring poller capacity

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 97. 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.	%
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 insertion rate to the raw poll data table ncpolldata.pollData, expressed as the number of records inserted during one hour. This metric is useful only if historical polling is used.	Count

Before you begin

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.

Procedure

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
```

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

Results

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.

```
Health (%) for Policy 'Default Chassis Ping' PollDef My Poll Definition 1'
(Type:SNMP Link State)
  A value less than 100% indicates the policy is behind and some devices were not polled
during the last polling cycle

    100 ------------------+-------------------------------------------------------------++
                          |+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++||
                          ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
                          ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
                          ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
     50 ------------------||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
                          ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
                          ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
                          ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      0 -++++++++++++++++-------------------------------------------------------------
         ^         ^         ^         ^         ^         ^         ^         ^
       13:46     14:06     14:26     14:46     15:06     15:26     15:46     16:06
       Time Window (4 hours):  2013-12-08T13:46:25  to  2013-12-08T17:46:25
(Sample interval: 2 minutes)
```

What to do next

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

Table 98. 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 98. 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" on page 525
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 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 an additional poller
Set up an additional poller on the Network Manager server if the default pollers are not sufficient to handle your network load.

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

Adjusting the size of poll data queues

Chapter 28. Troubleshooting network polling

Use this information to understand how to troubleshoot network polling.

Troubleshooting ping polling of the network

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.

Before you begin

By default, the tables and views defined in the `$NCHOME/precision/scripts/sql/database_type/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 domain. After 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.

About this task

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.

Procedure

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.

   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.

© Copyright IBM Corp. 2006, 2021

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 tasks

Troubleshooting Network Manager
Consult these troubleshooting notes to help determine the cause of the problem and what to do about it.

Troubleshooting SNMP polling

There is a configuration setting in the poller called AggregationLimit. This setting applies only to SNMP poll policies that have interface filtering enabled. The AggregationLimit setting determines how the `ncp_poller` process breaks up polls containing multiple SNMP GET requests. You might need to modify the default setting depending on the data that you are polling in your network.

About this task

The `ncp_poller` process breaks up poll requests into SNMP Protocol Data Unit (PDU) packets down to the aggregation limit. For example, when the value of the aggregation limit.is set to the default value of 30, the `ncp_poller` process creates PDUs with no fewer than 30 individual SNMP GET requests. This default setting, `AggregationLimit = 30`, can generate errors in the following situations. In both cases, you should decrease the value of the `AggregationLimit` parameter.

- For large poll data the results retrieved by 30 SNMP GET requests can overload the results PDU. This situation can occur when issuing bandwidth poll requests.
- If one of the OIDs being requested within the PDU does not exist then the results PDU is returned with an error code and the entire poll is dropped. This issue can occur when dealing with sparse tables. When this happens the poll will get dropped if just one entry is missing.

Note: Decreasing the value of the `AggregationLimit` parameter has the following impact on system performance:

- Increase in the volume of SNMP traffic data transmitted over the network. For example; a decrease in the value of `AggregationLimit` from 30 to 1 results in an approximately fourfold increase in the volume of SNMP traffic data transmitted over the network.
- Increase in the CPU utilization for the ncp_poller process: For example; a decrease in the value of `AggregationLimit` from 30 to 1 results in an approximately 50% increase in CPU utilization for the ncp_poller process.

There is a proportionate impact for values between 30 and 1. For example, if the value of the `AggregationLimit` parameter is decreased from 30 to 15, then this change results in an approximately twofold increase in the volume of SNMP traffic data transmitted over the network. The corresponding increase in CPU utilization for the ncp_poller process would be on the order of 25%.

Following a change in the value of `AggregationLimit` you should monitor ncp_poller process performance by monitoring the ncp_poller metrics.

To change the default AggregationLimit setting, perform the following steps.

Procedure

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

   ```
   update config.properties set AggregationLimit = aggregationLimit;
   ```

   Where *aggregationLimit* is the value of the aggregation limit setting. By default this is 30.
3. Restart the `ncp_poller` process.

Related tasks

Monitoring poller capacity

Troubleshooting the processing of historical poll data

You can define thresholds related to the age of data in the historical poll data tables, and related to the size of the poll data tables. If these thresholds are violated, this is an indication that the polling load might be too great, or that there is an issue with the historical poll data system or with the polling database. Threshold violations cause log messgaers to be generated, and also generate Tivoli Netcool/OMNIbus ObjectServer alerts which can be viewed in the Tivoli Netcool/OMNIbus Web GUI **Event Viewer**.

About this task

A Tivoli Netcool/OMNIbus ObjectServer alert is generated if the following events occur. These alerts have an Alert Group setting of ITNM Status:

Maximum insertion rate to the raw poll data table is exceeded

The maximum insertion rate to the raw poll data table ncpolldata.pollData is set in the config database table `config.tableMonitor`. If this values is exceeded then an ITNM Status alert is raised in the Tivoli Netcool/OMNIbus Web GUI **Event Viewer**, and a log message is written to the `NCHOME/log/precision/ncp_poller.SnmpPoller.`*`poller_name.domain`* log file.

If this maximum insertion rate is exceeded, then this is an indication that the polling load is too great. You might need to adjust polling load by decreasing number of devices that you are polling, decreasing the number of metrics that you are collecting on these devices, or by changing your polling intervals.

Age counts are exceeded for the historical poll data tables

Maximum age settings for the historical poll data tables in the ncpolldata database are set in the config database table `config.tableMonitor`. If these values are exceeded then an ITNM Status alert is raised in the Tivoli Netcool/OMNIbus Web GUI **Event Viewer** and a log message is written to the `NCHOME/log/precision/ncp_poller.SnmpPoller.`*`poller_name.domain`* log file.

If any of these maximum age settings is exceeded, then this is an indication that the automatic data purging mechanism is not working. The Apache Storm system, which processes historical poll data,

might not be running or might not be keeping up with polling and purging. Alternatively, there might be a database server issue on the server that hosts the ncpolldata database.

No heartbeat received from Apache Storm

The system checks the timestamp within an ncpolldata database. If this timestamp is not updating, then this indicates that no heartbeat has been received from the Apache Storm system, which processes historical poll data. In this case an ITNM Status alert is raised in the Tivoli Netcool/OMNIbus Web GUI **Event Viewer**.

If the heartbeat is not being received fromApache Storm, then this could mean Apache Storm is not running or needs attention.

Batches processed by Apache Storm falling behind polling batches

The Polling engine ncp_poller collects raw poll data in batches and assigns a batch ID to each batch. When processing this raw poll data into historical poll data, Apache Storm marks each batch to indicate that it has been processed. The batch ID status of the latest poll data batches is checked regularly. If the timings associated with the batch identifier of the last set of raw poll show that the data processed by Apache Storm is falling behind the latest batches written by ncp_poller, then an ITNM Status alert is raised in the Tivoli Netcool/OMNIbus Web GUI **Event Viewer**.

If batch processing is falling behind, then this could mean Apache Storm is not running or needs attention.

Related tasks

Starting and stopping Apache Storm

You can start, stop, and retrieve status on the Apache Storm system using the itnm_start, itnm_stop, and itnm_status commands.

Related reference

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`.

Chapter 29. 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

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.

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 99. 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 533	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 539	Event Event state
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 536	Event with filtered fields Event state

© Copyright IBM Corp. 2006, 2021

Table 99. Quick reference for event enrichment (continued)

Action	Further information	Data passed to next step
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 542 "Precedence value" on page 566	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 568	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 551	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 536 "Outgoing Event Gateway queue" on page 537	Fields enriched by Event Gateway
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 560	Event with filtered fields and enriched fields Event state Event map name Precedence value PollerEntityId

Table 99. Quick reference for event enrichment (continued)		
Action	**Further information**	**Data passed to next step**
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.	"Root Cause Analysis (RCA) plugin" on page 565 "Event Gateway plugins" on page 588	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 537	To ObjectServer Fields enriched by plug-ins

Related concepts

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.

Related tasks

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.

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.

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 100. 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.

```
insert into config.nco2ncp
(
    EventFilter,
    StandbyEventFilter,
    FieldFilter
)
```

```
values
(
    "LocalNodeAlias <> '' and (NmosDomainName =
         '$DOMAIN_NAME' or NmosDomainName = '')",
    "EventId in ('ItnmHealthChk', 'ItnmDatabaseConnection')",
    [
        "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"
    ]
);
```

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 534 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 101. 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.

Table 101. Lines of code relevant to the standby filter (continued)	
Line numbers	**Description**
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`.

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 534 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 102. 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 103. 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.

```
insert into config.ncp2nco
(
    FieldFilter
)
values
(
    [
        "NmosCauseType",
        "NmosDomainName",
        "NmosEntityId",
        "NmosManagedStatus",
        "NmosObjInst",
        "NmosSerial"
    ]
);
```

Related tasks

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.

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.

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.

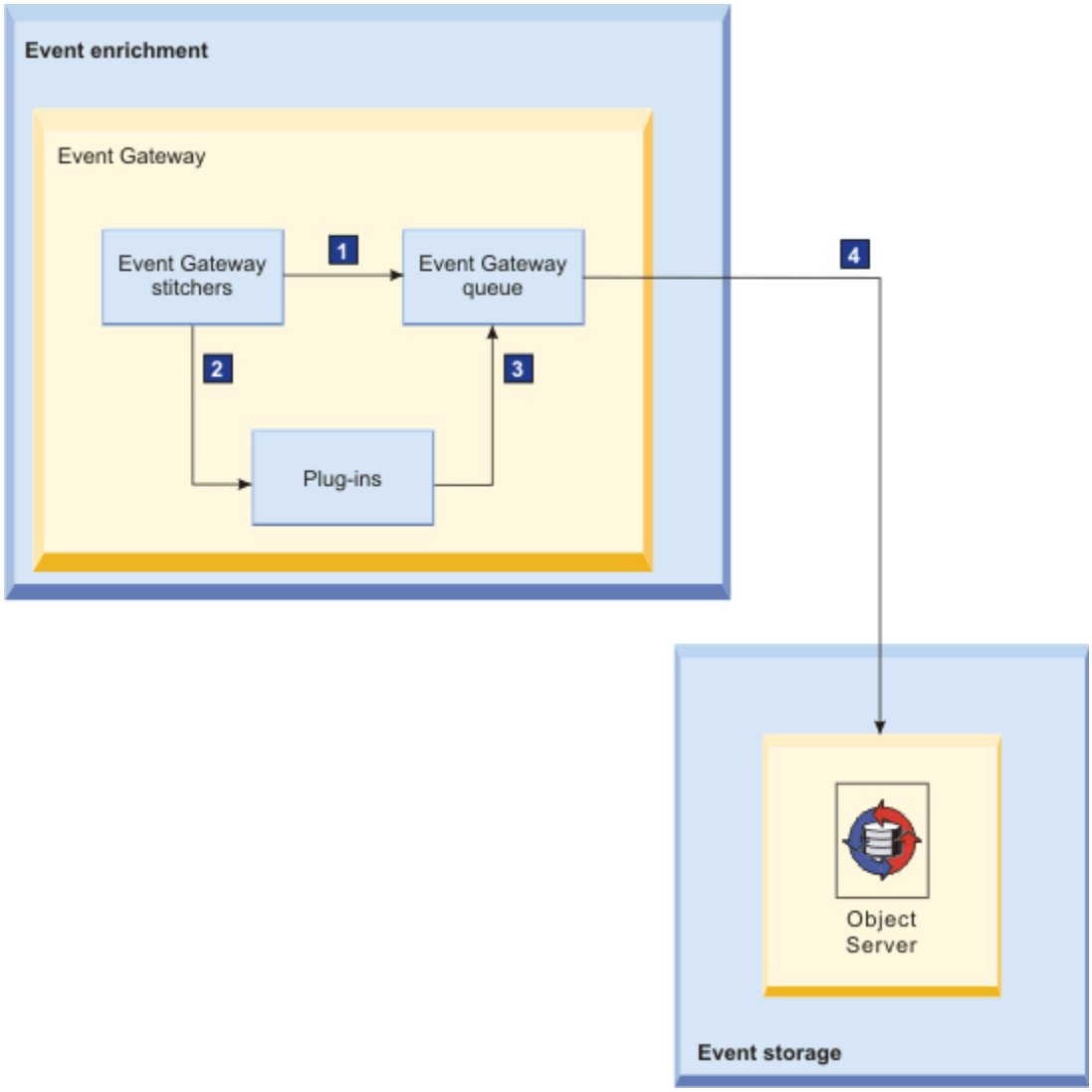


Figure 8. Outgoing Event Gateway queue

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
You can configure the interval that the Event Gateway uses to queue event enrichment updates to the ObjectServer.

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.

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
Use this information to understand what each RCA stitcher does.

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.

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 104. 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
4: Netcool/Visionary resolution	Resolution
7: Netcool/ISMs new alarm	Problem
8: Netcool/ISMs old alarm	Problem
11: More Severe	Problem

Table 104. Transition labels (continued)	
Value of Type field in alerts.status	**Event type as understood by the Event Gateway**
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.

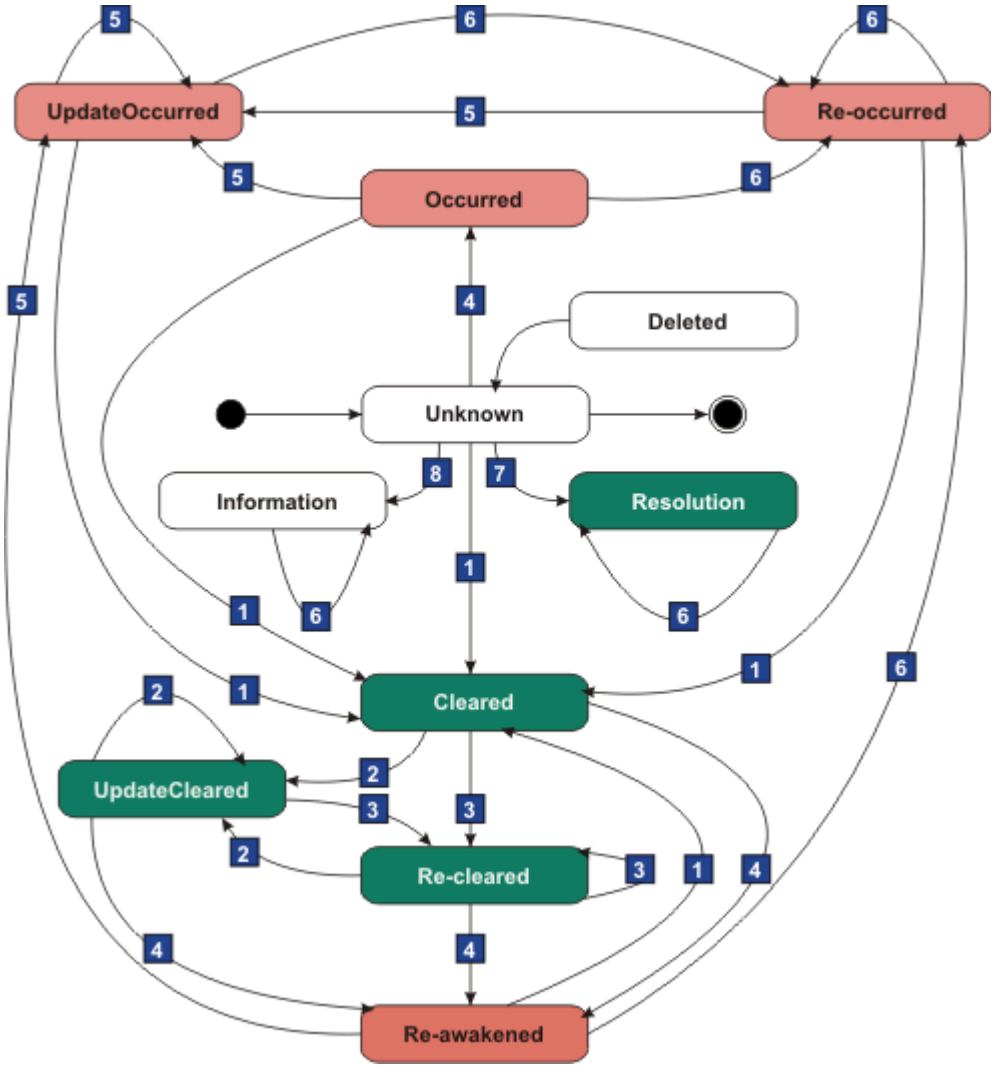


Figure 9. Event state diagram

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 105. 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
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 106. 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.

Table 106. Event states (continued)	
State	**Description**
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 107. 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.

Table 107. Lines of code relevant to the incoming event filter (continued)	
Line numbers	**Description**
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.

