Network Manager IP Edition
Version 4 Release 1.1

Management Database Reference

IBM
## Contents

**About this publication** ........................................ v
  Intended audience .............................................. vi
  What this publication contains ................................ vi
  Publications ......................................................... v i
  Accessibility ....................................................... ix
  Tivoli technical training ........................................ ix
  Support and community information ................................ x
  Conventions used in this publication ................................ xi

**Chapter 1. ncp_class database** .......................... 1
  class.activeClasses table .......................................... 1
  class.staticClasses table .......................................... 2
  class.classIds table ............................................... 2

**Chapter 2. ncp_ctrl database** ........................... 5
  The services.config Table .......................................... 5
  The services.inTray Table .......................................... 5
  The services.slaveCtrl Table ...................................... 7
  The services.unControlled Table .................................. 8

**Chapter 3. ncp_trapMux database** ..................... 11
  trapMux.command table ............................................. 11
  trapMux.config table .............................................. 11
  trapMux.sinkHosts table .......................................... 11

**Chapter 4. ncp_virtualdomain database** ............... 13
  config database schema ............................................ 13
  state database schema ............................................ 14
  Example Virtual Domain configuration ............................ 16

**Appendix. Network Manager glossary** .................. 19

**Notices** ........................................................... 23

**Trademarks** ........................................................ 25

**Index** ............................................................... 27
About this publication

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

The *IBM Tivoli Network Manager IP Edition Management Database Reference* describes the schemas of the component databases in Network Manager.

Intended audience

This publication is intended for system administrators who are responsible for configuring and administering IBM Tivoli Network Manager IP Edition, and for advanced users who need to query the component databases.

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

What this publication contains

This publication contains the following sections:

- **Chapter 1, “ncp_class database,” on page 1**
  Describes the ncp_class database, which enables the Active Object Class manager, ncp_class, to manage active object classes (AOCs).
- **Chapter 2, “ncp_ctrl database,” on page 5**
  Describes the ncp_ctrl database, which enables the master process controller, ncp_ctrl, to manage and monitor the status of Network Manager processes.
- **Chapter 3, “ncp_trapMux database,” on page 11**
  Describes the ncp_trapMux database, which enables the SNMP Trap Multiplexer, ncp_trapMux, to forward traps to multiple ports and capture and replay traps.
- **Chapter 4, “ncp_virtualdomain database,” on page 13**
  Describes the ncp_virtualdomain database, which enables the Virtual Domain subsystem to support Network Manager failover.
Publications

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

Your Network Manager library

The following documents are available in the Network Manager library:

- **IBM Tivoli Network Manager IP Edition Release Notes, GI11-9354-00**
  Gives important and late-breaking information about IBM Tivoli Network Manager IP Edition. This publication is for deployers and administrators, and should be read first.

- **IBM Tivoli Network Manager Getting Started Guide, GI11-9353-00**
  Describes how to set up IBM Tivoli Network Manager IP Edition after you have installed the product. This guide describes how to start the product, make sure it is running correctly, and discover the network. Getting a good network discovery is central to using Network Manager successfully. This guide describes how to configure and monitor a first discovery, verify the results of the discovery, configure a production discovery, and how to keep the network topology up to date. Once you have an up-to-date network topology, this guide describes how to make the network topology available to Network Operators, and how to monitor the network. The essential tasks are covered in this short guide, with references to the more detailed, optional, or advanced tasks and reference material in the rest of the documentation set.

- **IBM Tivoli Network Manager IP Edition Product Overview, GC27-2759-00**
  Gives an overview of IBM Tivoli Network Manager IP Edition. It describes the product architecture, components and functionality. This publication is for anyone interested in IBM Tivoli Network Manager IP Edition.

- **IBM Tivoli Network Manager IP Edition Installation and Configuration Guide, SC27-2760-00**
  Describes how to install IBM Tivoli Network Manager IP Edition. It also describes necessary and optional post-installation configuration tasks. This publication is for administrators who need to install and set up IBM Tivoli Network Manager IP Edition.

- **IBM Tivoli Network Manager IP Edition Administration Guide, SC27-2761-00**
  Describes administration tasks for IBM Tivoli Network Manager IP Edition, such as how to administer processes, query databases and start and stop the product. This publication is for administrators who are responsible for the maintenance and availability of IBM Tivoli Network Manager IP Edition.

- **IBM Tivoli Network Manager IP Edition Discovery Guide, SC27-2762-00**
  Describes how to use IBM Tivoli Network Manager IP Edition to discover your network. This publication is for administrators who are responsible for configuring and running network discovery.

- **IBM Tivoli Network Manager IP Edition Event Management Guide, SC27-2763-00**
  Describes how to use IBM Tivoli Network Manager IP Edition to poll network devices, to configure the enrichment of events from network devices, and to manage plug-ins to the Tivoli Netcool/OMNIbus Event Gateway, including configuration of the RCA plug-in for root-cause analysis purposes. This publication is for administrators who are responsible for configuring and running network polling, event enrichment, root-cause analysis, and Event Gateway plug-ins.
Describes how to use IBM Tivoli Network Manager IP Edition to troubleshoot network problems identified by the product. This publication is for network operators who are responsible for identifying or resolving network problems.

- **IBM Tivoli Network Manager IP Edition Network Visualization Setup Guide, SC27-2764-00**
  Describes how to configure the IBM Tivoli Network Manager IP Edition network visualization tools to give your network operators a customized working environment. This publication is for product administrators or team leaders who are responsible for facilitating the work of network operators.

