

Course Guide

Tivoli Network Manager 4.2 Operations and Administration

Course code TN325 ERC 2.0



July 2017 edition

Notices

This information was developed for products and services offered in the US.

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, MD-NC119
Armonk, NY 10504-1785
United States of America*

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

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 actual people or business enterprises is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

© Copyright International Business Machines Corporation 2017.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Unit 1. Introduction to Netcool Operations Insight	1-1
Introduction to Netcool Operations Insight	1-1
Unit objectives	1-2
Realize the benefits of Netcool Operations Insight	1-3
Netcool Operations Insight base features	1-4
1.1. Base solution components	1-5
Base solution components	1-5
Comprehensive dashboards	1-6
Event Viewer in Netcool Operations Insight	1-7
Optional network management components (1 of 5)	1-8
Optional network management components (2 of 5)	1-9
Optional network management components (3 of 5)	1-10
Optional network management components (4 of 5)	1-11
Optional network management components (5 of 5)	1-12
Topology search (1 of 2)	1-13
Topology search (2 of 2)	1-14
Network Health Dashboard	1-15
Network Performance Insight	1-16
Alert Notification	1-17
Three types of runbook automation	1-18
ITIL: How to handle technology problems	1-19
Problem determination (ITIL Step 1)	1-20
OMNIbus ObjectServer	1-21
OMNIbus probe operation and integration	1-22
Tivoli Network Manager	1-23
Information lookup (ITIL Step 2)	1-24
Basic event enrichment (1 of 5)	1-25
Basic event enrichment (2 of 5)	1-26
Basic event enrichment (3 of 5)	1-27
Basic event enrichment (4 of 5)	1-28
Basic event enrichment (5 of 5)	1-29
Problem diagnosis (ITIL Step 4)	1-30
Topological root cause analysis	1-31
Event search	1-32
Event isolation and correlation	1-33
Event Analytics for seasonal event identification	1-34
Event Analytics: Seasonality	1-35
Event Analytics for seasonal event identification	1-36
Related events	1-37
Event Analytics: Related events	1-38
Event Analytics dashboard	1-39
Problem bypass and recovery (ITIL Step 5)	1-40
Problem resolution (ITIL Step 6)	1-41
Tivoli Netcool/Impact Maintenance Manager	1-42
The need for Tivoli Netcool Configuration Manager	1-43
View configuration reports to see recent changes and their success or failure	1-44
Network Health Dashboard shows time of configuration changes and alarms	1-45
Use Runbook automations for consistent resolutions	1-46
Problem tracking and control (ITIL Step 7)	1-47
IBM Connections integration	1-48
Exercise: Review of Netcool Operations Insight architecture	1-49

Unit summary	1-50
Unit 2. Tivoli Network Manager 4.2: Introduction and architecture.....	2-1
Tivoli Network Manager 4.2 Introduction and architecture	2-1
Unit objectives	2-2
2.1. Features overview	2-3
Features overview	2-3
Key functions	2-4
OSI model	2-5
New features in 4.2 (1 of 2)	2-6
New features in 4.2 (2 of 2)	2-7
Platform and browser support	2-8
Database support	2-9
Prerequisite software for integration	2-10
Default environment variables in \$NCHOME/env.sh	2-11
Topic summary	2-12
2.2. Architectural overview.....	2-13
Architectural overview	2-13
Three architecture layers	2-14
Internal components (1 of 3)	2-16
Internal components (2 of 3)	2-17
Internal components (3 of 3)	2-18
Helper server components	2-19
External components (1 of 2)	2-20
External components (2 of 2)	2-21
Topic summary	2-22
2.3. Troubleshooting with log and trace files	2-23
Troubleshooting with log and trace files	2-23
Troubleshooting with log files	2-24
Set logging levels in CtrlServices.cfg	2-25
Dynamic updating of logging levels with a signal	2-26
Example of dynamic logging level updates	2-27
Updating services.inTray to set levels	2-28
services.inTray logging detail parameters	2-29
Topic summary	2-30
2.4. Architectural consideration	2-31
Architectural consideration	2-31
Installation objective	2-32
Sizing considerations	2-33
Multiple domains	2-34
Other architectural considerations	2-35
Planning deployments	2-36
Demonstration or educational deployment (1 of 2)	2-40
Demonstration or educational deployment (2 of 2)	2-41
Small network description	2-42
Small network deployment with redundancy	2-43
Medium network description	2-44
Medium network deployment	2-45
Large network description	2-46
Large network deployment	2-47
Large network deployment example	2-48
Very large network description	2-49
Very large network deployment	2-50
Single-domain ObjectServers	2-51
Multiple-domain ObjectServers	2-52
NmosDomainName	2-53