```
insert into config.precedence
(
    Precedence,
    EventMapName,
    NcoEventId
)
values
(
    910,
    "LinkDownIfIndex",
    "SNMPTRAP-LinkDown"
);
```

Event map selection methods
The event map and precedence can be assigned directly from the Tivoli Netcool/OMNIbus probe rules file, or in the Event Gateway configuration file.

Use one of the following methods to set the event map and precedence.

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. In the appropriate section of the rules file for the type of event you want to modify, add a line similar to the following example:

```
@NmosEventMap = "LinkDownIfIndex.910"
```

Event Gateway
Populate the config.precedence table using the insert defined in the Event Gateway configuration file, EventGatewaySchema.cfg. For an example, see “Event map selection using the Event Gateway” on page 542.

The eventPrecedence inserts configured in the Event Gateway override eventPrecedence inserts configured in the Tivoli Netcool/OMNIbus probe rules files. 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 current event maps.

Default event maps
The following table describes the default event maps.

Table 108. Default event maps

Event map	Stitcher called by event map	Event map description
ChassisFailure	LookupMainNode	Handles events where the `LocalNodeAlias` sufficiently specifies the chassis (main node) entity. Expected input fields: • `LocalNodeAlias`, where this field contains one of the following: – IP address – Entity name – DNS name – sysName – entityId
EMSNonPollingEvent	LookupEMSEntity	Handles polling events from element management systems (EMSs). Expected input fields: • `LocalNodeAlias`. • `LocalPriObj`. • `RemoteNodeAlias`, where this field contains the IP address or DNS name of polling station.
EMSPollingEvent	LookupEMSEntity	Handles non-polling events from element management systems (EMSs). Expected input fields: • `LocalNodeAlias`. • `LocalPriObj`. • `RemoteNodeAlias`.
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 one of the following: • IP address • Entity name • DNS name • sysName • entityId

Table 108. Default event maps (continued)

Event map	Stitcher called by event map	Event map description
EntityMibTrap	LookupEntPhysEntry	Handles traps from the ENTITY MIB. Expected input fields: • `LocalNodeAlias` , where this field contains one of the following: – IP address – Entity name – DNS name – sysName – entityId • `LocalPriObj` , where this field contains an index from the entPhysicalTable; for example, `'entPhysicalEntry.2'`.
EntityStateChange	LookupEntityId	Handles events raised by the Topology manager, ncp_model. No plug-ins currently register interest in this event map. Nevertheless the map allows for population of standard fields. The events raised by ncp_model are the following: • ItnmEntityCreation • ItnmEntityDeletion • ItnmMaintenanceState Expected input field: `NmosEntityId`.
genericip-event	LookupMainNode	Processes 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 one of the following: • IP address • Entity name • DNS name • sysName • entityId

Table 108. Default event maps (continued)

Event map	Stitcher called by event map	Event map description
ItncmResourceEvent	LookupMainNode	Used to handle events raised by Netcool Configuration Manager. This event map maps the entity identifier (`entityId`) in Network Manager to its corresponding Netcool Configuration Manager credentials. Expected input fields: `NmosEntityId`. If this field is not available, then uses as input: `LocalNodeAlias`, where this field contains one of the following: • IP address • Entity name • DNS name • sysName • entityId
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 one of the following: – IP address – Entity name – DNS name – sysName – entityId • `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 108. 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 one of the following: – IP address – Entity name – DNS name – sysName – entityId • `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 one of the following: – IP address – Entity name – DNS name – sysName – entityId • `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`.

Table 108. Default event maps (continued)

Event map	Stitcher called by event map	Event map description
LinkDownIfName	LookupIfEntry	Expects an interface event to be identified by the `ifName` value. Expected input fields: • `LocalNodeAlias` , where this field contains one of the following: – IP address – Entity name – DNS name – sysName – entityId • `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 one of the following: – IP address – Entity name – DNS name – sysName – entityId • `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 108. 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	Events assigned to this event map are used by the Disco plug-in to rediscover the device associated with them. By default, reboot events (events with event ID of NmosSnmpReboot) are assigned to this event map, based on the assumption that following reconfiguration of a device (for example, addition of a new card), the device will be rebooted, and then needs to be immediately discovered, so that the changes in configuration can be rapidly stored in the topology model. Expected input field: `LocalNodeAlias`, where this field contains one of the following: • IP address • Entity name • DNS name • sysName • entityId

Table 108. Default event maps (continued)		
Event map	**Stitcher called by event map**	**Event map description**
rttMonNotifications	LookupProbeSourceEntity	Fix Pack 4 Adds the main node and probe ID to incoming RTTMON traps. Expected input fields: • `LocalNodeAlias`, where this field contains the IP address, DNS name, sysName or entityName of the node. • `X733SpecificProb`, where this field identifies the trap type, for example `rttMonNotification`. • `LocalPriObj`, where this field identifies the trap index. • `ExtendedAttr`, where this field stores various RTT-related fields.

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 109. 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

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.

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 Reference*.

There are four types of Event Gateway stitcher:

- "Topology lookup stitchers" on page 553
- "Data extraction stitchers" on page 557
- "Entity retrieval stitchers" on page 558
- "Event enrichment stitchers" on page 560

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 562.

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 110. Event maps and stitcher selected

Event map	Selected stitcher
PollFailure	LookupIp.stch
ItnmMonitorEventNoRca	LookupEntityId.stch
PrecisionMonitorEvent	LookupEntityId.stch
LinkDownIfIndex	LookupIfEntry.stch

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.

Note: The code fragment shown below might be interspersed with other inserts in the actual code.

Table 111. Lines of code relevant to the incoming event filter

Line numbers	Description
1-12	Configure the event map PollFailure to select the stitcher LookupIp.stch.

Table 111. Lines of code relevant to the incoming event filter (continued)

Line numbers	Description
14-23	Configure the event map ItnmMonitorEventNoRca to select the stitcher LookupEntityId.stch.
25-34	Configure the event map PrecisionMonitorEvent to select the stitcher LookupEntityId.stch.
36-49	Configure the event map LinkDownIfIndex to select the stitcher LookupIfEntry.stch. 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.

```
insert into config.eventMaps
(
    EventMapName,
    Stitcher
    IsPollingEvent
)
values
(
    "PollFailure",
    "LookupIp"
    1
);

insert into config.eventMaps
(
    EventMapName,
    Stitcher
)
values
(
    "ItnmMonitorEventNoRca",
    "LookupEntityId"
);

insert into config.eventMaps
(
    EventMapName,
    Stitcher
)
values
(
    "PrecisionMonitorEvent",
    "LookupEntityId"
);

insert into config.eventMaps
(
    EventMapName,
    Stitcher,
    IsPollingEvent
    EventCanFlap
)
values
(
    "LinkDownIfIndex",
    "LookupIfEntry",
    0
```

```
    1
);
```

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 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 112. 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 following: • Polling engine, ncp_poller. • Topology manager, ncp_model. • Netcool Configuration Manager The stitcher 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 112. 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.

Table 112. Topology lookup stitchers (continued)

Stitcher	Description	Expected arguments	Returns
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.
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.
LookupProbeSource Entity	Looks up source entity for a Probe. Called by the `rttMonNotifications` Event Map.	None. Called from event map.	• 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 113. 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 Reference*.
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 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.

```
UserDefinedStitcher
{
    StitcherTrigger
    {
        // There is no trigger, as the eventMaps will automatically
        // call this with the event as the in-scope record
    }

    StitcherRules
    {
        Record entity;

        int nmosEntityId = eval(int, '&NmosEntityId');
        entity = GwEntityData( nmosEntityId );

        if ( entity == NULL )
        {
            entity = GwIpLookupUsing( "LocalNodeAlias" );
        }

        int foundEntity = 0;
        if ( entity <> NULL )
        {   ExecuteStitcher( "StandardEventEnrichment", entity );
            foundEntity = 1;
        }
```

```
        SetReturnValue( foundEntity );
    }
}
```

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 114. 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.	Integer representing a physical index.
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.	String representing an interface string, `ifString`.

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 115. 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 115. 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.

```
UserDefinedStitcher
{
    StitcherTrigger
    {
        //
        // Called from another stitcher using the syntax:
        //
        // text ifString = "";
        // ifString = ExecuteStitcher( 'ExtractIfString', myStringField );
        //
    }

    StitcherRules
    {
        text ifString = "";

        text ifInputStr = eval(text, '$ARG_1');

        text stringMatch = "^ifEntry\.(\S+)";

        int stringMatchCount = MatchPattern( ifInputStr, stringMatch );

        // We only recognise a match if we matched once on the entire field
        if (stringMatchCount == 1 AND REGEX0 == ifInputStr )
        {
            ifString = eval(text, '$REGEX1');
        }

        SetReturnValue( ifString );
    }
}
```

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 116. 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

Table 116. Entity retrieval stitchers (continued)		
Stitcher	**Description**	**Returns**
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
EntityFromNsei.stch	Takes as input a main node name in text form and a Network Service Entity End Point (NSEI) identifier in integer form and attempts to retrieve the implementing entity (typically an interface or chassis) for the NSEI.	The implementing entity (typically an interface or chassis) for the Network Service Entity End Point (NSEI) identified by the given NSEI on the given main node.
EntityIdFromProbeIdIndex	Takes as input the main node entity ID and probe ID, as configured on the device. Looks up the entity ID of a probe.	The entityId of the probe on the supplied device with the supplied probe ID, or 0.

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 117. 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 Reference*.

Table 117. Line-by-line description of the EntityFromIfString.stch stitcher (continued)	
Line numbers	**Description**
32	Pass the result of the entity lookup back to the invoking stitcher.