- **IBM Tivoli Network Manager IP Edition Management Database Reference, SC27-2767-00**
  Describes the schemas of the component databases in IBM Tivoli Network Manager IP Edition. This publication is for advanced users who need to query the component databases directly.

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

- **IBM Tivoli Network Manager IP Edition Language Reference, SC27-2768-00**
  Describes the system languages used by IBM Tivoli Network Manager IP Edition, such as the Stitcher language, and the Object Query Language. This publication is for advanced users who need to customize the operation of IBM Tivoli Network Manager IP Edition.

- **IBM Tivoli Network Manager IP Edition Perl API Guide, SC27-2769-00**
  Describes the Perl modules that allow developers to write custom applications that interact with the IBM Tivoli Network Manager IP Edition. Examples of custom applications that developers can write include Polling and Discovery Agents. This publication is for advanced Perl developers who need to write such custom applications.

- **IBM Tivoli Monitoring for Tivoli Network Manager IP User’s Guide, SC27-2770-00**
  Provides information about installing and using IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition. This publication is for system administrators who install and use IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition to monitor and manage IBM Tivoli Network Manager IP Edition resources.

### Prerequisite publications

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

- **IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide, SC23-9680**
  Includes installation and upgrade procedures for Tivoli Netcool/OMNIbus, and describes how to configure security and component communications. The publication also includes examples of Tivoli Netcool/OMNIbus architectures and describes how to implement them.

  Provides an overview of the desktop tools and describes the operator tasks related to event management using these tools.

- **IBM Tivoli Netcool/OMNIbus Administration Guide, SC23-9681**
Describes how to perform administrative tasks using the Tivoli Netcool/OMNIbus Administrator GUI, command-line tools, and process control. The publication also contains descriptions and examples of ObjectServer SQL syntax and automations.

- **IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide, SC23-9684**
  Contains introductory and reference information about probes and gateways, including probe rules file syntax and gateway commands.

- **IBM Tivoli Netcool/OMNIbus Web GUI Administration and User’s Guide SC23-9682**
  Describes how to perform administrative and event visualization tasks using the Tivoli Netcool/OMNIbus Web GUI.

**Accessing terminology online**

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

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

**Accessing publications online**

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

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

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

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

**Ordering publications**

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


You can also order by telephone by calling one of these numbers:
- In the United States: 800-879-2755
- In Canada: 800-426-4968

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

1. Go to the following Web site:

2. Select your country from the list and click Go. The Welcome to the IBM Publications Center page is displayed for your country.

3. On the left side of the page, click **About this site** to see an information page that includes the telephone number of your local representative.
Accessibility

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

Accessibility features

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

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

The Network Manager Information Center is accessibility-enabled. The accessibility features of the information center are described in Accessibility and keyboard shortcuts in the information center.

Extra steps to configure Internet Explorer for accessibility

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

To enable high contrast mode, complete the following steps:
1. Click Tools > Internet Options > Accessibility.
2. Select all the check boxes in the Formatting section.

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

IBM® and accessibility

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

Tivoli technical training

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

http://www.ibm.com/software/tivoli/education
Support 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 http://www.ibm.com/software/support/probsub.html and follow the instructions.

IBM Support Assistant

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

Tivoli user groups

Tivoli user groups are independent, user-run membership organizations that provide Tivoli users with information to assist them in the implementation of Tivoli Software solutions. Through these groups, members can share information and learn from the knowledge and experience of other Tivoli users. Tivoli user groups include the following members and groups:

- 23,000+ members
- 144+ groups

Access the link for the Tivoli Users Group at www.tivoli-ug.org.

Service Management Connect

Access Service Management Connect at https://www.ibm.com/developerworks/servicemanagement/. Use Service Management Connect in the following ways:

- Become involved with transparent development, an ongoing, open engagement between other users and IBM developers of Tivoli products. You can access early designs, sprint demonstrations, product roadmaps, and prerelease code.
- Connect one-on-one with the experts to collaborate and network about Tivoli and the (enter your community name here) community.
- Read blogs to benefit from the expertise and experience of others.
- Use wikis and forums to collaborate with the broader user community.
Conventions used in this publication

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

**Typeface conventions**

This publication uses the following typeface conventions:

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

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

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

**Bold monospace**
- Command names, and names of macros and utilities that you can type as commands
- Environment variable names in text
- Keywords
- Parameter names in text: API structure parameters, command parameters and arguments, and configuration parameters
- Process names
- Registry variable names in text
- Script names
Operating system-dependent variables and paths

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

On UNIX systems, preface environment variables with the dollar sign $. For example, on UNIX, NCHOME is $NCHOME.
Chapter 1. ncp_class database

The ncp_class database enables the Active Object Class (AOC) manager, ncp_class, to manage AOCs.

The CLASS database consists of two main tables, activeClasses and staticClasses, which are populated at startup as CLASS reads the AOC definitions. The two tables represent two different views of the class hierarchy.

The ncp_class database is described in the following file.

$NCHOME/etc/precision/ClassSchema.cfg

Note: For information about ncp_class command-line options, see the IBM Tivoli Network Manager IP Edition Administration Guide.

class.activeClasses table

The class.activeClasses table holds the full definition of every Active Object Class (AOC).

The table below describes the activeClasses table.