Select the correct domain name	2-54
Topic summary	2-55
Review questions	2-56
Unit summary	2-57
Review answers	2-58
Unit 3. Discovery basics	3-1
Discovery basics	3-1
Unit objectives	3-2
3.1. How discovery works	3-3
How discovery works	3-3
Discovery basics	3-4
Discovery is the key	3-5
What you can discover	3-6
Discovery protocols	3-7
Discovery terminology and architecture	3-8
Step 1: Finders	3-9
Finder options with either ping finder or file finder	3-10
Finder options with both ping finder and file finder	3-11
Step 2a: Details agent	3-12
Step 2b: Other agents	3-13
Step 3: Stitchers (1 of 2)	3-15
Step 3: Stitchers (2 of 2)	3-16
Step 4: Stitchers send topology to Model	3-17
Discovery Phases 0 and 1	3-18
The BlackoutState and Phase 1	3-19
Discovery Phases 2 – 4	3-20
Topic summary	3-21
3.2. The discovery wizard	3-22
The discovery wizard	3-22
Using the discovery wizard	3-23
Log on to IBM Dashboard Application Services Hub	3-24
Configuring discovery	3-25
Preparing for discovery	3-26
Starting the discovery configuration wizard	3-27
Choosing a scoped or unscoped discovery	3-28
Scope limits the extent of discovery	3-29
Discovery filters and exclusions	3-30
Out-of-scope interfaces	3-31
SNMP community strings	3-32
Processing SNMP community strings	3-34
SNMP password properties	3-35
Telnet access configuration	3-36
Telnet passwords	3-37
Discovery agents	3-38
VLAN modeling	3-39
VLAN considerations	3-40
Prediscovery filters	3-41
Discovery speed versus accuracy	3-42
Network reliability	3-43
Configuration summary	3-44
Topic summary	3-45
3.3. Starting discovery	3-46
Starting discovery	3-46
Starting full and partial discoveries	3-47
Start a full discovery from the GUI	3-48

Start a partial discovery from the GUI	3-49
Start scheduled full discoveries	3-50
Start a triggered rediscovery	3-52
Considerations for a triggered discovery	3-53
3.4. Checking discovery status	3-54
Checking discovery status	3-54
Checking discovery phase status	3-55
Checking discovery agent status	3-56
Checking individual agent status	3-57
Checking ping finder status	3-58
Topic summary	3-59
3.5. Discovery configuration tabs.	3-60
Discovery configuration tabs	3-60
Using discovery tabs	3-61
Modifying scope	3-62
Modifying seeds	3-63
Modifying community strings	3-64
Selecting agents	3-65
Agents	3-66
Domain Name Service (DNS)	3-67
Network Address translation (NAT)	3-69
Advanced discovery configuration	3-71
Advanced ping finder configuration	3-72
Advanced SNMP helper configuration	3-73
Advanced DNS helper configuration	3-74
Enable Feedback option on Advanced tab	3-75
Ping verification options	3-76
Allow Virtual option on Advanced tab	3-77
Other advanced discovery options	3-78
Topic summary	3-79
Exercise: Discovery basics	3-80
Review questions	3-81
Unit summary	3-82
Review answers	3-83
Unit 4. Advanced discovery options.	4-1
Advanced discovery options	4-1
Unit objectives	4-3
4.1. Configure discovery of multicast networks	4-4
Configure discovery of multicast networks	4-4
Multicasting	4-5
Unicasting	4-7
Reasons for using multicasting	4-8
Multicasting discovery features	4-9
Multicast discovery agents	4-10
Multicast discovery stitchers	4-11
IP multicast configuration tasks	4-12
Configure multicast groups	4-13
Configure multicast sources	4-15
Usage: IP multicast views	4-16
IP multicast integration	4-17
IGMP group membership view	4-18
Troubleshooting multicast discovery	4-19
Topic summary	4-20
4.2. Discovery filtering	4-21
Discovery filtering	4-21

Scenario 1: College campus classroom	4-22
Scenario 2: Dynamic dial backup interfaces	4-23
Scenario 2: Use scoping solution	4-24
Prediscovery filters	4-25
Creating prediscovery filters (GUI)	4-27
Interface filtering	4-29
Configuring SNMP interface filters	4-30
Interface filter syntax	4-31
When to use SNMP interface filters	4-32
How interface filtering works	4-33
Topic summary	4-34
4.3. Discovering with collectors	4-35
Discovering with collectors	4-35
Overview of EMS integration	4-36
Definition of a collector	4-37
Default Java collectors	4-38
Default Perl collectors	4-39
EMS integration components	4-40
Configuring an EMS discovery	4-41
How to configure a collector	4-42
EMS integration collector configuration	4-43
EMS integration (1 of 2)	4-44
EMS integration (2 of 2)	4-45
Running EMS integration collectors	4-46
Collector finder	4-47
Collector agents	4-48
The database finder	4-49
Topic summary	4-51
4.4. Troubleshooting discovery	4-52
Troubleshooting discovery	4-52
Discovery takes a long time	4-53
Discovery does not start	4-54
Other discovery issues	4-55
Uses for discovery table caching	4-56
Enable caching of discovery tables	4-57
The discovery cache files	4-58
Discovery agent and stitcher data	4-59
Data to send to technical support	4-60
Useful commands (1 of 2)	4-61
Useful commands (2 of 2)	4-62
Restarting discovery with an empty database	4-64
Topic summary	4-65
4.5. Optimizing discovery	4-66
Optimizing discovery	4-66
Tweaking discovery	4-67
Reducing Phase 1 discovery time with seed file	4-68
Discovery tips (1 of 5)	4-69
Discovery tips (2 of 5)	4-70
Discovery tips (3 of 5)	4-71
Discovery tips (4 of 5)	4-72
Discovery tips (5 of 5)	4-73
Poor ping sweep seeding	4-75
Efficient ping sweep seeding	4-76
Exercise: Advanced discovery options	4-77
Review questions	4-78
Unit summary	4-79