```
UserDefinedStitcher
{
    StitcherTrigger
    {
        // There is no trigger, as this is explicitly called from
        // other stitchers.
    }

    StitcherRules
    {
        text mainNodeEntityName = ARG_1;
        text ifString = ARG_2;

        Record entity;

        text ifStringQuery =
                "select * from ncimCache.entityData
                 where
                 ENTITYTYPE = 2
                 and
                 BASENAME = eval(text, '$mainNodeEntityName')
                 and
                 ( interface->IFNAME = eval(text, '$ifString')
                   or
                   interface->IFDESCR = eval(text, '$ifString')
                   or
                   interface->IFALIAS = eval(text, '$ifString') );";


        entity = RetrieveSingleOQL( ifStringQuery );

        SetReturnValue( entity );
    }
}
```

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 118. Event enrichment stitchers			
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.

Table 118. Event enrichment stitchers (continued)

Stitcher	Description	Expected arguments	Returns
ProbeSrcEndPtFromProbe	Retrieves information on the end points of the supplied probe.	The entity ID of the probe.	A record holding data on the associated source and target probe end points (if discovered).
IPSLAEventEnrichment	Enriches the current event by using GwEnrichEvent() using the supplied data.	The probe entity record as acquired from GwEntityData() for the probe entity id, probe end point data as acquired from ProbeSrcEndPtFromProbe.stch.	No return value.
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.

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 119. 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.

Table 119. Line-by-line description of the StandardEventEnrichment.stch stitcher (continued)	
Line numbers	**Description**
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 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 Reference*.

```
UserDefinedStitcher
{
    StitcherTrigger
    {
        // There is no trigger. This is called from other stitchers
        // with the event as the in-scope record, and the entity as
        // the single argument.
    }

    StitcherRules
    {
        Record entity = ARG_1;
        Record enrichedFields;

        int entityType = @entity.entityData.ENTITYTYPE;
        text entityName = @entity.entityData.ENTITYNAME;
        int entityId = @entity.entityData.ENTITYID;
        int mainNodeId = @entity.entityData.MAINNODEENTITYID;
        int managedStatus = GwManagedStatus( entityId );

        @enrichedFields.EntityType = entityType;
        @enrichedFields.EntityName = entityName;
        @enrichedFields.NmosDomainName = eval(text, '$DOMAIN_NAME');
        @enrichedFields.NmosEntityId = entityId;
        @enrichedFields.NmosManagedStatus = managedStatus;
        @enrichedFields.NmosObjInst = mainNodeId;

        GwEnrichEvent( enrichedFields );
    }
}
```

Related tasks

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.

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.

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 120. 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 SNMP probe (mttrapd) receives a Link Down trap from a device.
2. The probe rules process the trap and create a Tivoli Netcool/OMNIbus alert with the event ID `SNMPTRAP-LinkDown`. The snmptrap.rules rules file (from Netcool/OMNIbus Knowledge Library version 4.4.3 and above) assigns the event a value of `LinkDownIfIndex.910` for the `NmosEventMap` field, as determined by the probe rules. The `LinkDownIfIndex` value specifies that the LinkDownIfIndex event map should be used to process this event. The `.910` value appended to

the value specifies that the event has a precedence of 910. Events with a higher precedence suppress events with a lower precedence during RCA. The SNMP probe sends the alert to the ObjectServer.

3. The Event Gateway receives the event from the ObjectServer.
4. 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 the next step.
5. 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 on this information, the Event Gateway assigns the Occurred state to this event. This is a problem event and is a candidate for RCA.
6. 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
7. The Event Gateway determines how to handle this event, by determining which event map to use. Event maps define how to handle an event. A numerical precedence value can be associated with the event by the Trap Probe rules.

The event in this example has an NmosEventMap value of `LinkDownIfIndex.910`. The Event Gateway therefore processes the event using the event map LinkDownIfIndex. This event map covers link up and link down events from the Tivoli Netcool/OMNIbus SNMP 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 the event is sent for RCA, a precedence value of 910 is used for the event.

Note: The use of the NcoGateInserts file to define the event map and precedence is deprecated as of V4.4.3 of the Netcool/OMNIbus Knowledge Library.

8. The Event Gateway calls the LookupIfEntry stitcher named in the eventMap to match the event to an entity.

   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 to go up and down.
9. If the Event Gateway finds a matching entity in the network topology, the enriched event data is filtered by the outgoing field filter and placed on the Event Gateway queue. The event is also passed on to plugins that have registered interest.
10. Because the RCA plugin has registered interest, it receives the event for processing and performs RCA, using the precedence of 910 from the NmosEventMap field.
11. The event is enriched with the results of RCA and placed on the Event Gateway queue.

Related concepts

Quick reference for event enrichment
Use this information to understand how an event is processed as it passes through the Event Gateway.

Root Cause Analysis (RCA) plugin

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

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.

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 121. 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.	*Root Cause Analysis (RCA) plugin* in the *IBM Tivoli Network Manager IP Edition Administration Guide* *Event Gateway plugins* in the *IBM Tivoli Network Manager IP Edition Administration Guide*
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* in the *IBM Tivoli Network Manager IP Edition Administration Guide*
3. The event is passed to the ProcessEvent.stch, for root-cause analysis processing by the RCA stitchers.	

Related tasks

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.

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 122. 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.
10	Set the Precedence field to the value 910.