Table 1. class.activeClasses Database Table Schema

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassName</td>
<td>PRIMARY KEY</td>
<td>Text</td>
<td>Specifies the unique name of the AOC.</td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNIQUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClassId</td>
<td>PRIMARY KEY</td>
<td>Integer</td>
<td>Specifies an identifier for the AOC.</td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SuperClass</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the name of the parent AOC.</td>
</tr>
<tr>
<td>Dictionary</td>
<td>Text</td>
<td>List of dictionaries.</td>
<td></td>
</tr>
<tr>
<td>Instantiate</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the rules for instantiating the AOC.</td>
</tr>
<tr>
<td>VisualIcon</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the icon associated with the AOC, which is displayed in a user interface.</td>
</tr>
<tr>
<td>MenuRules</td>
<td>List type object</td>
<td>List of executable rules.</td>
<td></td>
</tr>
<tr>
<td>Menu</td>
<td>List type object</td>
<td>Menu for the AOC.</td>
<td></td>
</tr>
<tr>
<td>ActionType</td>
<td>Externally defined actions data type</td>
<td>Integer</td>
<td>Specifies the list of actions to be taken.</td>
</tr>
</tbody>
</table>
class.staticClasses table

The class.staticClasses table holds the contents of a raw Active Object Class (AOC) as it is defined in the .aoc file.

The table below describes the class.staticClasses table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassName</td>
<td>PRIMARY KEY</td>
<td>Text</td>
<td>Specifies the unique name of the AOC.</td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNIQUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClassId</td>
<td>PRIMARY KEY</td>
<td>Integer</td>
<td>Specifies an identifier for the AOC.</td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SuperClass</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the name of the parent AOC.</td>
</tr>
<tr>
<td>Dictionary</td>
<td>NOT NULL</td>
<td>Text</td>
<td>List of dictionaries.</td>
</tr>
<tr>
<td>Instantiate</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the rules for instantiating the AOC.</td>
</tr>
<tr>
<td>Extensions</td>
<td>Externally defined extension data type</td>
<td>Object of extension data type</td>
<td>Specifies the list of extensions contained within the AOC.</td>
</tr>
<tr>
<td>VisualIcon</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the icon associated with the AOC.</td>
</tr>
<tr>
<td>MenuRules</td>
<td></td>
<td>List type object</td>
<td>List of executable rules.</td>
</tr>
<tr>
<td>Menu</td>
<td></td>
<td>List type object</td>
<td>Menu for the AOC.</td>
</tr>
<tr>
<td>ActionType</td>
<td>Externally defined actions data type</td>
<td>Integer</td>
<td>Specifies a list of actions to be taken.</td>
</tr>
</tbody>
</table>

class.classIds table

The class.classIds table stores the lookup values for class IDs from class name.

The table below describes the classIds table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassId</td>
<td>PRIMARY KEY</td>
<td>Integer</td>
<td>Specifies the numerical ID for the Active Object Class (AOC).</td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNIQUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column name</td>
<td>Constraints</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>ClassName</td>
<td>PRIMARY KEY</td>
<td>Text</td>
<td>Specifies the unique name of the AOC.</td>
</tr>
<tr>
<td></td>
<td>NOT NULL UNIQUE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 2. ncp_ctrl database

The ncp_ctrl database enables the master process controller, ncp_ctrl, to manage and monitor the status of Network Manager processes.

The ncp_ctrl database is defined in the \$NCHOME/etc/precision/CtrlSchema.cfg file.

The services.config Table

The services.config database table is an active table for managed processes.

This table is used for the ncp_ctrl process when it overwrites trace files. To enable and configure trace file rotation, do not change this database; use the log file rotation environment variables.

The table below describes the services.config database table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enableTraceFileRotation</td>
<td>NOT NULL</td>
<td>Integer</td>
<td>Specifies whether log file rotation is enabled.</td>
</tr>
</tbody>
</table>

The services.inTray Table

The services.inTray database table is an active table for managed processes.

Data is inserted into this table by ncp_ctrl when it reads its configuration file. Processes with an entry in this database are started by ncp_ctrl as managed processes. If an entry is removed from this table, it is stopped by ncp_ctrl. The ncp_ctrl process also monitors the status of processes named in the inTray table and restarts them if they are stopped.

The table below describes the services.inTray database table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>argList</td>
<td></td>
<td>List of type text</td>
<td>Specifies a list of arguments sent to the service.</td>
</tr>
<tr>
<td>binaryName</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Base name for the binary that implements the service. This is used in preference to the serviceName field to launch the service.</td>
</tr>
<tr>
<td>dependsOn</td>
<td></td>
<td>List of type text</td>
<td>Specifies a list of processes, prerequisites, required to run the current services.</td>
</tr>
<tr>
<td>domainName</td>
<td></td>
<td>Text</td>
<td>Specifies the domain under which the service is running.</td>
</tr>
<tr>
<td>hostName</td>
<td></td>
<td>Text</td>
<td>Specifies the name of the system upon which the service is running.</td>
</tr>
<tr>
<td>Column name</td>
<td>Constraints</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>interval</td>
<td></td>
<td>Integer</td>
<td>Specifies the average interval between heartbeat signals from the service.</td>
</tr>
<tr>
<td>logFile</td>
<td></td>
<td>Text</td>
<td>Specifies the name of file in which output is logged.</td>
</tr>
<tr>
<td>logLevel</td>
<td></td>
<td>Text</td>
<td>The logging level for the process. By default, this field has the value <code>warn</code>. The value in this table updates dynamically if the logging level is changed for the process on the command line. Changing this value in the table dynamically changes the logging level for the process even if it is already running.</td>
</tr>
<tr>
<td>processId</td>
<td></td>
<td>Long integer</td>
<td>Specifies the process ID of the service.</td>
</tr>
<tr>
<td>memUsage</td>
<td></td>
<td>Long integer</td>
<td>The native memory of the process.</td>
</tr>
<tr>
<td>retryCount</td>
<td></td>
<td>Integer</td>
<td>Specifies the number of times to attempt to restart a service. Note: Providing the value 0 (zero) means no attempt is made to restart a service when it stops, while -1 sets the service to always start again when it stops.</td>
</tr>
<tr>
<td>serviceld</td>
<td>PRIMARY KEY NOT NULL UNIQUE</td>
<td>Integer</td>
<td>Specifies the unique ID of the service and is assigned internally.</td>
</tr>
<tr>
<td>serviceKey</td>
<td>NOT NULL UNIQUE</td>
<td>Text</td>
<td>Auto-generated unique key for a process.</td>
</tr>
<tr>
<td>serviceName</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the name of the service to be managed.</td>
</tr>
<tr>
<td>servicePath</td>
<td></td>
<td>Text</td>
<td>Specifies the full path to the service, which might be NCHOME/precision/platform/$PLATFORM/bin if NULL.</td>
</tr>
</tbody>
</table>
Table 5. services.inTray Database Table Schema (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceState</td>
<td>Externally defined serviceState data type</td>
<td>Integer</td>
<td>Specifies an integer which reflects the current operational state of the service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 - The service is alive but currently idle.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - The service is waiting for its prerequisites, that is, waiting for its dependencies to be satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 - The service is waiting to begin sending heartbeats.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 - The application is currently starting, which is Fork service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 - The service is alive and running.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 - The service is not functioning correctly.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 - The service is stopped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 - The service has failed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8 - The service is shutdown.</td>
</tr>
<tr>
<td>traceLevel</td>
<td></td>
<td>Integer</td>
<td>The trace level for the process. By default, this field has the value 0 (zero). The value in this table updates dynamically if the trace level is changed for the process on the command line. Changing this value in the table dynamically changes the trace level for the process even if it is already running.</td>
</tr>
</tbody>
</table>