Review answers	4-80
Unit 5. Topology visualization basics.....	5-1
Topology visualization basics	5-1
Unit objectives	5-2
5.1. Visualization architecture	5-3
Visualization architecture	5-3
Dashboard Application Services Hub	5-4
Topoviz architecture	5-5
5.2. Event views.....	5-6
Event views	5-6
Event Dashboard	5-7
Active Event List (AEL)	5-8
Event Viewer	5-9
Key Performance Indicators	5-10
Netcool Health KPI	5-11
Creating an event filter (1 of 2)	5-12
Creating an event filter (2 of 2)	5-13
Create a view (1 of 2)	5-14
Create a view (2 of 2)	5-15
New view is available	5-16
Show root cause event from symptom event	5-17
Show suppressed events from root cause event	5-18
Fault Finding View	5-19
Network Health View	5-20
Topology visualization icons	5-21
Find in Hop View from event list	5-23
Find in Network View from event list	5-24
5.3. The Hop View	5-25
The Hop View	5-25
Hop View	5-26
Search for a device	5-27
Graph layout icons	5-28
Zoom features	5-29
Find in map	5-30
Connectivity views	5-31
Device icons reflect event severity	5-32
Network table layout	5-33
BGP visualization enhancements	5-34
OSPF visualization enhancements	5-35
5.4. Network views.....	5-36
Network views	5-36
Default network views (1 of 3)	5-37
Default network views (2 of 3)	5-39
Default network views (3 of 3)	5-40
Tree table search	5-41
Network view bookmarks	5-42
Creating a container	5-43
Customizable network view icons (1 of 2)	5-44
Customizable network view icons (2 of 2)	5-45
Standard versus dynamic network views	5-46
Standard network views	5-47
Creating a standard network view	5-48
Dynamic network views	5-49
Creating a dynamic network view	5-50
Creating a filtered view	5-51

Subnet view	5-52
IP filter views (1 of 2)	5-53
IP filter views (2 of 2)	5-54
Unacknowledged events by severity	5-55
All routers	5-56
All switches	5-57
IGMP groups view	5-58
Multicast routing distribution trees	5-59
Protocol-independent multicast networks	5-60
Virtual Private Label Switching	5-61
Manually created devices	5-62
HSRP groups	5-63
MPLS Core and VPNs	5-64
5.5. Link status and capacity	5-65
Link status and capacity	5-65
Link status and capacity	5-66
Representing link status	5-68
Link status display	5-70
Specify link status options on a network view	5-71
Representing link capacity	5-72
Link capacity	5-73
Link tooltips	5-74
Contextual link tools	5-75
Contextual tools on link bundles	5-76
Show link events tool	5-77
Tabular layout option	5-78
Standard right-click options apply to tabular view	5-79
Tabular layout overview (1 of 2)	5-80
Tabular layout overview (2 of 2)	5-81
Limitations	5-82
Troubleshooting tabular views	5-83
5.6. Tools	5-84
Tools	5-84
Topology visualization tools (1 of 3)	5-85
Topology visualization tools (2 of 3)	5-86
Topology visualization tools (3 of 3)	5-87
Webtools	5-88
Polling and MIB Info tool	5-89
Browse SNMP MIB data	5-90
Ping tool	5-91
Exercise: Topology visualization basics	5-92
Review questions	5-93
Unit summary	5-94
Review answers	5-95
Unit 6. Advanced visualization	6-1
Advanced visualization	6-1
Unit objectives	6-2
6.1. Path views	6-3
Path views	6-3
Path views overview	6-4
MPLS TE paths	6-5
User-defined paths (1 of 2)	6-6
Create a path view from a Hop or Network View	6-7
Create a Path View from the menu	6-8
Specify nodes and gateway	6-9

User-defined paths (2 of 2)	6-10
Path View administration	6-11
6.2. Topology editing tools.	6-12
Topology editing tools	6-12
Graphical Topology Editor	6-13
Use cases for the Topology Editor	6-14
Add a device with the Topology Editor	6-15
Use the Add Device Wizard	6-16
Confirm device details	6-17
Manually Added Devices network view	6-18
Delete Device	6-19
Add connection	6-20
Add a network connection	6-21
Specify connection details	6-22
View added devices and connections	6-23
Troubleshooting the Topology Editor	6-24
6.3. Device Structure browser	6-25
Device Structure browser	6-25
Device Structure browser description	6-26
Device Structure browser	6-27
Structure browser tree	6-28
Collapsing and expanding structure browser tree	6-29
Structure browser basic search	6-30
Device Structure browser search function	6-31
Structure browser troubleshooting	6-32
Device Structure browser BGP support	6-33
Device Structure browser BGP protocol endpoints	6-34
Device Structure browser OSPF support	6-35
Structure browser OSPF protocol endpoints	6-36
Device Structure browser OSPF	6-37
Device Structure browser VLAN support	6-38
Device Structure browser VRF support	6-39
Device Structure browser support for pseudowire	6-40
Device Structure browser data flow	6-41
Device Structure browser tables (1 of 2)	6-42
Device Structure browser tables (2 of 2)	6-43
Network view bookmarks	6-44
6.4. Adding classes and icons	6-45
Adding classes and icons	6-45
Adding a device class (1 of 2)	6-46
Adding a device class (2 of 2)	6-47
Adding icons	6-48
6.5. Cross-domain network views	6-49
Cross-domain network views	6-49
Cross-domain processing and visualization	6-50
Prepare configuration files	6-51
Enable cross-domain processing	6-52
Cross-domain processing stitchers	6-53
Create default views for the new domain (1 of 2)	6-54
Create default views for the new domain (2 of 2)	6-55
Create the AGGREGATION_VIEW	6-56
Cross-domain Layer 2 view	6-57
Cross-domain Layer 3 view	6-58
Aggregation domain is for visualization only	6-59
Topic summary	6-60
Exercise: Advanced visualization	6-61