```
insert into config.precedence
(
    Precedence,
    EventMapName,
    NcoEventId
)
values
(
    910,
    "LinkDownIfIndex",
    "SNMPTRAP-LinkDown"
);
```

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 123. 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
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

Table 123. Default precedence values (continued)		
Value	**Meaning**	**Example events**
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 124. 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.

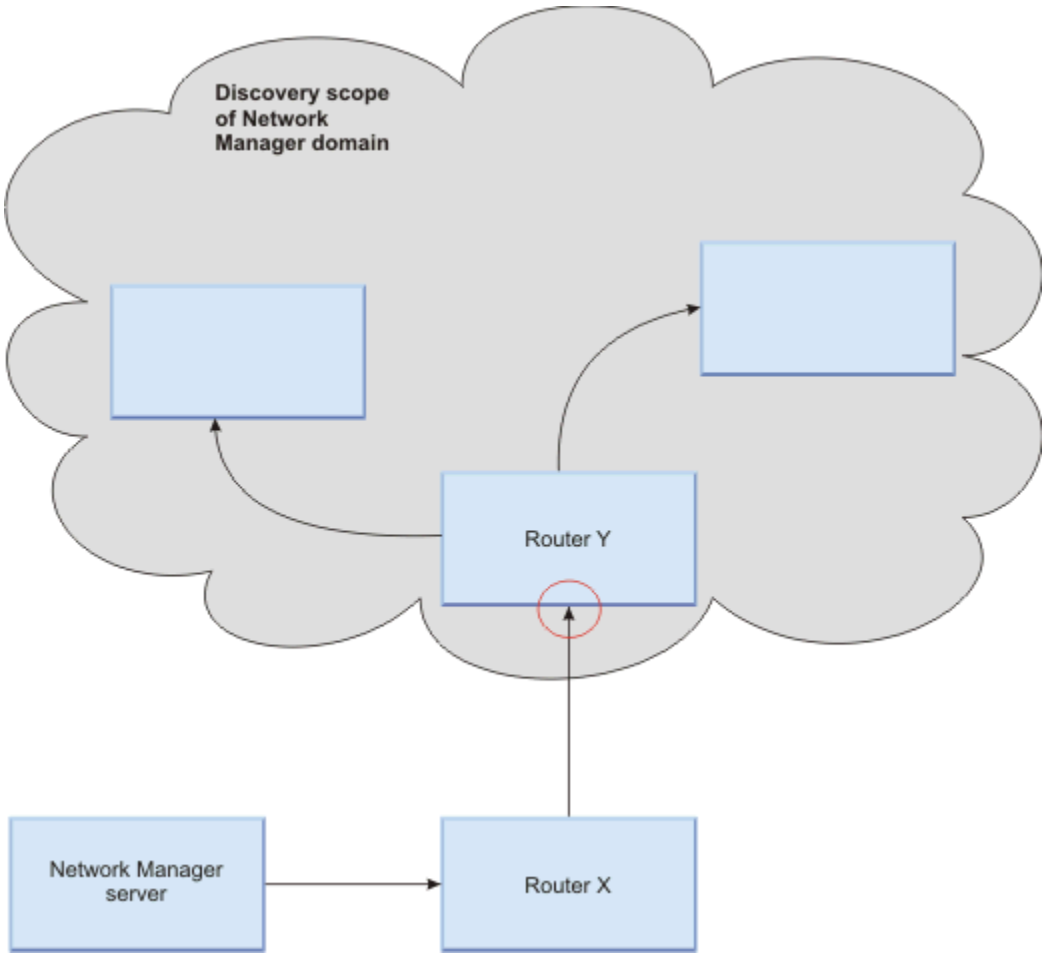


Figure 10. Ingress interface

Related tasks

Configuring the poller entity

When the Network Manager server is not within the scope of your network domain, or you have several domains, specify the IP address or DNS name of 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 placed into mojo.events but then ignored in terms of RCA processing.
- 0: instructs the RCA plug-in to process events from unmanaged devices as normal events.

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 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 then handles the event as if it had been deleted; that is, it reprocesses all the suppressees of the event because this event is 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.

RCA stitchers

RCA stitchers process a trigger event as it passes through the RCA plug-in.

RCA plug-in stitchers are stored in the following location: `$NCHOME/precision/eventGateway/stitchers/RCA`.

For information on stitcher language, see the *IBM Tivoli Network Manager 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	The `ProcessResolutionEvent.stch` stitcher processed cleared and deleted events. Fix Pack 3 The `TimedClearEventProcessing.stch` stitcher reprocesses events that were suppressed by cleared or deleted events. This stitcher incorporates a delay of 30 seconds to prevent issues caused by events arriving close to each other.

3. Two 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 on the exact same entity. 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

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 to 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 125. RCA stitchers

Stitcher	Description
ConnectedEntitySuppression.stch	Uses the trigger event to attempt to suppress other events by using connected entity principles. For example, when two connected interfaces both have an event on them, 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 by using contained entity principles. The event on the containing entity suppresses events on all contained entities.

Table 125. RCA stitchers (continued)

Stitcher	Description
EntitySuppression.stch	Uses the trigger event to attempt to suppress other events by 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 by using downstream entity principles.
PeerEntitySuppression.stch	Determines whether the trigger event can be suppressed by an existing OSPF or BGP event.
ProcessEvent.stch	This stitcher is called each time a trigger event is passed to the RCA plug-in. 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 by using other events. It then calls the `EntitySuppression`, `ContainedEntitySuppression`, `ConnectedEntitySuppression`, and `IsolatedEntitySuppression` stitchers, in that order, to try to suppress other events by using the trigger event. Starting in version 4.2, events are not sent back to the ObjectServer if there is no change in the RCA status.
ProcessResolutionEvent.stch	Handles resolution events, that is, events with event states `Cleared` and `Deleted`.
SuppressTrigger.stch	Determines whether the trigger event can be suppressed by an existing event.
TimedClearEventProcessing.stch	Fix Pack 3 A timed stitcher that runs every 30 seconds. This interval is defined by the `m_IntervalSeconds` property in the stitcher. It calls the `AmosTimedClearEventProcessing()` stitcher, which reprocesses the events that are suppressed by a clear event. The `AmosTimedClearEventProcessing()` stitcher processes the suppressed events only if the clear event occurred more than 30 seconds ago. This time threshold is defined by the `Age` property in the stitcher.

Table 125. RCA stitchers (continued)

Stitcher	Description
TimedEventSuppression.stch	The purpose of this stitcher is to save resources by preventing the RCA plug-in from processing flapping events. 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 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 deletes the Link Down event. No flapping events are processed by the TimedEventSuppression because they were already 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

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 root cause analysis system suppresses events on entities based on the following hierarchy: of suppression:

1. Same entity suppression.
2. Contained suppression
3. Isolated suppression
4. Connected suppression

For example, if an event is suppressed by another event on the same entity then that event cannot be contained-suppressed, isolated-suppressed or connected-suppressed. Similarly, if an event on an interface is suppressed by an event on the containing device, then that event cannot be isolated-suppressed or connected-suppressed.

The colors shown in the diagrams match the following event colors in the **Event Viewer**:

- Red: root-cause event.
- Purple: symptom (suppressed) event.

For more information on identifying and investigating root-cause events in the **Event Viewer**, see the *IBM Tivoli Network Manager User Guide*.

Related reference

RCA considerations in a cross-domain network

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.

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.

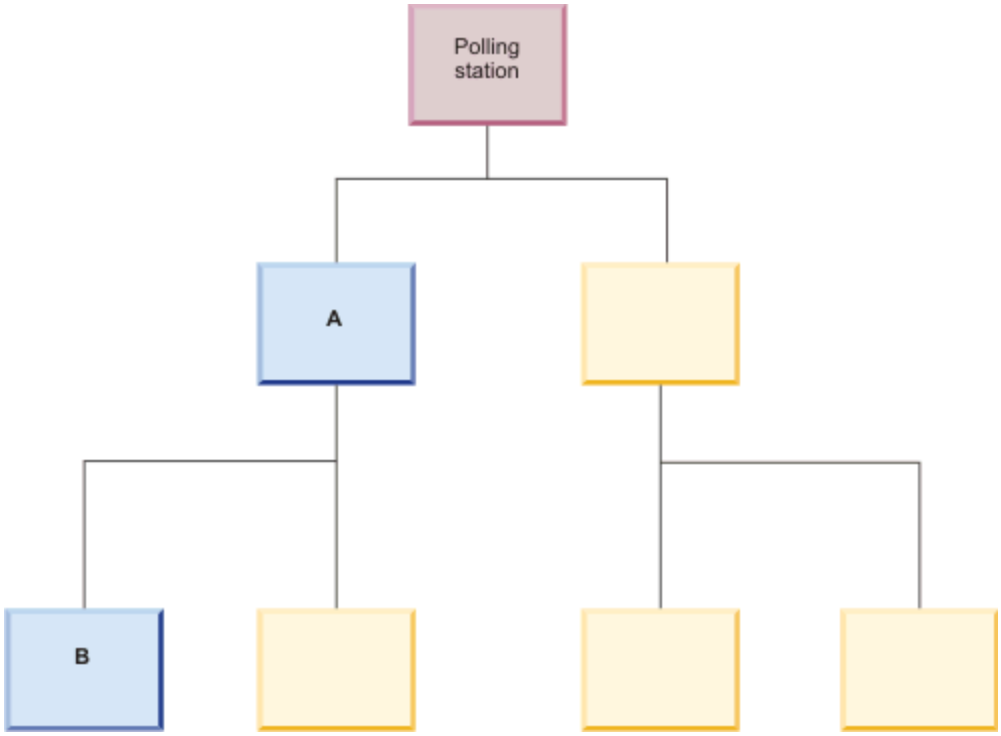


Figure 11. Downstream and upstream devices

Related reference

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*.

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*.

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 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.

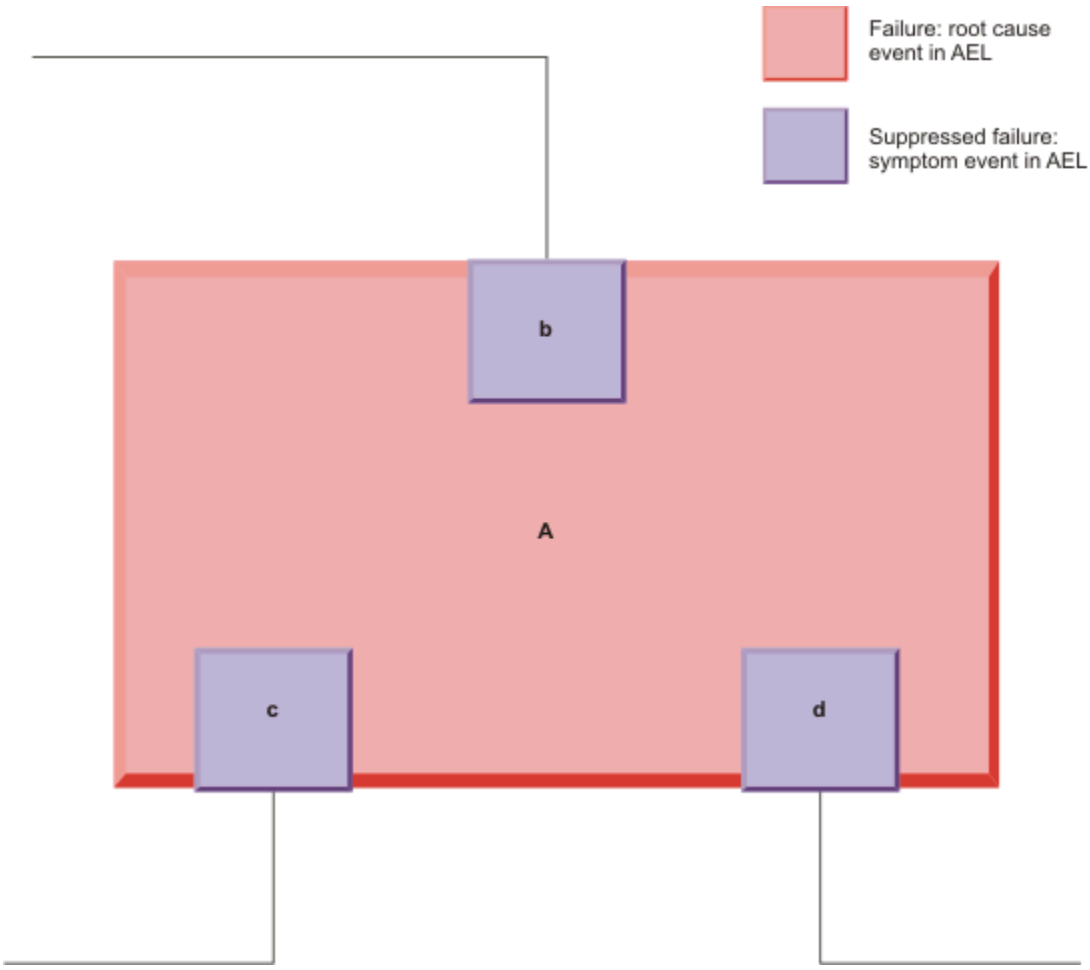


Figure 12. Chassis failure suppresses failures on contained interfaces

Connected interfaces

A chassis failure suppresses all failures on interfaces connected to that chassis device. Failures are suppressed on both upstream and downstream interfaces.

In the figure below, device A suppresses failures on interfaces b, c, and d.

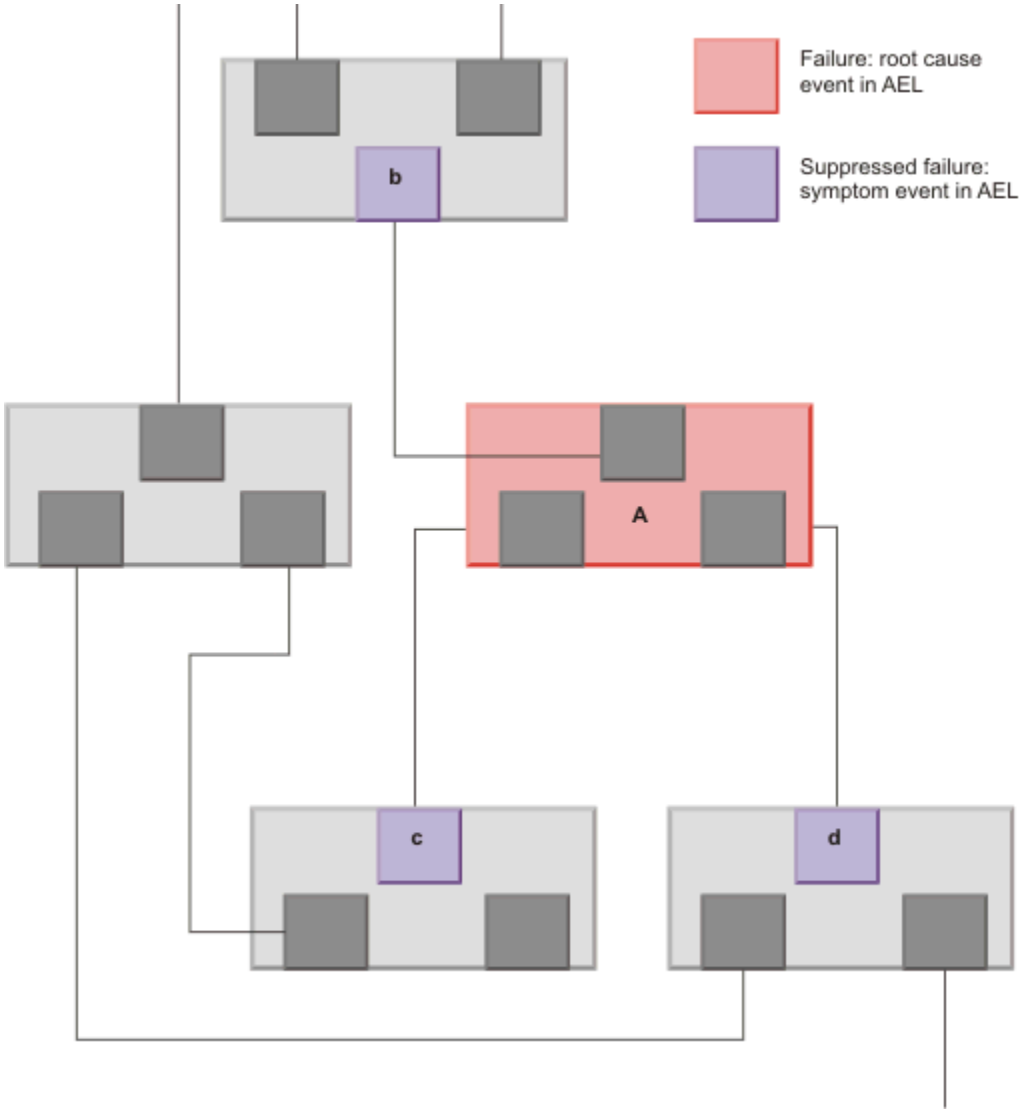


Figure 13. Chassis failure suppresses failures on connected interfaces

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.

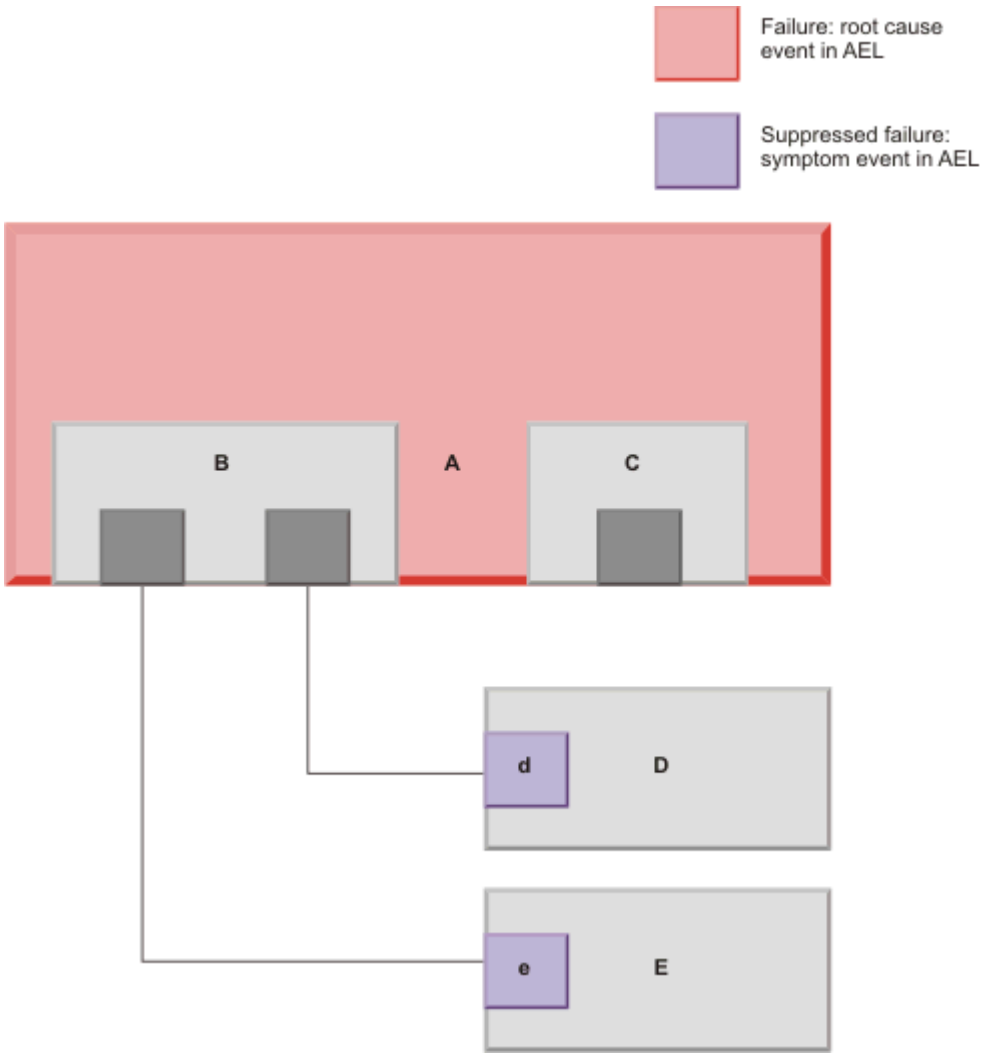


Figure 14. Chassis failure suppresses failures on devices connected to contained entities

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.

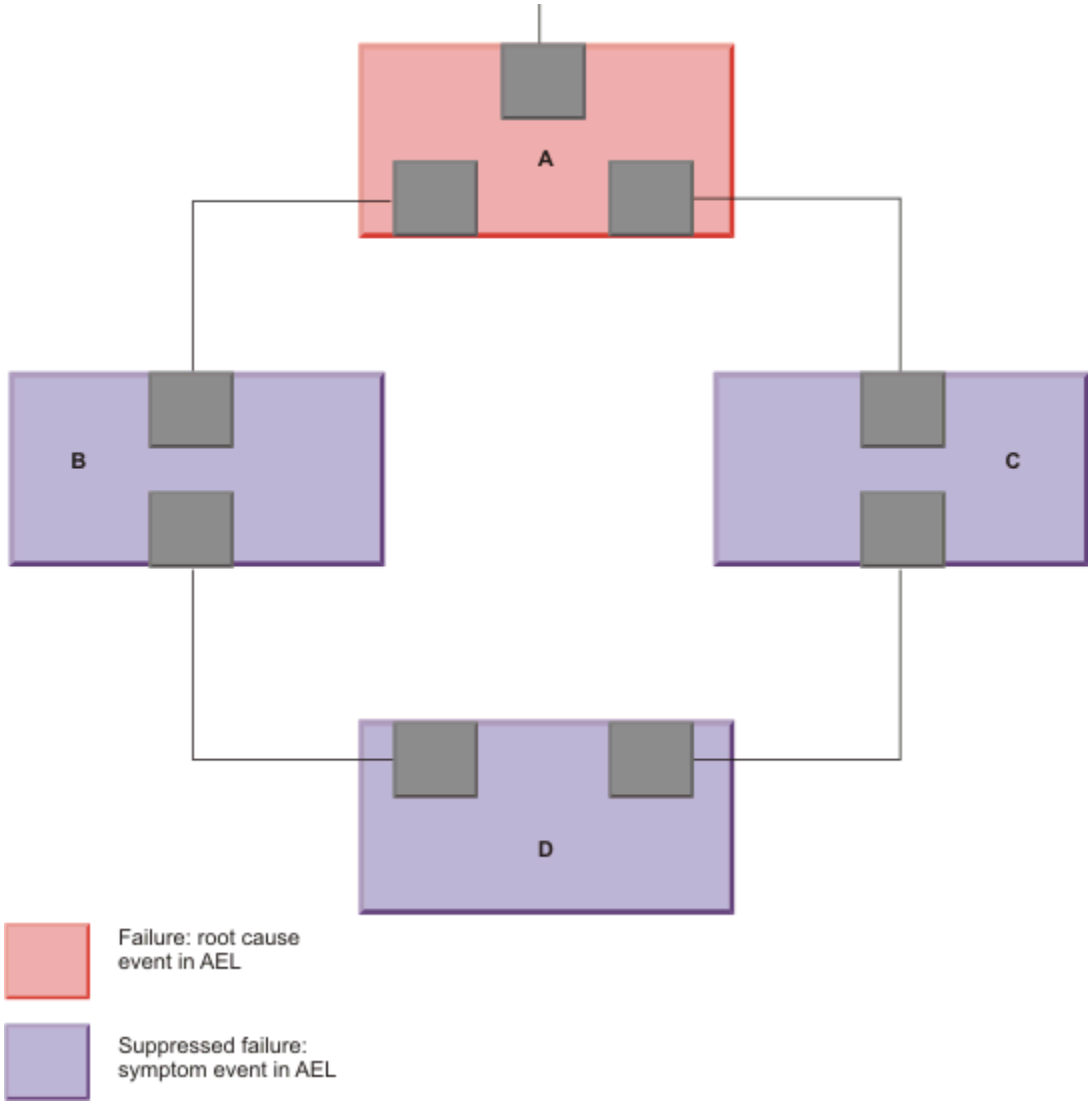


Figure 15. Chassis failure suppresses failures on downstream entities

Related reference

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.

Interfaces

If an interface is isolating downstream failures, then the interface failure can suppress the downstream failures.

Connected interfaces

An interface failure A can suppress a subsequent interface failure B if the two interfaces are directly connected. The interface whose suppression rule fires first, suppresses the other interface. Suppression of an interface failure B by an earlier failure A on a connected interface can only occur if the following conditions hold:

- Interface failure B is not already being contained-suppressed.
- Interface failure A is not already being isolated-suppressed.
- Interface failure A is not already being connected-suppressed.

- The chassis that contains interface A is not being isolated-suppressed.
- The chassis that contains interface A is not being connected-suppressed.

Furthermore, if interface failure B happened to be connect-suppressing interface failure A first and then later on the topology state changed so that instead A now isolated B, then the connected suppression of A by B would be removed and A would then isolate-suppress B.

Physical and logical interfaces

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.

Directly connected interface

A standard physical interface failure suppresses a second physical interface failure if the two interfaces are directly connected.

In general, a suppressed interface can suppress connected interfaces. However, note the following constraints related to suppression of directly connected interfaces:

- A contained-suppressed interface cannot be suppressed by an entity to which it is connected.
- A connected-suppressed interface cannot suppress an entity to which it is connected.
- An isolated-suppressed interface cannot suppress an entity to which it is connected.

In the figure below, failure on interface a suppresses the more recent failure on directly connected interface b.

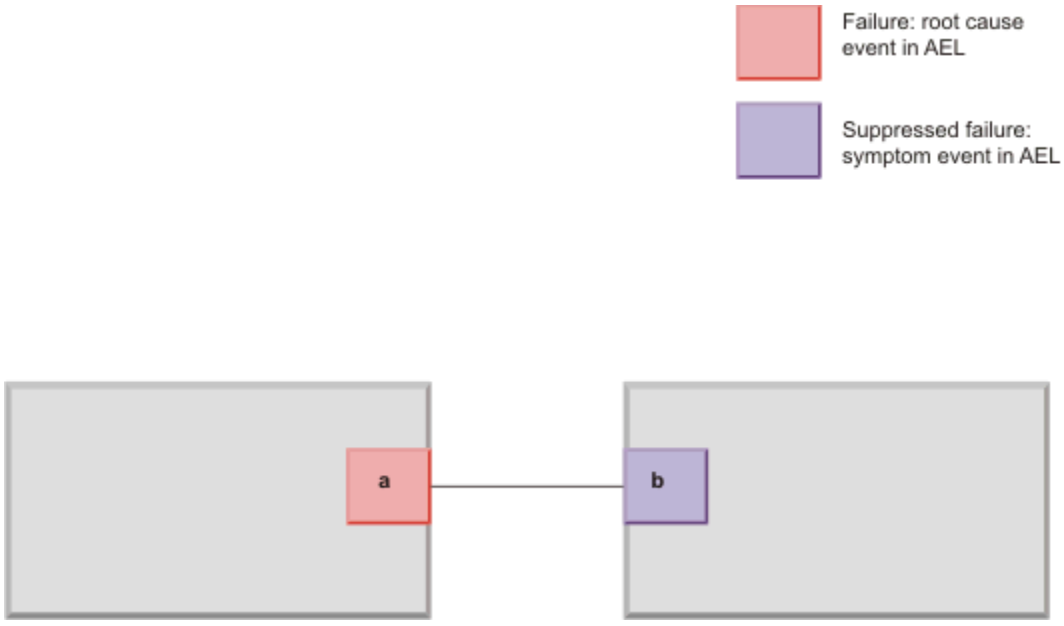


Figure 16. Interface failure suppresses more recent failure on directly connected neighbor interface

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.

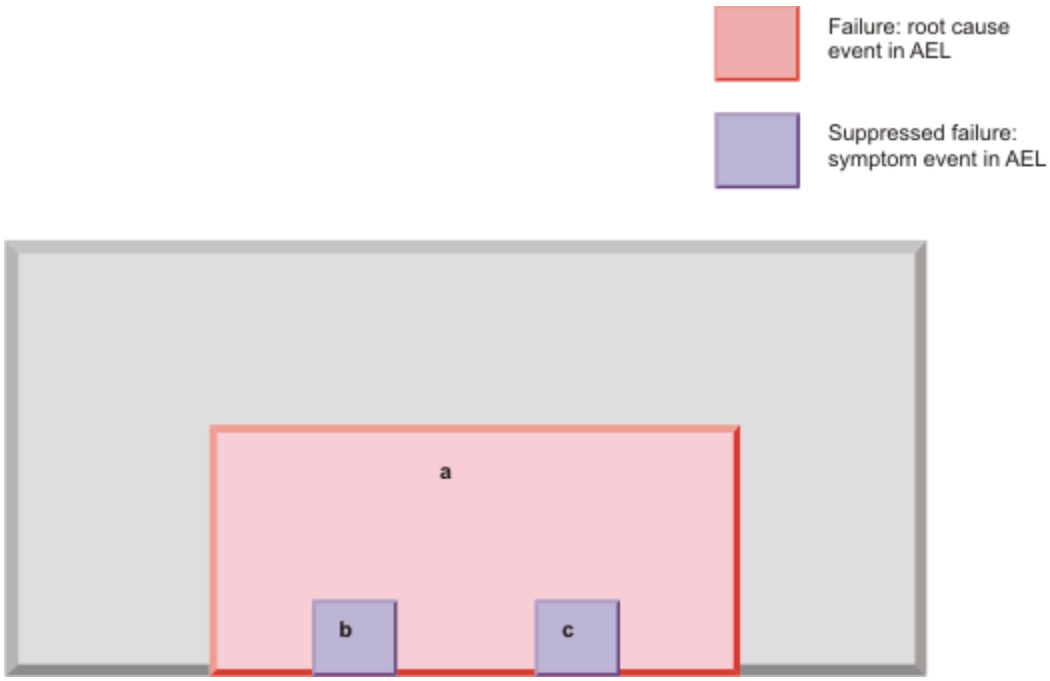


Figure 17. Physical interface failure suppresses failures on contained logical interfaces

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.

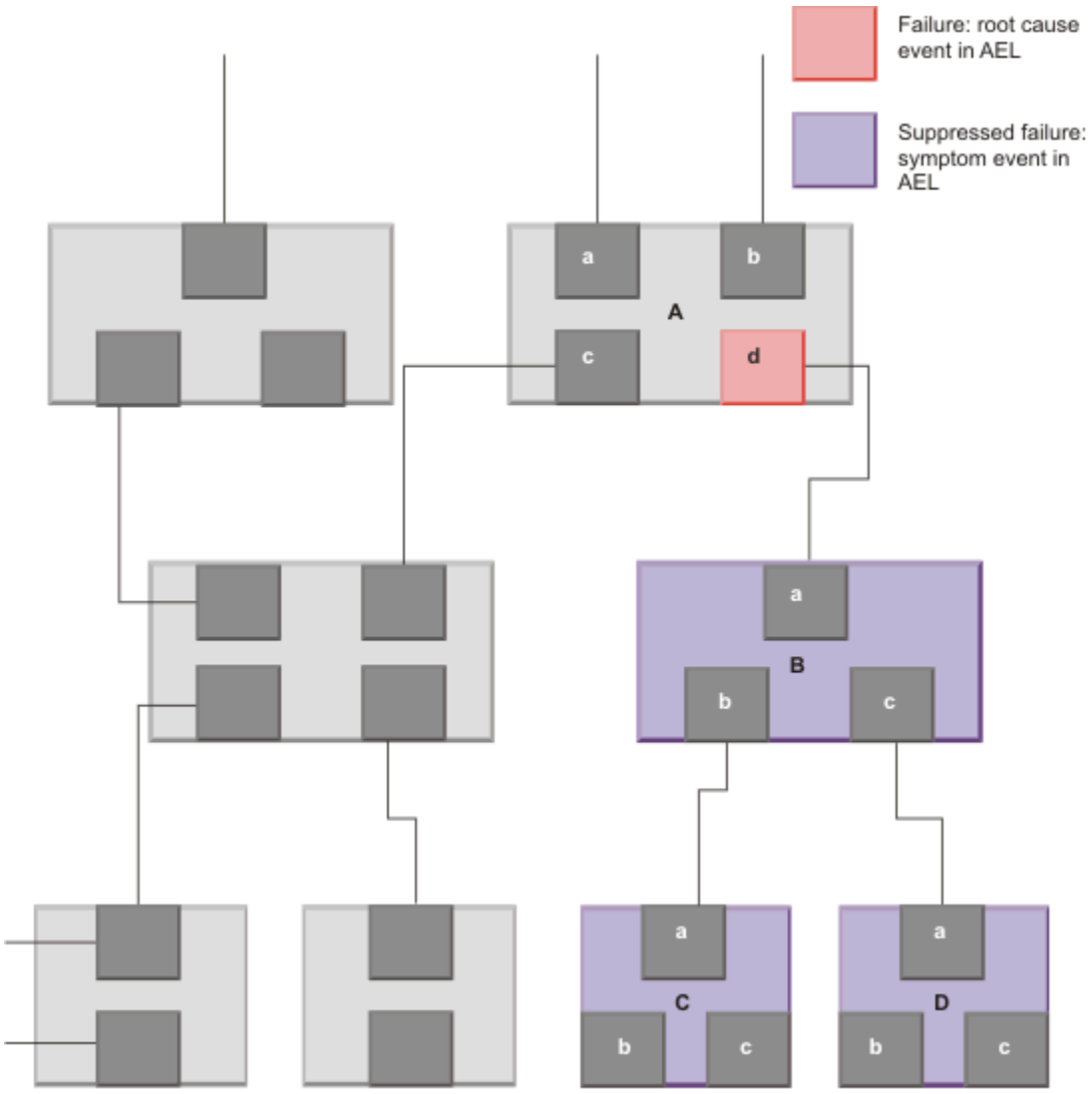


Figure 18. Interface failure suppresses more recent failure on directly connected neighbor interface

Related reference

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.

RCA considerations in a cross-domain network

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.

Related concepts

About cross-domain discovery
Cross-domain discovery can be configured to join two or more discovered domains together.

Related tasks

Configuring cross-domain discoveries
To enable links between devices in different domains to be shown in the network and topology views, you must configure and run cross-domain discoveries on the different domains.

Configuring the poller entity
When the Network Manager server is not within the scope of your network domain, or you have several domains, specify the IP address or DNS name of the poller entity.

Related reference

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.

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; however, it might be necessary to do this 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: There is no `atoz` database, nor are there `atoz.full` and `atoz.current` tables. This notional database and its tables form part of the OQL select mechanism used to communicate with the RCA plugin in order to obtain information about its topology graph

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 )
```