The services.slaveCtrl Table

The services.slaveCtrl table is populated automatically and serves as a reference for other instances of the ncp_ctrl process that are running in slave mode.

The table below describes the services.slaveCtrl database table.

Important: Do not use the OQL Service Provider to manually insert records into this table.

Table 6. services.slaveCtrl Database Table Schema

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slaveId</td>
<td>UNIQUE PRIMARY KEY NOT NULL</td>
<td>Integer</td>
<td>Specifies the unique ID number of the subordinate ncp_ctrl process running in slave mode.</td>
</tr>
</tbody>
</table>
Table 6. services.slaveCtrl Database Table Schema (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainName</td>
<td></td>
<td>Text</td>
<td>Specifies the domain under which the subordinate ncp_ctrl process is operating.</td>
</tr>
<tr>
<td>hostName</td>
<td></td>
<td>Text</td>
<td>Specifies the name of the system on which the subordinate ncp_ctrl process is running.</td>
</tr>
<tr>
<td>processId</td>
<td></td>
<td>Long integer</td>
<td>Specifies the process ID of the subordinate ncp_ctrl process.</td>
</tr>
<tr>
<td>slaveState</td>
<td>Externally defined serviceState data type</td>
<td>Integer</td>
<td>Specifies the current operational state of the subordinate process.</td>
</tr>
</tbody>
</table>

The services.unControlled Database Table

The services.unControlled table is a read-only table used to monitor uncontrolled services.

If the ncp_ctrl process receives a heartbeat from a service but does not start, the service is considered an uncontrolled service.

**Important:** Do not insert the information provided into this table. If you insert the information into the unControlled table, then your system might not function correctly.

The table below describes the services.unControlled database table.

Table 7. services.unControlledDatabase Table Schema

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceId</td>
<td>UNIQUE PRIMARY KEY NOT NULL</td>
<td>Integer</td>
<td>Specifies the unique ID of the service.</td>
</tr>
<tr>
<td>serviceName</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the name of the service.</td>
</tr>
<tr>
<td>domainName</td>
<td></td>
<td>Text</td>
<td>Specifies the domain under which the service is running.</td>
</tr>
<tr>
<td>hostName</td>
<td></td>
<td>Text</td>
<td>Specifies the name of the system on which the service is running.</td>
</tr>
<tr>
<td>processId</td>
<td></td>
<td>Long integer</td>
<td>Specifies the process ID of the service.</td>
</tr>
<tr>
<td>serviceState</td>
<td>Externally defined serviceState data type</td>
<td>Integer</td>
<td>Specifies an integer which reflects the current operational state of the service.</td>
</tr>
<tr>
<td>interval</td>
<td></td>
<td>Integer</td>
<td>Specifies the average interval between heartbeat signals from the service.</td>
</tr>
</tbody>
</table>
The services.unManaged Table

The services.unManaged table is used by the ncp_ctrl process to start and stop unmanaged processes. This table is also used by other Network Manager processes to instruct the ncp_ctrl process to start their subprocesses.

The ncp_disco process, for example, instructs the ncp_ctrl process to start the finders, helpers and agents by sending inserts into the services.unManaged table of the ncp_ctrl process. Inserting or deleting records from the table causes the corresponding processes to be started or stopped.

The table below describes the services.unManaged database table.