Review questions	6-62
Unit summary	6-63
Review answers	6-64
Unit 7. The Network Health Dashboard	7-1
The Network Health Dashboard	7-1
Unit objectives	7-2
Network Health Dashboard	7-3
Select a network view	7-4
Resource availability for selected network view	7-5
Top Performers listing for selected network view (1 of 2)	7-6
Top Performers listing for selected network view (2 of 2)	7-7
Selecting component in a frame changes event data frame	7-8
Do events correlate to configuration changes?	7-9
Create custom dashboards for status views of critical services	7-10
Dashboards can use event, topology, and log data	7-11
Exercise: Using the Network Health Dashboard	7-12
Unit summary	7-13
Unit 8. Monitoring and polling network devices	8-1
Monitoring and polling network devices	8-1
Unit objectives	8-2
8.1. Polling architecture and principles	8-3
Polling architecture and principles	8-3
Monitoring architecture	8-4
NCMONITOR database	8-5
How new polls affect the gateway	8-6
Polling policies	8-7
Multiple poll definitions per policy (1 of 3)	8-8
Multiple poll definitions per policy (2 of 3)	8-9
Multiple poll definitions per policy (3 of 3)	8-10
Polling policy and definition update frequency	8-11
Chassis and interface polling	8-12
Remote ping polling	8-13
Initiating the remote ping operation	8-14
Ping enhancements	8-15
Topic summary	8-16
8.2. Polling policies	8-17
Polling policies	8-17
Polling GUI (1 of 2)	8-18
Polling GUI (2 of 2)	8-19
Default ping polling policies	8-20
Default SNMP and remote ping polling policies	8-21
Poll Configuration Wizard	8-22
Enter poll policy details	8-23
Ways to configure scope	8-24
Deciding what gets polled	8-25
Network Views filter	8-26
Benefits of applying polling policies to Network Views	8-27
Assigning polling policies to device classes	8-28
Device Filter tab	8-29
Policy throttle	8-30
Assign a distributed poller to the policy	8-31
Verify the polling policy configuration	8-33
Device membership link	8-34
Enhanced poll policy status	8-35

Topic summary	8-36
8.3. Polling definitions	8-37
Polling definitions	8-37
Polling definition	8-38
Poll definition types	8-39
Select poll definition type	8-40
Set general polling properties	8-41
Set Poll Data Definition	8-42
Set polling thresholds and event messages	8-43
Interface filtering in poll definition properties	8-44
Poll definitions table	8-45
Cross-check to see impact of definition deletion	8-46
Adding new MIBs	8-47
Topic summary	8-49
8.4. Manage and unmanage devices and interfaces	8-50
Manage and unmanage devices and interfaces	8-50
NmOsManagedStatus values	8-51
Manage and unmanage tools	8-52
Raising events on unmanaged devices	8-53
Marking a device as unmanaged from the Device Structure browser	8-54
Identifying unmanaged interfaces	8-55
Unmanaging devices in PopulateDNCIM_ManagedStatus.stch	8-56
Topic summary	8-57
8.5. Configure adaptive polling	8-58
Configure adaptive polling	8-58
Adaptive polling overview	8-59
Adaptive polling examples	8-60
Adaptive polling example 1	8-61
Adaptive polling example 2	8-62
Topic summary	8-63
8.6. Real-time MIB graphing	8-64
Real-time MIB graphing	8-64
Starting the SNMP MIB Grapher	8-65
Specify polling parameters	8-66
SNMP MIB Grapher	8-67
MIB graph elements	8-68
Topic summary	8-69
8.7. Troubleshooting information	8-70
Troubleshooting information	8-70
Reasons to troubleshoot polling	8-71
Troubleshooting ping polling (1 of 3)	8-72
Troubleshooting ping polling (2 of 3)	8-73
Troubleshooting ping polling (3 of 3)	8-74
Log files for troubleshooting SNMP polling	8-75
Troubleshooting polling GUI	8-76
Troubleshooting information	8-77
Troubleshooting monitoring status	8-78
Polling exception report (1 of 2)	8-79
Polling exception report (2 of 2)	8-80
Troubleshooting ping polling with itnm_poller.pl (1 of 2)	8-81
Troubleshooting ping polling with itnm_poller.pl (2 of 2)	8-82
Topic summary	8-83
Exercise: Configuring polling	8-84
Review questions	8-85
Unit summary	8-86
Review answers	8-87

Unit 9. Understanding DNCIM	9-1
Understanding DNCIM	9-1
Unit objectives	9-3
9.1. DNCIM database basics.....	9-4
DNCIM database basics	9-4
DNCIM defined	9-5
Data flow	9-6
Data structure and format	9-7
Version 4.2 discovery data processing	9-8
Reasons to use DNCIM	9-9
Stitching process changes	9-10
DNCIM configuration (1 of 2)	9-11
DNCIM configuration (2 of 2)	9-12
DNCIM key tables (1 of 2)	9-13
DNCIM key tables (2 of 2)	9-14
Removing the DNCIM database tables	9-15
How to start a new clean discovery without discovery cache files	9-16
9.2. Populating DNCIM database	9-17
Populating the DNCIM database	9-17
PopulateDNCIM.stch (1 of 2)	9-18
PopulateDNCIM.stch (2 of 2)	9-19
RecordToDncimDb([optional record])	9-20
9.3. Mapping information into the MODEL service	9-21
Mapping information into the MODEL service	9-21
Example mapping in ModelNcimDb.cfg	9-22
InferDNCIMObjects.stch	9-23
PopulateDNCIM.stch	9-24
Unit summary	9-25
Unit 10. Customizing discovery	10-1
Customizing discovery	10-1
Unit objectives	10-2
10.1. Discovery in-depth	10-3
Discovery in-depth	10-3
Discovery overview	10-4
Discovery details	10-5
Enhancing discovery with business information	10-6
Discovery beginnings	10-7
FnderRetProcessing.stch	10-8
Record data flow	10-9
FnderProcToDetailsDesp.stch	10-10
DetailsRetProcessing.stch	10-11
IpToBaseName.stch	10-13
AssocAddressRetProcessing.stch	10-14
Final-phase stitchers define entities and containment	10-15
Passing entity data to the DNCIM database	10-16
Topic summary	10-17
10.2. Stitcher language constructs	10-18
Stitcher language constructs	10-18
Stitcher language constructs	10-19
Types of discovery stitchers	10-20
Discovery stitchers	10-21
Stitcher data types	10-22
Stitcher assignment operators	10-23
Stitcher relational operators	10-24
Stitcher language constructs	10-25