Related tasks

Querying management databases from the command line

You can use the OQL Service Provider to perform queries on Network Manager component databases.

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.

Related concepts

Poller entity
Use this information to understand what the poller entity is and how to configure it.

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:

```
X ------- E ------- F ------- G
\                             \
 \                             \
  \                             \
   \                             \
    A ------------ B ------------- C ------------- D
```

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.

Related concepts

Poller entity
Use this information to understand what the poller entity is and how to configure it.

RCA plug-in subscriptions

Use this information to understand which event maps and event states the RCA plugin is subscribed to.

Event map subscriptions

The RCA plug-in subscribes to the following event maps:

1. ChassisFailure
2. EMSNonPollingEvent
3. EMSPollingEvent
4. EntityFailure
5. EntityIfDescr
6. EntityMibTrap
7. ItnmLinkDownIfIndex
8. LinkDownIfDescr
9. LinkDownIfIndex
10. LinkDownIfName
11. NbrFail
12. NbrFailIfDescr
13. OSPFIfStateChange
14. OSPFIfStateChangeIP
15. PollFailure
16. 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

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.

Note: The RCA plugin is considerably more complex than the other Event Gateway plugins, and is documented under the RCA Plugin section.

The following plug-ins are enabled by default:

- Adaptive polling
- Compatibility Check
- Disco
- Failover
- PostNcimProcessing
- RCA
- All Service-affected event (SAE) plug-ins

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.

Table 126. 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.
Compatibility Check	Attempts to update the ObjectServer tables with any values that are needed by Network Manager.
Disco	Receives certain reboot events from the Event Gateway and triggers partial rediscovery based on these events. By default, this plug-in is only interested in events that indicate that a reboot has taken place.
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.
PostNcimProcessing	Triggers the stitching of multiple domains into a single aggregation domain when a topology update event is received. It does this by running any stitchers that are necessary after the NCIM database has been updated.
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.
SAE Aggregated Link	Generates a Service Affected Event (SAE) if any port that participates in a Link Aggregation Group has an alert of severity 5 or higher. Such ports are of Collection entity type 170, `aggregatedLink`.
SAE IP Path	Generates a Service Affected Event for IP paths.
SAE ITNM Service	Generic Network Manager service: can be configured to generated synthetic events when a an event occurs on a device associated with a custom service.
SAE MPLS VPN	Generates service-affected events for MPLS VPNs.

Table 126. 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

The individual topics on each plugin describe the operation of each plugin in more detail. To check which event maps are registered with each plugin, run the following command:

```
$NCHOME/precision/scripts/perl/scripts/ncp_gwplugins.pl
-domain DOMAIN -plugin Plugin name
```

Related concepts

Root Cause Analysis (RCA) plugin

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.

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

Event map subscriptions

The Adaptive polling plug-in subscribes to the following event maps:

1. ChassisFailure
2. EMSNonPollingEvent

3. EMSPollingEvent
4. EntityFailure
5. EntityIfDescr
6. EntityMibTrap
7. genericip-event
8. ItnmLinkDownIfIndex
9. ItnmMonitorEventNoRca
10. LinkDownIfDescr
11. LinkDownIfIndex
12. LinkDownIfName
13. NbrFail
14. NbrFailIfDescr
15. OSPFIfState
16. OSPFIfStateChange
17. OSPFIfStateChangeIP
18. PollFailure
19. PrecisionMonitorEvent
20. Reconfiguration

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

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 User Guide*.

Table 127. 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.

Table 127. Fields in the activeEvent table populated by the adaptive polling plug-in (continued)	
Plug-in	**Description**
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.
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 **Event Viewer**.
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 plug-in configuration parameters can optionally be set using the `ncp_gwplugins.pl` script.

Table 128. 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

Table 128. Optional configuration of the adaptive polling plug-in (continued)			
Parameter name	**Value**	**Purpose**	**Default**
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

Setting plug-in configuration parameters
You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

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 information

Tivoli Netcool/OMNIbus documentationWithin the IBM Knowledge Center for Tivoli Netcool/OMNIbus, refer to the topic *'Specifying the IDUC port '*. By default, when an ObjectServer starts, an available port number is chosen for the IDUC connection. You can also specify the IDUC port to use. You must specify the IDUC port when accessing an ObjectServer protected by a firewall.

Compatibility Check plug-in

The Compatibility Check plugin automatically adds fields and values required by Network Manager to the relevant databases in Tivoli Netcool/OMNIbus.

The plugin attempts to make the necessary database changes. If the changes cannot be made, the plugin generates an event of type ItnmConfiguration explaining what change was attempted.

Required fields

This plug-in does not have any required fields. The plug-in runs when Network Manager connects to a Tivoli Netcool/OMNIbus ObjectServer.

Event map subscriptions

The Compatibility Check plug-in does not subscribe to the any event maps.

Configuration of the plug-in

You do not need to configure the plug-in for normal operation. All required parameters are set by default.

Table 129. Mandatory configuration of the compatibility check plug-in			
Parameter name	**Description**	**Default value**	**Purpose**
StartupStitcher	Name of the stitcher in the StitcherSubDir directory to run when the plugin starts.	StartupCheck	If a startup stitcher name is supplied then this stitcher will be run without arguments at startup, upon SIGHUP, or upon failover or failback.

Table 129. Mandatory configuration of the compatibility check plug-in (continued)			
Parameter name	**Description**	**Default value**	**Purpose**
StitcherSubDir	Name of the subdirectory within the `$NCHOME/precision/eventGateway/stitchers/` directory containing the stitchers for this plug-in.	CompatibilityCheck	Specifying the name of this directory allows only this plug-in to parse its stitchers.

The following plug-in configuration parameters can optionally be set using the `ncp_gwplugins.pl` script.

Table 130. Optional configuration of the compatibility check plug-in			
Parameter name	**Value**	**Purpose**	**Default**
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

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 rebootDetection poll policy, and indicate that there has been a reboot on a device. To configure the Disco plug-in to handle other events, configure the related event to be handled by the Reconfiguration event map.

Important: The rebootDetection poll policy is not enabled by default. You must enable this poll policy in order for events to be generated based on a device reboot.

The following diagram provides a brief summary of the operation of the Disco plug-in.

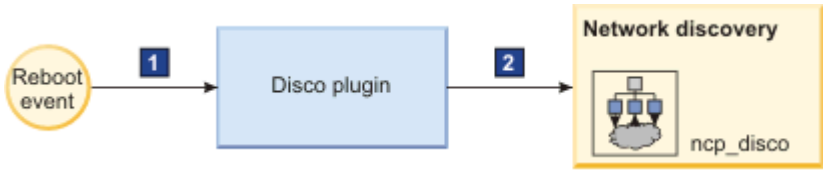


Figure 19. Operation of the Disco plug-in

1 Reboot event is received
A reboot event is passed from the Event Gateway to the Disco plugin

2 Partial rediscovery is triggered
The Disco plugin triggers a partial rediscovery of the device that rebooted.

Required fields

This plug-in expects events supplied by the Event Gateway to be with populated with the following fields:

- NmosObjInst

Event map subscriptions

The Disco plug-in subscribes to the Reconfiguration event map.

Configuration of the plug-in

The following plug-in configuration parameters can optionally be set using the `ncp_gwplugins.pl` script.

Table 131. Optional configuration of the Disco plug-in

Parameter name	Value	Purpose	Default
SchemaFile	Name of a schema file in `$NCHOME/etc/precision/` to be parsed at initialization.	If a schema file name is supplied then this file will be parsed at startup, upon SIGHUP, or upon failover or failback, before any startup stitcher is run.	
StartupStitcher	Name of a stitcher in the subdirectory within `$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
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`

Related tasks

Setting plug-in configuration parameters
You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

Enabling and disabling polls
To activate Network Manager polling, you must enable the poll policies.

Configuring the Disco plug-in
By default the Disco plug-in triggers rediscovery of devices associated with reboot events from the Tivoli Netcool/OMNIbus ObjectServer. You can configure the Disco plug-in to trigger rediscovery based on the receipt of any event.

Related information

Tivoli Netcool/OMNIbus documentationWithin the IBM Knowledge Center for Tivoli Netcool/OMNIbus, refer to the topic *'Specifying the IDUC port '*. By default, when an ObjectServer starts, an available port number is chosen for the IDUC connection. You can also specify the IDUC port to use. You must specify the IDUC port when accessing an ObjectServer protected by a firewall.

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.

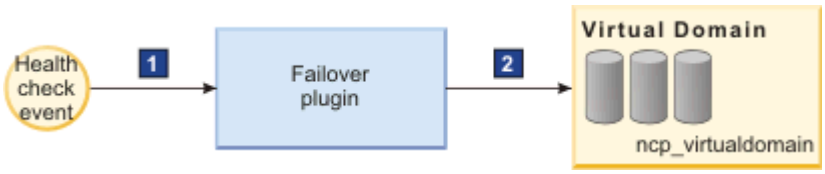


Figure 20. Operation of the Failover plug-in

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.

Event map subscriptions

This plug-in subscribes to the ItnmHealthChk event map.

The Failover plug-in subscribes to the following event states:

1. Cleared
2. Occurred
3. ReAwakened
4. ReCleared
5. ReOccurred
6. Resolution
7. ReSync

This plug-in subscribes to the following event states:

1. Information
2. Occurred

Related tasks

Setting plug-in configuration parameters
You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

Related information

Tivoli Netcool/OMNIbus documentationWithin the IBM Knowledge Center for Tivoli Netcool/OMNIbus, refer to the topic *'Specifying the IDUC port '*. By default, when an ObjectServer starts, an available port number is chosen for the IDUC connection. You can also specify the IDUC port to use. You must specify the IDUC port when accessing an ObjectServer protected by a firewall.

PostNCIMProcessing plug-in

The PostNCIMProcessing plug-in runs any stitchers that are necessary after the NCIM database has been 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 supplied by the Event Gateway to be with populated with the following fields:

- NmosObjInst

Event map subscriptions

The PostNcimProcessing plug-in subscribes to the ItnmStatus event map.

The PostNcimProcessing plug-in subscribes to the following event states:

1. Information
2. Occurred

Configuration of the plug-in

The following parameter is set by default and must be present in the ncmonitor.gwPluginConf table.

Table 132. Optional configuration of the PostNCIMProcessing 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 PostNCIMProcessing plug-in to parse its stitchers.	`PostNcimProcessing`

The following plug-in configuration parameters can optionally be set using the `ncp_gwplugins.pl` script.

Table 133. Optional configuration of the PostNCIMProcessing 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 tasks

Setting plug-in configuration parameters

You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

SAE Aggregated Link plug-in

The SAE Aggregated Link plugin generates Service Affected Events for Link Aggregation Groups (LAGs).

This plugin generates a Service Affected Event (SAE) if any port that participates in an LAG has an alert of severity 5 or has an alert which is a root cause or symptom alert (indicated by NmosCauseType of greater than 0). Aggregated Link entities are of Collection entity type 170, aggregatedLink.

Required fields

This plug-in expects events supplied by the Event Gateway to be with populated with the following fields:

- NmosEntityId
- Severity
- NmosCauseType
- NmosDomainName

SAE IP Path plug-in

The SAE IP Path plugin generates Service Affected Events for devices on IP Paths.

A synthetic event is 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.

Note: As with all SAE plug-ins, this plug-in generates a Service Affected Event (SAE) only if the underlying alerts have a severity of 5 or are root cause or symptom alerts (indicated by NmosCauseType of greater than 0).

Required fields

This plug-in expects events supplied by the Event Gateway to be with populated with the following fields:

- NmosEntityId
- Severity
- NmosCauseType
- NmosDomainName

Event map subscriptions

This plug-in subscribes to the following event state: ReSync.

SAE ITNM Service plug-in

The SAE ITNM Service plug-in allows you to extend the SAE functionality by creating events for your own service.

Configuring a custom service

The SAE ITNM Service plug-in can be configured to generate synthetic events when an event occurs on a device associated with a custom service. 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.

For more information on `ncp_disco` and on the `itnmService` NCIM database table see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Required fields

This plug-in expects events supplied by the Event Gateway to be with populated with the following fields:

- NmosEntityId
- Severity
- NmosCauseType
- NmosDomainName

Event map subscriptions

This plug-in subscribes to the following event state: ReSync.