Table 8. services.unManaged Database Table Schema

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>argList</td>
<td>List of type Text</td>
<td>Specifies a list of arguments sent to the service process.</td>
<td></td>
</tr>
<tr>
<td>dependency</td>
<td>Long integer</td>
<td>Specifies the process ID of the parent that the service is dependent upon. Examine the content of this field to determine whether this unmanaged process is dependent or independent.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If this field contains a process ID value then this means that this is a dependent unmanaged process. A dependent unmanaged process is stopped by ncp_ctrl if the parent process dies. When this field is set then it contains the PID of the parent process. An example of dependent unmanaged processes are the discovery agents started by the parent Discovery engine, ncp_disco, process.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If this field contains a NULL value then this means that this is an independent unmanaged process. An independent unmanaged process continues to run if the parent process dies.</td>
<td></td>
</tr>
<tr>
<td>endSignal</td>
<td>Integer</td>
<td>The signal to be used to end the process. The default value is 9.</td>
<td></td>
</tr>
<tr>
<td>hostName</td>
<td>Text</td>
<td>Specifies the system on which the service is running.</td>
<td></td>
</tr>
<tr>
<td>logFile</td>
<td>Text</td>
<td>Specifies the name of the file in which output is logged.</td>
<td></td>
</tr>
<tr>
<td>processId</td>
<td>Long integer</td>
<td>Specifies the service process ID.</td>
<td></td>
</tr>
<tr>
<td>serviceId</td>
<td>UNIQUE PRIMARY KEY NOT NULL Integer</td>
<td>Specifies the unique ID of the service.</td>
<td></td>
</tr>
<tr>
<td>serviceName</td>
<td>NOT NULL Text</td>
<td>Specifies the name of the service.</td>
<td></td>
</tr>
</tbody>
</table>
Table 8. services.unManaged Database Table Schema (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| servicePath |             | Text      | Specifies the full path to the service process. If the value is set to NULL, then this default path is used: 
NCHOME/precision/platform/$PLATFORM/bin. |
| serviceState | Externally defined serviceState data type | Integer | Specifies an integer which reflects the current operational state of the service. |
Chapter 3. ncp_trapMux database

The ncp_trapMux database enables the SNMP Trap Multiplexer, ncp_trapMux, to forward traps to multiple ports and capture and replay traps.

The ncp_trapMux database is defined in the $NCHOME/etc/precision/TrapMuxSchema.cfg file.

**trapMux.command table**

The trapMux.command database table is an active table used to control the ncp_trapmux process.

The table below describes the trapMux.command table.

*Table 9. trapMux.command Database Table Schema*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the command to issue to the ncp_trapmux process.</td>
</tr>
<tr>
<td>fileName</td>
<td></td>
<td>Text</td>
<td>Specifies the file on which to perform the command, if applicable.</td>
</tr>
</tbody>
</table>

**trapMux.config table**

The trapMux.config table contains the main configuration data for the ncp_trapmux process.

The table below describes the trapMux.config table.

*Table 10. trapMux.config Database Table Schema*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td></td>
<td>Integer</td>
<td>Specifies the port on which to listen for traps.</td>
</tr>
</tbody>
</table>

**trapMux.sinkHosts table**

The trapMux.sinkHosts table contains details of the hosts to which traps are forwarded and the port numbers.

The table below describes the trapMux.sinkHosts table.

*Table 11. trapMux.sinkHosts Database Table Schema*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the host name or IP address to which traps are forwarded.</td>
</tr>
<tr>
<td>port</td>
<td>NOT NULL</td>
<td>Integer</td>
<td>Specifies the port number of the host name or IP address to which traps are forwarded.</td>
</tr>
</tbody>
</table>
Chapter 4. ncp_virtualdomain database

The ncp_virtualdomain database enables the virtual domain subsystem to support Network Manager failover.

The ncp_virtualdomain database uses two database tables: config and state. The health check status and filters are stored in these tables.

To configure the operation of virtual domains specify OQL inserts to the VirtualDomainSchema.cfg file. The primary server and the backup server each have a VirtualDomainSchema.cfg file. Ensure that you make the same changes to both configuration files.

config database schema

The config database schema is defined in the NCHOME/etc/precision/VirtualDomainSchema.cfg directory. The config database schema has one table: config.defaults.

The table below describes the defaults table. The defaults table holds the time periods for the failover checks.

Table 12. config.defaults table description

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| m_AutoTopologyDownload | NOT NULL    | Integer   | Specifies the network topology transferred from the primary server when the backup server starts. If this option is set, then the Network Manager server downloads the topology every time the TCP connection is lost and reestablished.  
  • 0 - Topology is only downloaded once when the Network Manager server is started.  
  • 1 - Topology is downloaded each time a new TCP connection is made. This is the default value. |
| m_FailoverTime          | NOT NULL    | Integer   | Specifies the maximum difference between the current time and the Health Check Resolution event timestamp. If the difference exceeds this value, then Network Manager fails over.  
  The default value is 300 seconds. |
| m_HealthCheckPeriod     | NOT NULL    | Integer   | Specifies the time period between each health check. The health check applies the filters in state.filters to the values in state.services (the current state of the processes monitored by the CTRL process).  
  The default value is 60 seconds. |
Table 12. config.defaults table description (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_SocketKeepAlivePeriod</td>
<td>NOT NULL</td>
<td>Integer</td>
<td>Specifies the time, in seconds, between messages that Virtual Domain sends to indicate that a connection is still active. If three of these messages are missed, the connection is considered inactive and Virtual Domain tries to open a new connection. By default this is 60, that is, one minute.</td>
</tr>
</tbody>
</table>

**state database schema**

The state database schema is defined in the NCHOME/etc/precision/VirtualDomainSchema.cfg directory. The state database schema has three tables: state.services, state.domains, and state.filters.

The services table holds the status of the processes monitored by the CTRL process. The table below describes the columns of the services table.