Creating a RecordList with RetrieveOQL	10-26
Creating a RecordList with SQLData and PrepareSQL	10-27
Processing the RecordList	10-28
Stitcher structure	10-29
Using variables in the stitcher rule	10-30
Stitcher triggers	10-31
Generic stitcher triggers	10-32
Discovery stitcher triggers	10-33
Generic stitcher rules (1 of 2)	10-34
Generic stitcher rules (2 of 2)	10-35
FnderProcToDetailsDesp.stch	10-36
Using fields from a trigger	10-37
Evaluation statement	10-38
Other services add fields to NCIM entity records	10-39
Topic summary	10-40
10.3. Creating database tables	10-41
Creating database tables	10-41
Creating database tables	10-42
Business scenario	10-43
Create custom tables	10-44
Create lookup tables that are used for stitcher processing (1 of 2)	10-45
Create lookup tables that are used for stitcher processing (2 of 2)	10-47
10.4. Prepare stitchers to enrich discovery	10-48
Prepare stitchers to enrich discovery	10-48
Edit the \$ITNHOME/disco/stitchers/DNCIM/PopulateDNCIM.stch	10-49
Edit RefreshDNCIM.stch	10-50
Build custom stitcher (such as CustomEnrichmentNew.stch) (1 of 9)	10-51
Build custom stitcher (2 of 9)	10-52
Build custom stitcher (3 of 9)	10-53
Build custom stitcher (4 of 9)	10-54
Build custom stitcher (5 of 9)	10-55
Build custom stitcher (6 of 9)	10-56
Build custom stitcher (7 of 9)	10-57
Build custom stitcher (8 of 9)	10-58
Build custom stitcher (9 of 9)	10-59
10.5. Run the custom scripts and stitchers	10-61
Run the custom scripts and stitchers	10-61
Build the new DNCIM database	10-62
Verify that custom DNCIM tables populated	10-63
Define custom tables in \$NCHOME/etc/precision/ModelNcimDb.cfg	10-64
Populate custom table with data in ModelNcimDb.cfg	10-65
Restart the MODEL service	10-66
Create Network Views with any keyName-keyValue pair in entityDetails	10-67
Modify ncimMetaData.xml to use custom table fields	10-68
Build new views with enriched information	10-69
Build useful views	10-70
Topic summary	10-71
10.6. Using the PresetLayer stitcher	10-72
Using the PresetLayer stitcher	10-72
Using the PresetLayer stitcher	10-73
Islands of connectivity	10-74
PresetLayer.stch introduction	10-76
Manually instantiating devices	10-77
Manually instantiating interfaces	10-78
Manually instantiate containment relationships	10-79
Building connections between interfaces	10-80

Topic summary	10-81
Exercise: Customizing discovery	10-82
Review questions	10-83
Unit summary	10-84
Review answers	10-85
Unit 11. The gateway and RCA	11-1
The gateway and RCA	11-1
Unit objectives	11-2
11.1. Architecture and basic concepts	11-3
Architecture and basic concepts	11-3
Basic concepts (1 of 2)	11-4
Basic concepts (2 of 2)	11-5
Architectural overview	11-6
Architectural rationale	11-7
Architectural details	11-8
Plug-in summary	11-9
Event processing	11-10
Tivoli Network Manager data flow	11-12
Event enrichment data flow	11-14
Selecting an entity for processing	11-15
Event enrichment	11-16
Event processing plug-ins	11-17
Custom plug-in development (1 of 3)	11-18
Custom plug-in development (2 of 3)	11-19
Custom plug-in development (3 of 3)	11-20
Access to ObjectServer tables (1 of 2)	11-21
Access to ObjectServer tables (2 of 2)	11-22
Custom enrichment	11-23
Gateway stitcher rule summary	11-24
11.2. RCA overview	11-25
RCA overview	11-25
RCA overview	11-26
RCASchema.cfg	11-27
MaxAgeDifference	11-28
HonourManagedStatus	11-29
RCA process	11-30
Precedence values in NcKL rules files, formerly in NcoGateInserts.cfg file	11-31
11.3. Gateway plug-ins	11-32
Gateway plug-ins	11-32
New fields in mojo.events	11-33
RCA stitcher processing order	11-34
RCA stitchers	11-35
Timed RCA rule	11-36
Customizing plug-ins	11-37
Example modification	11-38
Calling a stitcher from ProcessProblemEvent.stch	11-39
More stitcher functions	11-42
Optional stitcher functions (1 of 2)	11-45
Optional stitcher functions (2 of 2)	11-46
Suppression types in RCA rules	11-47
Determining what plug-ins are enabled	11-48
Enabling a plug-in	11-49
11.4. New rules for special events	11-50
New rules for special events	11-50
1. Does the gateway already handle the event?	11-51