SAE MPLS VPN plug-in

The SAE MPLS VPN plug-in creates Service Affected Events for MPLS VPNs.

A synthetic event is generated when a Severity 5 (critical) fault event, root cause fault event, or symptom 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.

For more information on the BGPPeerNextHopInterface discovery agent and the AddLayer3VPNInterfaceDependency.stch stitcher, see the *IBM Tivoli Network Manager IP Edition Administration Guide*

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.

Required fields

This plug-in expects events supplied by the Event Gateway to be with populated with the following fields:

- NmosEntityId
- Severity
- NmosCauseType
- NmosDomainName

Event map subscriptions

The MPLS VPN SAE plug-in subscribes to the following event state: ReSync.

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 134. 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.

Event map subscriptions

The zNetView plug-in subscribes to the following event maps:

1. ChassisFailure
2. EntityIfDescr
3. EntityFailure
4. EntityMibTrap
5. genericip-event
6. ItnmLinkDownIfIndex
7. ItnmMonitorEventNoRca
8. LinkDownIfIndex
9. LinkDownIfDescr
10. LinkDownIfName

11. NbrFailIfDescr
12. NbrFail
13. OspfIfState
14. OSPFIfStateChange
15. OSPFIfStateChangeIP
16. PollFailure
17. PrecisionMonitorEvent
18. 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

Configuration of the plug-in

The following plug-in configuration parameters can optionally be set using the `ncp_gwplugins.pl` script.

Table 135. Optional configuration of the adaptive polling plug-in

Parameter name	Value	Purpose	Default
SchemaFile	Name of a schema file in `$NCHOME/etc/precision/` to be parsed at initialization.	If a schema file name is supplied then this file will be parsed at startup, upon SIGHUP, or upon failover or failback, before any startup stitcher is run.	None
StartupStitcher	Name of a stitcher in the subdirectory within `$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
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`

Related tasks

Setting plug-in configuration parameters

You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

Related information

Tivoli Netcool/OMNIbus documentationWithin the IBM Knowledge Center for Tivoli Netcool/OMNIbus, refer to the topic *'Specifying the IDUC port '*. By default, when an ObjectServer starts, an available port number is chosen for the IDUC connection. You can also specify the IDUC port to use. You must specify the IDUC port when accessing an ObjectServer protected by a firewall.

Chapter 30. 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.

About this task

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.

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.

Before you begin

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*.

About this task

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.

© Copyright IBM Corp. 2006, 2021

For more information on the structure of NCIM cache tables and fields, see the *IBM Tivoli Network Manager Reference*.

The following steps explain how to configure this extra event enrichment.

Procedure

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 StandardEventEnrichment 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 StandardEventEnrichment 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 Reference*.

Table 136. 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 Reference*.
5	Retrieve the sysLocation data from the chassis table. **Note:** Whenever you retrieve data from NCIM cache, the field within the entity data must be specified 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.

```
Record mainNode = GwMainNodeLookupUsing( "LocalNodeAlias" );

if ( mainNode <> NULL )
{
```

```
      text sysLocation = @mainNode.snmpSystem.SYSLOCATION;

      if ( sysLocation <> eval(text, '&Location') )
      {
           @enrichedFields.Location = sysLocation;
      }
}
```

Related concepts

Outgoing field filter

The outgoing field filter defines the set of ObjectServer fields that may be updated by the Event Gateway.

Related reference

Example: StandardEventEnrichment.stch

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.

Before you begin

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*.

About this task

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 Reference*.

The following steps explain how to configure this extra event enrichment.

Procedure

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 StandardEventEnrichment 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 StandardEventEnrichment 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 Reference*.

Table 137. 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:** Whenever you retrieve data from NCIM cache, the field within the entity data must be specified 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.

```
if ( entityType == 2 )
{
    text interfaceName = @entity.networkInterface.IFNAME;

    if ( interfaceName <> eval(text, '&InterfaceName') )
    {
          @enrichedFields.InterfaceName = interfaceName;
    }
}
```

Related concepts

Outgoing field filter

The outgoing field filter defines the set of ObjectServer fields that may be updated by the Event Gateway.

Related reference

Example: StandardEventEnrichment.stch

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.

About this task

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.

Procedure

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

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.

Chapter 31. 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 root
```

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.

© Copyright IBM Corp. 2006, 2021

Chapter 32. 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 rereads the RCA`Schema.cfg` configuration file if the file has been modified since it was last read in at startup.
6. 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 RCA stitchers at this point.
7. Event Gateway retrieves all events from the Object Server, in the same way that it would at startup.
8. Event Gateway processes all events in the same way that it would at startup and passes any relevant events to the plug-ins.
9. Event Gateway sends a resync end event to its plug-ins.
10. Event Gateway resumes listening for events on the ObjectServer IDUC channel,

Related tasks

Configuring root-cause analysis
You can configure the RCA plug-in.

© Copyright IBM Corp. 2006, 2021

Chapter 33. Configuring common Event Gateway properties

You can configure common Event Gateway properties by editing the NCP_G_EVENT.props file.

Before you begin

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.

Note: The NCP_G_EVENT.props file can be domain-specific.

About this task

To configure the common properties, complete the following steps:

Procedure

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, uncomment, and update the property you want to change.

   You can edit the following properties:

   Ipc.Timeout
   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.

   Fix Pack 4 **SSLCommonNames**
   If you are connecting to a remote ObjectServer that is part of a failover pair using SSL, you must configure the names of the primary and backup ObjectServers. If you do not configure this value correctly, the Event Gateway does not connect to the ObjectServer failover pair.
4. Save the NCP_G_EVENT.props properties file.

Example

The following example configures a timeout of 20 seconds and specifies primary and backup ObjectServer names.

```
# INTEGER (IPC Session timeout), default 60 seconds
Ipc.Timeout: 20
# STRING (Comma separated list of common names),
used only if connecting to remote object servers
with SSL and failover configuration.
SSLCommonNames: 'NCOMS_Primary,NCOMS_Backup'
```

© Copyright IBM Corp. 2006, 2021

Chapter 34. 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.

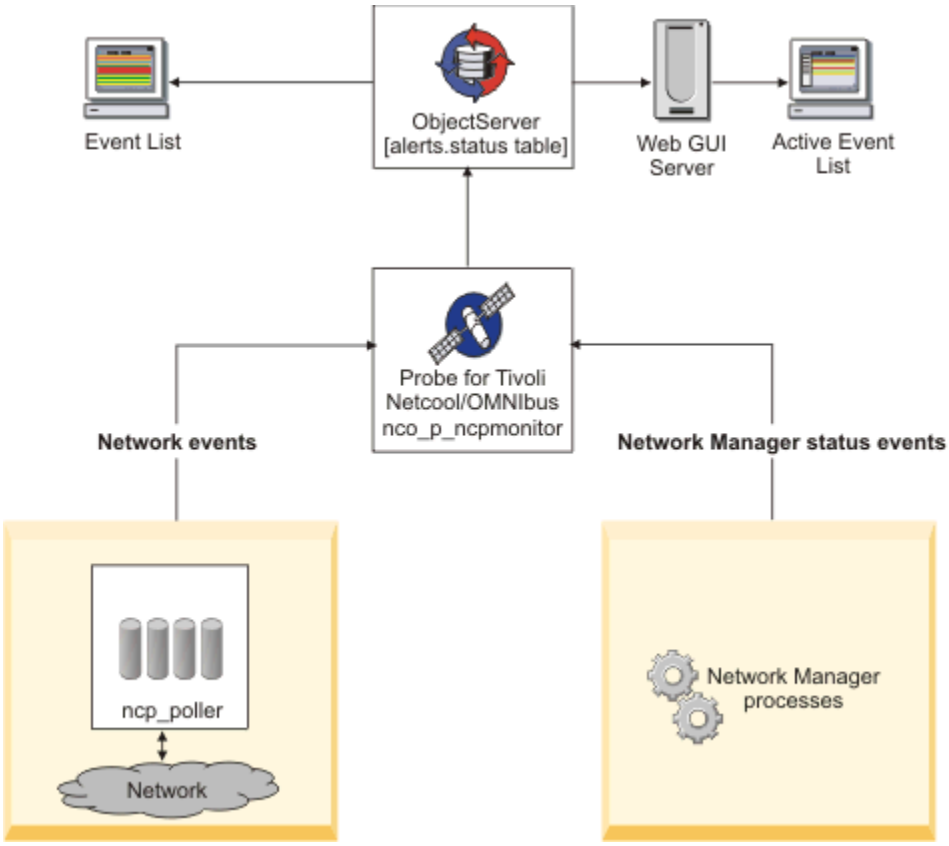


Figure 21. Flow of events from Network Manager to Tivoli Netcool/OMNIbus

Related reference

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 current event maps.

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`.

© Copyright IBM Corp. 2006, 2021

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.

ItnmConfiguration
This type of event is generated to indicate a configuration error. The Summary field describes the problem.

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.

The generation of this type of event can be disabled by changing the value of the m_CreateStchrEvents field in the disco.config table.

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.

The generation of this type of event can be disabled by changing the value of the m_CreateStchrEvents field in the disco.config table.

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.

- If ncp_disco is using the dNCIM database:
  - PopulateDNCIM

Discovery stitcher events in the ObjectServer are overwritten when a subsequent discovery is run.

The generation of this type of event can be disabled by changing the value of the m_CreateStchrEvents field in the disco.config table.

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 ChassisCreationEvents and IpInterfaceCreationEvents column to 1 in the INSERT statement for the model.config table. For example:

```
insert into model.config
(
    LingerTime,
    ChassisCreationEvents,
    IpInterfaceCreationEvents,
    MaintenanceStateEvents,
```

```
    ManagedStatusUpdateInterval,
    DeleteRenamedDevices,
    KeepOldEntityDetails
)
values
(
    3,
    1,  // If set to 1, generates ItnmEntityCreation and ItnmEntityDeletion events
when a chassis entity is created.
    1,  // If set to 1, generates ItnmEntityCreation and ItnmEntityDeletion events
when an interface with its own IP address is created.
    0,
    30,
    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.

ItnmGwPluginInitialization
This type event identifies whether each enabled Event Gateway plugin has been successfully initialized. If this is a Problem event, consult the Event Gateway log file (ncp_g_event.*domain_name*.log) for details.

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. This event contains the following parameter: `DISCOMODE`, for which 0 indicates that the topology was updated because a full discovery cycle finished, and 1 a partial discovery cycle.

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.

Related tasks
Monitoring process status messages
You can view status messages from Network Manager to understand the health and status of the product.

Chapter 35. Configuring Event Gateway plug-ins

You can configure Event Gateway plug-ins. You can also view currently enabled plug-ins.

Enabling and disabling plugins

You can enable and disable plug-ins.

About this task

Use the `ncp_gwplugins.pl` script to enable and disable plugins. The script is located at `$NCHOME/precision/scripts/perl/scripts/ncp_gwplugins.pl`.

To run the script to enable event map subscriptions, issue a command similar to the following. This example enables the zNetView plug-in in all domains.

```
$NCHOME/precision/bin/ncp_perl $NCHOME/precision/scripts/perl/scripts/
ncp_gwplugins.pl -domain NCOMS -plugin zNetView -enable -global
```

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 138. 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. You must specify a domain, even if you want to change a value for plug-ins in all domains.

© Copyright IBM Corp. 2006, 2021

Table 138. ncp_gwplugins.pl command-line options (continued)	
Command-line option	**Description**
`-plugin` *PluginName*	Name of the plug-in. **Note:** You can only run the script for one plug-in at a time. Plug-in names for use in this command line option are as follows. If the plug-in name is made up of more than one word, then the name must be enclosed in double quotes; for example: "Adaptive Polling". • Adaptive Polling • Compatibility Check • Disco • Failover • PostNcimProcessing • RCA • SAE Aggregated Link • 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 specified using the `-domain` parameter.

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.

About this task

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 139. 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. You must specify a domain, even if you want to change a value for plug-ins in all domains.
`-plugin` *PluginName*	Name of the plug-in. **Note:** You can only run the script for one plug-in at a time. Plug-in names for use in this command line option are as follows. If the plug-in name is made up of more than one word, then the name must be enclosed in double quotes; for example: "Adaptive Polling". • Adaptive Polling • Compatibility Check • Disco • Failover • PostNcimProcessing • RCA • SAE Aggregated Link • SAE IP Path • SAE ITNM Service • SAE MPLS VPN • zNetView

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.

About this task

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 140. 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. You must specify a domain, even if you want to change a value for plug-ins in all domains.
`-plugin` *PluginName*	Name of the plug-in. **Note:** You can only run the script for one plug-in at a time. Plug-in names for use in this command line option are as follows. If the plug-in name is made up of more than one word, then the name must be enclosed in double quotes; for example: "Adaptive Polling". • Adaptive Polling • Compatibility Check • Disco • Failover • PostNcimProcessing • RCA • SAE Aggregated Link • 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
Use this information to understand how an event is processed as it passes through the Event Gateway.

Quick reference for RCA
Use this information to understand how an event is processed as it passes through the RCA plug-in.

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.