Table 13. state.services Table Descriptions

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_ArgList</td>
<td></td>
<td>List</td>
<td>Specifies a list of arguments sent to the service process.</td>
</tr>
<tr>
<td>m_ChangeTime</td>
<td>NOT NULL</td>
<td>Timestamp</td>
<td>Specifies a timestamp from the last time the status of this service was updated by the CTRL process.</td>
</tr>
<tr>
<td>m_CtrlState</td>
<td>NOT NULL</td>
<td>Integer</td>
<td>Specifies an integer which reflects the operational state of the service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 - The service is alive but idle.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 - The service is waiting for its prerequisites, that is, waiting for its dependencies to be satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2 - The service is waiting for a heartbeat from another service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3 - The application is starting, that is, Fork service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4 - The service is alive and running.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5 - The service is not functioning correctly.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6 - The service is stopped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 - The service failed.</td>
</tr>
<tr>
<td>m_Domain</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the domain name under which the service is running.</td>
</tr>
<tr>
<td>m_ExtraInfo</td>
<td></td>
<td>Vblist</td>
<td>Specifies additional information. The default value is empty.</td>
</tr>
<tr>
<td>m_Pid</td>
<td>NOT NULL</td>
<td>Integer</td>
<td>Specifies the process ID for the component.</td>
</tr>
<tr>
<td>m_ServiceName</td>
<td>PRIMARY KEY, NOT NULL</td>
<td>Text</td>
<td>Specifies the service name of the component monitored by the CTRL process.</td>
</tr>
</tbody>
</table>
The domains table holds the status of the primary and backup domains in the failover architecture. This table always contains an entry for the primary server and the backup server.

The table below describes the columns of the domains table.

*Table 14. state.domains Table Descriptions*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| m_ActingPrimary | NOT NULL    | Integer   | Specifies whether the Network Manager server is acting as the primary server and is monitoring the network.  
   • 0 - Not acting as the primary server  
   • 1 - Acting as the primary server |
| m_Backup      | NOT NULL    | Integer   | Specifies whether the server is configured as the backup server. This value is automatically set by the configuration defined in the $NCHOME/etc/precision/ConfigItnm.DOMAIN.cfg file, or by inclusion of the -primaryDomain command-line option on components started by the CTRL process.  
   • 0 - Not configured as the backup server  
   • 1 - Configured as the backup server |
| m_ChangeTime  | NOT NULL    | Long      | Specifies the timestamp from the last time the status of the domain was updated by the Event Gateway. |
| m_Domain      | NOT NULL    | Text      | Specifies the domain in which this installation of Network Manager is running. The domain name must be different from the names of the primary and backup servers. |
| m_HealthStatus| NOT NULL    | Integer   | Specifies the status of the Health Check events.  
   • 0 - Unhealthy. The Network Manager server generated a Health Check Problem event or the existing Health Check Resolution event exceeded the m_failover time period.  
   • 1 - Healthy. The Network Manager server generated a Health Check Resolution event within the m_failover time period. |

The filters table contains the filters that are to be applied to the values in the state.services table. The table below describes the columns of the filters table.

*Table 15. state.filters Table Descriptions*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_ServiceName</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the unique name of the service to which the filter is applied.</td>
</tr>
<tr>
<td>m_Filter</td>
<td>NOT NULL</td>
<td>Text</td>
<td>Specifies the OQL filter to apply.</td>
</tr>
</tbody>
</table>
Table 15. *state.filters* Table Descriptions (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Constraints</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_Description</td>
<td>Text</td>
<td>Specifies a description of the filter operation.</td>
<td></td>
</tr>
</tbody>
</table>

Network Manager provides a default filter for the following components:

There is one filter for each component except for the ncp_poller component, which instead can have a filter for each poller defined. Each filter checks that the m_CtrlState value is not set to 7 (the service has not failed), and that the timestamp m_ChangeTime is not older than 300 seconds.

Default filters for the *state.filters* table are provided for the components below.
- Polling engine, ncp_poller: multiple filters can be defined, one for each poller defined in the *CtrlServices.cfg* file.
- Event Gateway, ncp_g_event
- Topology manager, ncp_model

**Example Virtual Domain configuration**

Use these examples to familiarize yourself with the default OQL inserts that are appended to the *VirtualDomainSchema.cfg* file to configure Virtual Domain when it is launched.

**Remember:** Both the primary server and backup server have a *VirtualDomainSchema.cfg* configuration file. Ensure that you make identical changes to both configuration files.

**Example configuration of the *config.defaults* table**

The OQL sample below configures the *config.defaults* table.

```sql
insert into config.defaults
(
    m_HealthCheckPeriod,
    m_FailoverTime,
    m_AutoTopologyDownload
)
values
(60, 300, 1);
```

This insert triggers the following behavior:
- Virtual Domain performs a health check of each of the processes listed in the *state.services* table every 60 seconds. The health check applies the filters in the *state.filters* table to the values in *state.services* (the current state of the processes being monitored by the CTRL process).
- Virtual Domain has a default time difference of 300 seconds between the current time and the Health Check Resolution event timestamp. When the difference exceeds this value Network Manager fails over. The time difference is set to this value to avoid spurious failover.
- The backup server downloads the topology every time the TCP connection is lost and remade with the primary server.
Example configuration of the state.filters table: Event Gateway filter

The following sample OQL insert specifies one of the default filters provided with Network Manager.

```
insert into state.filters
(
    m_ServiceName,
    m_Filter,
    m_Description
)
values
(
    "ncp_g_event",
    "m_ChangeTime > eval(long,'$TIME - 300') and m_CtrlState <> 7",
    "The Gateway has been running within the last 300 seconds"
);
```

This insert triggers the following behavior:

- Virtual Domain applies the filter (to be specified later in the insert) against the Event Gateway process, ncp_g_event.
- Virtual Domain applies a filter that checks that the m_CtrlState value is not equal to 7 (the service has not failed), and that the time stamp m_ChangeTime is not older than 300 seconds.
- Virtual Domain outputs the following description of the filter: The Gateway has been running within the last 300 seconds.