2. Determine whether plug-in handles event	11-53
3. Reassign event to trigger plug-in	11-54
Example of reusing an existing event map	11-55
Example of creating an eventMap (1 of 2)	11-56
Example of creating an event map (2 of 2)	11-57
4. Restart the gateway	11-58
11.5. Troubleshooting gateway event processing	11-59
Troubleshooting gateway event processing	11-59
Troubleshooting with trace and log files	11-60
Tracking progress of event (1 of 2)	11-61
Tracking progress of event (2 of 2)	11-62
Troubleshooting the handling of events (1 of 2)	11-63
Troubleshooting the handling of events (2 of 2)	11-64
Troubleshooting entity selection	11-65
Troubleshooting field updates	11-66
11.6. Service-affecting events	11-67
Service-affecting events	11-67
Introduction	11-68
Service-affecting events plug-in	11-69
SAESchema.cfg	11-72
Checking ServiceType queries (1 of 2)	11-73
Checking ServiceType queries (2 of 2)	11-74
Key ncimCache tables	11-75
Exercise: The gateway and RCA	11-76
Review questions	11-77
Unit summary	11-78
Review answers	11-79
Unit 12. Tivoli Network Manager scripts	12-1
Tivoli Network Manager scripts	12-1
Objectives	12-2
\$ITNMHOME/scripts/perl/scripts (1 of 3)	12-3
\$ITNMHOME/scripts/perl/scripts (2 of 3)	12-4
\$ITNMHOME/scripts/perl/scripts (3 of 3)	12-5
\$ITNMHOME/bin	12-6
\$ITNMHOME/scripts/upgrade	12-7
Other scripts	12-8
Unit summary	12-9
Unit 13. Tivoli Network Manager failover	13-1
Tivoli Network Manager failover	13-1
Objectives	13-2
13.1. Failover basics	13-3
Failover basics	13-3
Basic concepts of failover	13-4
Architecture overview	13-5
Event gateway	13-6
Database replication	13-7
Replicating failover	13-8
13.2. Configuring failover	13-9
Configuring failover	13-9
Failover configuration steps	13-10
The ConfigInnm.cfg file	13-11
Locking ports in ServiceData.cfg	13-12
13.3. What happens with failover	13-13
What happens with failover	13-13

When failover happens	13-14
When failback happens	13-15
Tivoli Network ManagerHealthChk events	13-16
ItnmDatabaseConnection event	13-17
Tivoli Network Manager failover events	13-18
ItnmFailoverConnection events	13-19
Database high availability with DB2 HADR	13-20
ObjectServer failover	13-21
DB2 installed by Tivoli Network Manager	13-22
High availability installations	13-23
Replicating failover	13-24
13.4. Troubleshooting failover	13-25
Troubleshooting failover	13-25
Determining why failover occurred	13-26
No health check success message is received from the primary server	13-27
TCP connection problems	13-28
Virtual domain log messages	13-29
Troubleshooting domain issues in failover	13-30
Review questions	13-31
Unit summary	13-32
Review answers	13-33
Unit 14. Installing Tivoli Network Manager	14-1
Installing Tivoli Network Manager	14-1
Unit objectives	14-3
14.1. Installation preparation	14-4
Installation preparation	14-4
Supported operating systems	14-5
Included base products and components	14-6
Library requirements for Linux	14-7
Supported browsers for GUI	14-8
Processors, memory, and bandwidth requirements	14-9
Disk space requirements	14-10
Disable SELinux and firewall before installation	14-11
Create essential users	14-12
Prepare the operating system	14-13
Setting file limits for the non-root user	14-14
Verify write permissions	14-15
Install required software components	14-16
14.2. Installing Tivoli Network Manager	14-17
Installing Tivoli Network Manager	14-17
Run the prerequisite scanner	14-18
Configure the NCIM database	14-19
Using the Installation Manager	14-20
Select a repository	14-21
Install the Network Manager core components	14-22
Start the installation	14-23
Configure ObjectServer options	14-24
Select passwords	14-25
Specify Tivoli Network Manager domain name	14-26
Configure database table creation	14-27
Confirm installation directory for Python	14-28
Start the installation	14-29
14.3. Component communications	14-30
Component communications	14-30
Core component communication configuration (1 of 2)	14-31

Core component communication configuration (2 of 2)	14-32
ServiceData.cfg example	14-33
Uninstalling Tivoli Network Manager 4.2	14-34
14.4. Postinstallation tasks	14-35
Postinstallation tasks	14-35
Set processes to run as non-root	14-36
Verify that OMNIBus processes are run as non-root	14-37
Configure non-root running of nco_p_mttrapd	14-38
14.5. Process control.....	14-39
Process control	14-39
CtrlServices.cfg file	14-40
A managed process entry	14-41
An unmanaged process entry	14-42
Discovery startup in CtrlServices.cfg	14-43
14.6. Starting Tivoli Network Manager.....	14-44
Starting Tivoli Network Manager	14-44
Run the itnm_start command	14-45
Review questions	14-46
Unit summary	14-47
Review answers	14-48
Unit 15. Tivoli Network Manager reports	15-1
Tivoli Network Manager reports	15-1
Unit objectives	15-2
15.1. Overview.....	15-3
Overview	15-3
Tivoli Common Reporting for Network Manager	15-4
Reports overview (1 of 2)	15-5
Reports overview (2 of 2)	15-6
Tivoli Common Reporting menu options	15-7
Default report sets	15-8
15.2. Viewing and scheduling reports.....	15-10
Viewing and scheduling reports	15-10
Accessing the report list	15-11
Start a report	15-12
Start polling reports from device icon	15-13
Available report formats	15-14
View a report	15-15
Toolbar selections	15-16
Report options	15-17
Click column names to sort reports	15-18
Edit a Cognos report	15-19
Schedule reports	15-20
Ways to run and visualize reports (1 of 2)	15-21
Ways to run and visualize reports (2 of 2)	15-22
Data model	15-23
Create a report in Report Studio	15-24
15.3. Included reports	15-25
Included reports	15-25
Asset reports	15-26
Current Status reports	15-27
Network Technology reports	15-28
Monitoring reports	15-29
Network Views reports	15-30
New network health reports	15-31
Path Views reports	15-32