Setting plug-in configuration parameters

You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

About this task

Use the `ncp_gwplugins.pl` script to set optional configuration parameters. The script is located 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 [ -global ]
-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 141. 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. You must specify a domain, even if you want to change a value for plug-ins in all domains.
`-global`	Optional. Sets the parameter for this plug-in in all domains. If you do not include the -global parameter, the parameter is set for the plug-in in only the specified domain.
`-name`*ParameterName*	Name of the parameter to set.

Table 141. ncp_gwplugins.pl command-line options (continued)	
Command-line option	**Description**
`-plugin` *PluginName*	Name of the plug-in. **Note:** You can only run the script for one plug-in at a time. Plug-in names for use in this command line option are as follows. If the plug-in name is made up of more than one word, then the name must be enclosed in double quotes; for example: "Adaptive Polling". • Adaptive Polling • Compatibility Check • Disco • Failover • PostNcimProcessing • RCA • SAE Aggregated Link • SAE IP Path • SAE ITNM Service • SAE MPLS VPN • zNetView
`-set`	Indicates that a variable is to be set.
`-value`*Parametervalue*	Value to set for this parameter.

Related concepts

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.

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.

Failover plug-in
Use this information to understand plug-in operation as well as configuration details associated with the plug-in.

PostNCIMProcessing plug-in
The PostNCIMProcessing plug-in runs any stitchers that are necessary after the NCIM database has been updated. By default, the plug-in triggers the stitching of multiple domains into a single aggregation domain when a topology update event is received.

zNetView plug-in

Use this information to understand plug-in prerequisites as well as configuration details associated with the plug-in.

Configuring the SAE plug-in

Use this information to understand how to configure the SAE plug-in.

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.

About this task

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.

Procedure

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 those 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).

Example

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'`

Configuring the Disco plug-in

By default the Disco plug-in triggers rediscovery of devices associated with reboot events from the Tivoli Netcool/OMNIbus ObjectServer. You can configure the Disco plug-in to trigger rediscovery based on the receipt of any event.

About this task

By default, only reboot events (events with an `eventId` field set to `NmosSnmpReboot`) are passed to the Disco plug-in. In order to pass events with a different `eventId` field value to the Disco plug-in, any one of the following methods can be used:

- Assign a different `eventId` to the existing Reconfiguration event map using one of the following event selection methods:
  - Configure the relevant IBM Tivoli Netcool/OMNIbus probe rules files.
  - Populate the Event Gateway config.precedence table using an appropriate insert.

  This can be done if the `LocalNodeAlias` of the event identifies the chassis, or if the `NmosEntityId` is set.

  Note: By default, the Reconfiguration event map does not pass events to the RCA plug-in for root-cause analysis. Do not use this method if the events to be passed to the Disco plug-in also need to be considered for root-cause analysis (RCA).

- Register an existing event map with the Disco plug-in, using the script `ncp_gwplugins.pl`.

  Note: All events using this existing event map will be passed to the Disco plug-in. Do not use this method if there are any events that use the existing map that must not trigger rediscovery.
- Create a new event map, assign the events with a particular `eventId` field value to that event map, and register the new event map with the Disco plugin, using the script `ncp_gwplugins.pl`. This is the method presented in the example that follows.

Configuring the Disco plug-in to respond to any event

This example assumes that there exists an event type, that is, a group of events with the same `eventId` field value, that contains an IP address in the LocalNodeAlias field. There is a requirement to consider this event type for RCA and also be able to rediscover the entities associated with events of this event type.

1. Add a new event map to the EventGatewaySchema.cfg file, with an appropriate lookup stitcher. The existing LookupMainNode stitcher will select a chassis from an IP. This example assumes that the name of the new event map is `NewEventMap`.

   Note: The EventGatewaySchema.cfg file is located at: `$NCHOME /etc/precision/EventGatewaySchema.cfg`.

   ```
   insert into config.eventMaps
   (
       EventMapName,
       Stitcher
   )
   values
   (
       "NewEventMap",
       "LookupMainNode"
   );
   ```

2. Associate the event type with the new event map by configuring the relevant probe rules file to populate the field NmosEventMap with the name of the new event map, `NewEventMap`.

   Note: Configuring the probe rules file is the preferred way of associating the event with the new event map, as this method updates the event at source, and therefore the change to the event map is automatically detected by the Netcool/OMNIbus Knowledge Library and by all Network Manager installations that connect to the Tivoli Netcool/OMNIbus ObjectServer to which the probe is connected. Where *EventId* is the event identifer for the event type to be handled by the new event map.

   a. On the server where the Netcool/OMNIbus Knowledge Library is installed, locate the rules file that corresponds to the eventID of the type of events that you want to customize. For vendor-specific rules, the rules files are placed under the vendor directory, for example, `include-snmptrap/cisco`.

   b. Locate the appropriate rules file with the suffix `user.include.rules`. For example, `/include-snmptrap/adtran/adtran-ADTRAN-ACTDAXL3-MIB.user.include.snmptrap.rules`. Putting your customizations in a separate file makes it easier to identify and back them up later.

   c. Edit the file and locate the section that corresponds to the type of trap that you want to customize.

   d. Set the @NmosEventMap field using the following format: *event map name.precedence value*. For example:

      ```
      @NmosEventMap = "NewEventMap.0"
      ```

3. Alternatively if you are unable to make changes to the probe rules file, then you can associate the event with the new event map by add a config.precedence insert to the EventGatewaySchema.cfg file, as in the following example:

   ```
   # Precedence of 0 implies this event cannot suppress others
   # 'MyEventId' should be the EventId field of the event in the Omnibus
   alerts.status table
   insert into config.precedence
   (
   ```

```
    Precedence,
    EventMapName,
    NcoEventId
)
values
(
    0,
    "NewEventMap",
    "EventId"
);
```

4. Configure the Disco and RCA plug-ins to subscribe to the new event map by running the following commands. This ensures that the event type of interest is considered as a candidate for suppression in RCA and is used to trigger rediscovery of the entity associated with the event.

```
ncp_gwplugins.pl -domain domain -plugin Disco -add -eventMap NewEventMap
ncp_gwplugins.pl -domain domain -plugin RCA -add -eventMap NewEventMap
```

   Where *domain* is the current domain.

5. Perform the following verification operations:

   - Verify the plug-in configuration by running the following commands:

```
ncp_gwplugins.pl -domain domain -plugin Disco
ncp_gwplugins.pl -domain domain -plugin RCA
```

     The responses to each of these commands should show that the plug-in subscribes to the NewEventMap event map.

   - Verify the event map configuration by running the following command:

```
ncp_gwplugins.pl -domain domain -eventMap NewEventMap
```

     The responses to this commands should show that the NewEventMap event map is subscribed to by the Disco and RCA plug-ins.

Related concepts

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.

Event map selection methods
The event map and precedence can be assigned directly from the Tivoli Netcool/OMNIbus probe rules file, or in the Event Gateway configuration file.

Related tasks

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.

Chapter 36. Configuring root-cause analysis

You can configure the RCA plug-in.

Related reference

Resynchronizing events with the ObjectServer

Issue the SIGHUP command to the Event Gateway to change the configuration of the Event Gateway.

Configuring the poller entity

When the Network Manager server is not within the scope of your network domain, or you have several domains, specify the IP address or DNS name of the poller entity.

About this task

The poller entity is a designated node in the topology from which RCA is calculated. By default, the poller entity is the same as the server where the Network Manager core components are installed.

Define one poller entity for each discovery domain. The poller entity must exist in the discovered network topology for the domain in which it is defined. If the server where Network Manager is installed is itself not in the topology, set the poller entity to the nearest node that is within the topology.

For effective downstream and isolated suppression, set the poller entity to be the server from where the network is polled, or as close to that server as possible.

The configuration file for the Event Gateway is the EventGatewaySchema.*DOMAIN*.cfg configuration file, where *DOMAIN* is the name of the domain in which the Event Gateway is running. This file is located at: `NCHOME/etc/precision/`. The poller entity value is stored in the config.defaults table, in the field NcpServerEntity.

Procedure

1. Open the EventGatewaySchema.*DOMAIN*.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,
    NcimHandleCount,
    NcpServerEntity
)
values
(
    5,
    5,
    1
    ""
);
```

   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, for example:

```
insert into config.defaults
(
    IDUCFlushTime,
    ObjectServerUpdateInterval,
    NcimHandleCount,
    NcpServerEntity
)
values
```

© Copyright IBM Corp. 2006, 2021

```
(
    5,
    5,
    1
    "switch108-abc.example.co.uk"
);
```

4. If you are using RCA across multiple domains, repeat these steps for each domain, using a different poller for each domain.

Related concepts

Poller entity

Use this information to understand what the poller entity is and how to configure it.

Related reference

RCA considerations in a cross-domain network

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.

Configuring the maximum age difference for events

By default, events can suppress each other regardless of the age of the events. An event received today can suppress an event received yesterday. 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.

About this task

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`.

Procedure

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:

```
//  MaxAgeDifference is in minutes
insert into config.defaults
(
    RequeueableEventIds,
    MaxAgeDifference,
    HonourManagedStatus,
    TopologyChangesThreshold,
    GraphTopologyNames
)
values
(
    [
        'NmosPingFail',
        'NmosSnmpPollFail'
    ],
        0,
        1,
        100,
    [

        'Layer2Topology',
        'LocalVlanTopology'
    ]
);
```

   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 that have a difference in age greater than fifteen minutes cannot suppress each other.

Chapter 37. 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 at http://www.ibm.com/support/knowledgecenter/SSSHTQ/landingpage/NetcoolOMNIbus.html.

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 uncomment and then modify the value as required. String values must be enclosed in quotation marks; other values do not require quotation marks. For example:

```
Buffering         :    1
BufferSize        :    15
```

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.

Note: The following properties have standard default values:

Server
Defaults to the ObjectServer identified in the ConfigItnm schema, thereby ensuring consistency with the Network Manager gateway.

PropsFile
Defaults to $NCHOME/probes/*platform*/nco_p_ncpmonitor.props.

RulesFile
Defaults to $NCHOME/probes/*platform*/nco_p_ncpmonitor.rules.

© Copyright IBM Corp. 2006, 2021

MessageLog
Defaults to $NCHOME/log/precision/nco_p_ncpmonitor.*domain_name*.log.

RawCaptureFile
Defaults to $NCHOME/var/precision/nco_p_ncpmonitor.*domain_name*.cap.

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://www.ibm.com/support/knowledgecenter/SSSHTQ/landingpage/NetcoolOMNIbus.html.

nco_p_ncpmonitor.rules configuration reference

The $NCHOME/probes/*arch*/nco_p_ncpmonitor.rules file defines how the Probe for Tivoli Netcool/OMNIbus processes Network Manager event data to create a meaningful Tivoli Netcool/OMNIbus event.

In practice, this 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://www.ibm.com/support/knowledgecenter/SSSHTQ/landingpage/NetcoolOMNIbus.html.

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
ExtraInfo->SERVICE	LocalPriObj

Note: The full input to and output from the probe rules can be seen in the probe trace file. Set the trace to debug 4. The probe trace file can be found at: `$NCHOME/log/precision`. For more information on setting log levels, see *IBM Tivoli Network Manager IP Edition Administration Guide*.

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 142. 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 142. 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)

Table 142. Network Manager fields that populate events (continued)

Network Manager field name	Field content	Always available?
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
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 143. 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 **Event Viewer** 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.

Table 143. alerts.status fields used by Network Manager (continued)

alerts.status field	Data type	Description	Network Manager field name/ Default value in rules file
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.
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	

Table 143. alerts.status fields used by Network Manager (continued)

alerts.status field	Data type	Description	Network Manager field name/ Default value in rules file
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
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.	

Table 143. alerts.status fields used by Network Manager (continued)

alerts.status field	Data type	Description	Network Manager field name/ Default value in rules file
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.	
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)	

Table 143. alerts.status fields used by Network Manager (continued)

alerts.status field	Data type	Description	Network Manager field name/ Default value in rules file
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
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 **Event Viewer**, and in the Occurred column in the `mojo.events` table.	

Table 143. alerts.status fields used by Network Manager (continued)			
alerts.status field	**Data type**	**Description**	**Network Manager field name/ Default value in rules file**
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 at IBM Tivoli Netcool/OMNIbus product information.

Notices

This information applies to the PDF documentation set for IBM Tivoli Network Manager IP Edition.

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 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

© Copyright IBM Corp. 2006, 2021

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 144 on page 651 are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Table 144. IBM trademarks

AIX	Informix®	PR/SM
BNT	iSeries	System p
ClearQuest®	Lotus®	System z®
Cognos	Lotus	Tivoli
Db2	Netcool	WebSphere
Db2 Universal Database	NetView®	z/OS
developerWorks®	OMEGAMON®	z/VM®
DS8000	Passport Advantage	zSeries
Enterprise Storage Server®	PowerPC	
IBM	PowerVM®	

Adobe, Acrobat, Portable Document Format (PDF), PostScript, and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

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 may collect IP addresses, user names and passwords for the purpose of performing network discovery. Failure to enable the collection of this information would likely eliminate important functionality provided by this Software Offering. You as customer should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

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 and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details, and the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/privacy.



Part Number:

Printed in the Republic of Ireland

(1P) P/N:

2021-4213-01