Example configuration of the state.filters table: filter for additionally configured poller

Additionally configured pollers can trigger failover. The following sample OQL insert specifies the commented out example filter for an additionally configured poller provided in the default version of the VirtualDomainSchema.cfg file. Remove the slashes in order to uncomment this filter. The m_ServiceName setting in the filter below must match the serviceName column specified for the additionally configured poller in the $NCHOME/etc/precision/CtrlServices.cfg file.

```
// insert into state.filters
// (
//    m_ServiceName,
//    m_Filter,
//    m_Description
// )
// values
// (
//    "PingPoller",
//    "m_ChangeTime > eval(time,'$TIME - 300') and m_CtrlState <> 7",
//    "The Poller has been running within the last 300 seconds"
// );
```

This insert triggers the following behavior:

- Virtual Domain applies the filter (to be specified later in the insert) against the additionally configured poller, PingPoller.
- Virtual Domain applies a filter that checks that the m_CtrlState value is not equal to 7 (the service has not failed), and that the time stamp m_ChangeTime is not older than 300 seconds.
- Virtual Domain outputs the following description of the filter: The Poller has been running within the last 300 seconds.
Appendix. Network Manager glossary

Use this information to understand terminology relevant to the Network Manager product.

The following list provides explanations for Network Manager terminology.

**AOC files**
Files used by the Active Object Class manager, `ncp_class` to classify network devices following a discovery. Device classification is defined in AOC files by using a set of filters on the object ID and other device MIB parameters.

**active object class (AOC)**
An element in the predefined hierarchical topology of network devices used by the Active Object Class manager, `ncp_class`, to classify discovered devices following a discovery.

**agent**
See, discovery agent.

**bookmark**
See, network view bookmark.

**class hierarchy**
Predefined hierarchical topology of network devices used by the Active Object Class manager, `ncp_class`, to classify discovered devices following a discovery.

**configuration files**
Each Network Manager process has one or more configuration files used to control process behaviour by setting values in the process databases. Configuration files can also be made domain-specific.

**discovery agent**
Piece of code that runs during a discovery and retrieves detailed information from discovered devices.

**Discovery Configuration GUI**
GUI used to configure discovery parameters.

**Discovery engine (ncp_disco)**
Network Manager process that performs network discovery.

**discovery phase**
A network discovery is divided into four phases: Interrogating devices, Resolving addresses, Downloading connections, and Correlating connectivity.

**discovery seed**
One or more devices from which the discovery starts.

**discovery scope**
The boundaries of a discovery, expressed as one or more subnets and netmasks.

**Discovery Status GUI**
GUI used to launch and monitor a running discovery.

**discovery stitcher**
Piece of code used during the discovery process. There are various
discovery stitchers, and they can be grouped into two types: data collection
stitchers, which transfer data between databases during the data collection
phases of a discovery, and data processing stitchers, which build the
network topology during the data processing phase.

dNCIM database
The dNCIM is a relational database embedded into the Discovery engine,
ncp_disco, and it stores the containment model that is derived from the
fullTopology database (and created by stitchers). This is the version of
the topology that is sent to the Topology manager, ncp_model. The dNCIM
database performs the same function as the scratchTopology database did
in previous versions of Network Manager.

domain
See, network domain

t entity
A topology database concept. All devices and device components
discovered by Network Manager are entities. Also device collections such
as VPNs and VLANs, as well as pieces of topology that form a complex
connection, are entities.

event enrichment
The process of adding topology information to the event.

Event Gateway (ncp_g_event)
Network Manager process that performs event enrichment.

Event Gateway stitcher
Stitchers that perform topology lookup as part of the event enrichment
process.

failover
In your Network Manager environment, a failover architecture can be used
to configure your system for high availability, minimizing the impact of
computer or network failure.

Failover plug-in
Receives Network Manager health check events from the Event Gateway
and passes these events to the Virtual Domain process, which decides
whether or not to initiate failover based on the event.

Fault Finding View
Composite GUI view consisting of an Active Event List (AEL) portlet
above and a Network Hop View portlet below. Use the Fault Finding View
to monitor network events.

full discovery
A discovery run with a large scope, intended to discover all of the network
devices that you want to manage. Full discoveries are usually just called
discoveries, unless they are being contrasted with partial discoveries. See
also, partial discovery

message broker
Component that manages communication between Network Manager
processes. The message broker used by Network Manager is called Really
Small Message Broker. To ensure correct operation of Network Manager,
Really Small Message Broker must be running at all times.

NCIM database
Relational database that stores topology data, as well as administrative
data such as data associated with poll policies and definitions, and
performance data from devices.
ncp_disco
See, Discovery engine.

ncp_g_event
See, Event Gateway.

ncp_model
See, Topology manager.

ncp_poller
See, Polling engine.

network domain
A collection of network entities to be discovered and managed. A single Network Manager installation can manage multiple network domains.

Network Health View
Composite GUI view consisting of a Network Views portlet above and an Active Event List (AEL) portlet below. Use the Network Health View to display events on network devices.

Network Hop View
Network visualization GUI. Use the Network Hop View to search the network for a specific device and display a specified network device. You can also use the Network Hop View as a starting point for network troubleshooting. Formerly known as the Hop View.

Network Polling GUI
Administrator GUI. Enables definition of poll policies and poll definitions.