Performance reports	15-33
Reports for troubleshooting network devices	15-34
Utility reports	15-35
15.4. Installing Tivoli Common Reporting	15-36
Installing Tivoli Common Reporting	15-36
Installing Tivoli Common Reporting	15-37
Deployment options	15-38
Content Manager configuration	15-39
Use Cognos administration to change sign-on password	15-40
Changing sign-on password (1 of 2)	15-41
Changing sign-on password (2 of 2)	15-42
15.5. Troubleshooting Cognos	15-43
Troubleshooting Cognos	15-43
Tivoli Network Manager data model (1 of 2)	15-44
Tivoli Network Manager data model (2 of 2)	15-45
The Cognos report package	15-46
Tivoli Common Reporting log files	15-47
Troubleshooting Cognos	15-48
Cognos viewer errors (1 of 2)	15-49
Cognos viewer errors (2 of 2)	15-50
Tivoli Common Reporting documentation	15-51
Troubleshooting an Oracle data source (1 of 3)	15-52
Troubleshooting an Oracle data source (2 of 3)	15-53
Troubleshooting an Oracle data source (3 of 3)	15-54
Troubleshooting a DB2 data source (1 of 2)	15-55
Troubleshooting a DB2 data source (2 of 2)	15-56
Exercises: Tivoli Network Manager reports	15-57
Review questions	15-58
Unit summary	15-59
Review answers	15-60
Unit 16. Event searches	16-1
Event searches	16-1
Objectives	16-2
Event Search Analytics for operational agility and efficiency	16-3
Event search overview	16-4
Searching events	16-5
Refining the event search	16-6
OMNIbus Event Dashboard	16-7
Search for similar events	16-8
Search for all events on the same device	16-9
Exercise: Event searches in Netcool Operations Insight	16-10
Unit summary	16-11
Unit 17. Log Analysis basics	17-1
Log Analysis basics	17-1
Objectives	17-3
Collect and process log data	17-4
Extracting meaning from logs	17-5
Simple chart and dashboard creation	17-6
Compare messages from different log sources	17-7
Expert guidance	17-8
Insight Packs provide content	17-9
Exercise: Log analysis	17-10
Unit summary	17-11

Unit 18. Seasonal events	18-1
Seasonal events	18-1
Objectives	18-2
Seasonal event function	18-3
How seasonality is determined	18-4
Creating a configuration	18-5
Run a configuration	18-6
Viewing seasonal events analysis results	18-7
Sample seasonal reports	18-8
Examining event details	18-9
Seasonal event rules	18-10
Creating a seasonal event rule	18-11
Seasonal event rule results	18-12
Event rule result	18-13
Exercise: Find related events	18-14
Unit summary	18-15
Unit 19. Related events	19-1
Related events	19-1
Objectives	19-2
Feature overview	19-3
Related events workflow	19-4
Creating a configuration	19-5
Run a configuration	19-6
Viewing related events analysis results	19-7
Examining event group details	19-8
Deploying a configuration	19-9
Viewing grouped events	19-10
Exercises: Finding seasonal events, and Using Tivoli Netcool Configuration Manager	19-11
Unit summary	19-12
Unit 20. Optical transport and Radio Access Networks	20-1
Optical transport and Radio Access Networks	20-1
Objectives	20-3
Optical, RAN discovery support	20-4
Components that support optical transport and RAN devices	20-5
Tivoli Network Manager vendor EMS support	20-6
Optical transport and RAN EMS integration	20-8
Optical and RAN topology visualization	20-9
GSM, GPRS, UMTS, and RAN topology visualization	20-10
Multiple domain visualizations	20-11
NCIM EMS integration support	20-12
Optical, RAN discovery: Collector agents	20-14
Starting a Java collector	20-15
Stopping a Java collector	20-16
Optical and RAN device icons	20-17
Optical and RAN devices in Device Structure browser	20-18
Managed systems view	20-19
Unit summary	20-20
Appendix A. Databases and passwords	A-1
Key DB2 commands (1 of 2)	A-2
Key DB2 commands (2 of 2)	A-3
Re-create a topology database instance	A-4
General troubleshooting commands	A-5
Diagnostic commands (1 of 4)	A-6

Diagnostic commands (2 of 4)	A-7
Diagnostic commands (3 of 4)	A-8
Diagnostic commands (4 of 4)	A-9
Performance Analyst tool	A-10
Analyze the snapshot.txt file	A-11
Statement tab	A-12
Performance tuning with autoconfigure	A-13
Transaction log size (1 of 2)	A-14
Transaction log size (2 of 2)	A-15
Maximum number of connections	A-16
DB2 database tuning (1 of 2)	A-17
DB2 database tuning (2 of 2)	A-18
Automatic tuning	A-19
Registry settings	A-20
Database manager settings	A-21
Changing passwords	A-22
Changing DB2 database passwords	A-23
Changing passwords in configuration files	A-24
Tivoli Network Manager GUI database access	A-25
Tivoli Common Reporting BIRT and Cognos reports	A-26
Tivoli Common Reporting BIRT reports	A-27
Tivoli Common Reporting Cognos with Tivoli Data Warehouse	A-28
Use Cognos administration to change Signon password	A-30
Cognos Signon page (1 of 3)	A-31
Cognos Signon page (2 of 3)	A-32
Cognos Signon page (3 of 3)	A-33
Helpful IBM Knowledge Center links	A-34
Useful DB2 references	A-35