Network Views
Network visualization GUI that shows hierarchically organized views of a discovered network. Use the Network Views to view the results of a discovery and to troubleshoot network problems.

network view bookmark
Network view bookmarks group together just those network views that you or your team need to monitor. Create new bookmarks or change existing bookmarks to help network operators visualize just those devices that they need to monitor.

OQL databases
Network Manager processes store configuration, management and operational information in OQL databases.

OQL language
Version of the Structured Query Language (SQL) that has been designed for use in Network Manager. Network Manager processes create and interact with their databases using OQL.

partial discovery
A subsequent rediscovery of a section of the previously discovered network. The section of the network is usually defined using a discovery scope consisting of either an address range, a single device, or a group of devices. A partial discovery relies on the results of the last full discovery, and can only be run if the Discovery engine, ncp_disco, has not been stopped since the last full discovery. See also, full discovery.

Path Views
Network visualization GUI that displays devices and links that make up a
network path between two selected devices. Create new path views or change existing path views to help network operators visualize network paths.

**performance data**
Performance data can be gathered using performance reports. These reports allow you to view any historical performance data that has been collected by the monitoring system for diagnostic purposes.

**Polling engine (ncp_poller)**
Network Manager process that polls target devices and interfaces. The Polling engine also collects performance data from polled devices.

**poll definition**
Defines how to poll a network device or interface and further filter the target devices or interfaces.

**poll policy**
Defines which devices to poll. Also defines other attributes of a poll such as poll frequency.

**Probe for Tivoli Netcool/OMNIbus (nco_p_ncpmonitor)**
Acquires and processes the events that are generated by Network Manager polls and processes, and forwards these events to the ObjectServer.

**RCA plug-in**
Based on data in the event and based on the discovered topology, attempts to identify events that are caused by or cause other events using rules coded in RCA stitchers.

**RCA stitcher**
Stitches that process a trigger event as it passes through the RCA plug-in.

**root-cause analysis (RCA)**
The process of determining the root cause of one or more device alerts.

**SNMP MIB Browser**
GUI that retrieves MIB variable information from network devices to support diagnosis of network problems.

**SNMP MIB Grapher**
GUI that displays a real-time graph of MIB variables for a device and uses the graph for fault analysis and resolution of network problems.

**stitcher**
Code used in the following processes: discovery, event enrichment, and root-cause analysis. See also [discovery stitcher](#), [Event Gateway stitcher](#), and [RCA stitcher](#).

**Structure Browser**
GUI that enables you to investigate the health of device components in order to isolate faults within a network device.

**Topology Manager (ncp_model)**
Stores the topology data following a discovery and sends the topology data to the NCIM topology database where it can be queried using SQL.

**WebTools**
Specialized data retrieval tools that retrieve data from network devices and can be launched from the network visualization GUIs, Network Views and Network Hop View, or by specifying a URL in a web browser.
Notices

This information applies to the PDF documentation set for IBM Tivoli Network Manager IP Edition.

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, Nihonbash-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
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 16 are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Table 16. IBM trademarks

<table>
<thead>
<tr>
<th>AIX®</th>
<th>Informix®</th>
<th>RDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNT</td>
<td>iSeries®</td>
<td>solidDB®</td>
</tr>
<tr>
<td>ClearQuest®</td>
<td>Lotus®</td>
<td>System p</td>
</tr>
<tr>
<td>Cognos®</td>
<td>Netcool®</td>
<td>System z®</td>
</tr>
<tr>
<td>DB2®</td>
<td>NetView®</td>
<td>Tivoli®</td>
</tr>
<tr>
<td>DB2 Universal Database</td>
<td>Notes®</td>
<td>WebSphere®</td>
</tr>
<tr>
<td>developerWorks®</td>
<td>OMEGAMON®</td>
<td>z/OS®</td>
</tr>
<tr>
<td>DS8000</td>
<td>PowerPC</td>
<td>z/VM®</td>
</tr>
<tr>
<td>Enterprise Storage Server®</td>
<td>PowerVM®</td>
<td>zSeries</td>
</tr>
<tr>
<td>IBM</td>
<td>PR/SM™</td>
<td></td>
</tr>
</tbody>
</table>

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.
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.

# Index

## A
- accessibility ix
- activeClasses database 1
- audience v

## C
- class.staticClasses database 2
- config.defaults database 13
- conventions, typeface xi

## D
- databases
  - activeClasses 1
  - class.staticClasses 2
  - classIds 2
  - config.defaults 13
  - gateway 1
  - ncp_class 1
  - ncp_ctrl 5
  - ncp_virtualdomain 13
  - services.config 5
  - services.inTray 5
  - services.slaveCtrl 7
  - services.unControlled 8
  - services.unManaged 9
  - state 14
  - TrapMux 11
  - trapMux.command 11
  - trapMux.config 11
  - trapMux.sinkHosts 11

## E
- education
  - see Tivoli technical training ix
- environment variables, notation xi

## G
- glossary 19

## M
- manuals vi

## N
- ncp_class 1
- ncp_ctrl database 5
- ncp_virtualdomain database 13
- Network Manager glossary 19

## O
- online publications vi
- ordering publications vi

## P
- publications vi

## S
- service x
- service management connect x
- services.config database 5
- services.inTray database 5
- services.slaveCtrl database 7
- services.unControlled database 8
- services.unManaged database 9
- SMC x
- state database 14
- support x
- support information x

## T
- Tivoli software information center vi
- Tivoli technical training ix
- training, Tivoli technical ix
- TrapMux database 11
- trapMux.config database 11
- trapMux.command database 11
- trapMux.sinkHosts database 11
- typeface conventions xi

## V
- variables, notation for xi
- virtual domain configuration 16

## W
- What this publication contains v