Appendix B. Configuring DB2 high availability	B-1
DB2 instances	B-2
DB2 instance configuration (1 of 2)	B-3
DB2 instance configuration (2 of 2)	B-4
Creating the primary DB2 database	B-5
Key DB2 concepts: Services	B-6
Troubleshooting: Problems accessing DB2 (1 of 3)	B-7
Troubleshooting: Problems accessing DB2 (2 of 3)	B-8
Key DB2 concepts: Logging and configuration	B-9
Configuring a DB2 server for HADR: Logs	B-10
Troubleshooting: Problems accessing DB2 (3 of 3)	B-11
Creating the standby DB2 database (1 of 3)	B-12
Creating the standby DB2 database (2 of 3)	B-13
Creating the standby DB2 database (3 of 3)	B-14
Troubleshooting: Creating the standby DB	B-15
Key DB2 concepts: HADR	B-16
Configuring HADR hosts and services	B-17
Configuring HADR	B-18
Key DB2 concepts: ACR	B-19
Enabling HADR	B-20
Troubleshooting: HADR fails to start	B-21
Troubleshooting: Validate HADR configuration	B-22
Key DB2 concepts: Catalogs and nodes	B-23
Configuring a client catalog	B-24
Troubleshooting: Client catalog	B-25
Tivoli Network Manager back-end configuration (1 of 2)	B-26
Tivoli Network Manager back-end configuration (2 of 2)	B-27

Troubleshooting: Tivoli Network Manager back-end configuration (1 of 2)	B-28
Troubleshooting: Tivoli Network Manager back-end configuration (2 of 2)	B-29
Tivoli Integrated Portal configuration	B-30
Failing over NCIM	B-31
Key DB2 concept: Clustering with Tivoli System Automation for Multiplatform	B-32
Failing over NCOM without manual intervention	B-33
Troubleshooting: HADR state and connected processes	B-34
Troubleshooting: Connections to alternative server	B-35
Failing over Tivoli Network Manager	B-36
Topic summary	B-37
Reference material (1 of 2)	B-38
Reference material (2 of 2)	B-39
Appendix C. List of abbreviations	C-1

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AIX®	Cognos®	DB™
DB2®	FileNet®	IBM Connections™
Informix®	Insight™	Jazz™
Netcool®	Notes®	Tivoli®
WebSphere®	400®	

ITIL is a Registered Trade Mark of AXELOS Limited.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

VMware is a registered trademark or trademark of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

Course description

Tivoli Network Manager 4.2 Operations and Administration

Duration: 5 days

Purpose

This course is intended to teach the skills that are needed to discover network devices at Layers 1 – 3 of the OSI model, use the topology view, and configure and manage polling of network devices. You learn how to view topology reports and to use the Network Health Dashboard.

Audience

This course is intended for operators, administrators, systems designers, and integration partners for Tivoli network management products.

Prerequisites

- Basic understanding of network management concepts
- Basic understanding of UNIX commands because the workshop uses a Linux environment
- Basic understanding of database concepts
- Familiarity with Tivoli Netcool/OMNIbus
- Ability to edit files in UNIX or Linux with vi or gedit
- Understanding of Layers 1 – 3 of the open systems interconnection (OSI) model
- Familiarity with router configuration tasks, access control lists (ACLs), and network address translation

Objectives

- Configure, run, and schedule a network discovery
- Navigate network maps
- Create custom network partition views
- Enrich network events with topology information
- Add business information to topology maps
- Customize the graphical user interface (GUI)
- Configure, customize, and run network monitoring
- Configure, customize, and schedule reports
- Create incident filters and views

- Search for similar events
- Identify seasonal events
- Use the Network Health Dashboard

Contents

- Overview of Network Operations Insight
- Overview of Tivoli Network Manager
- Using Network Operations Insight search capabilities
- Discovering and viewing network topology
- Using the Network Health Dashboard

Curriculum relationship

This course is based on Tivoli Network Manager 4.2. Its predecessor, TN324, was based on Tivoli Network Manager 4.1.1. The two products are fundamentally different in the GUI and in the discovery database operations. TN324 must be maintained while the customer base continues to use Tivoli Network Manager 4.1.1. TN325 serves those customers who use or upgrade to Tivoli Network Manager 4.2.

Agenda



Note

The following unit and exercise durations are estimates, and might not reflect every class experience.

Day 1

- (00:15) Course introduction
- (00:15) Preface
- (01:30) Unit 1. Introduction to Netcool Operations Insight (part 1)
- (01:15) Exercise 1. Review of Netcool Operations Insight architecture
- (01:30) Unit 2. Tivoli Network Manager 4.2 introduction and architecture
- (01:30) Unit 3. Discovery basics
- (00:00) Exercise 2. Discovery basics
- (01:30) Unit 4. Advanced discovery options
- (00:00) Exercise 3. Advanced discovery options

Day 2

- (01:00) Unit 5. Topology visualization basics
- (00:00) Exercise 4. Topology visualization basics
- (01:30) Unit 6. Advanced visualization
- (00:00) Exercise 5. Advanced visualization
- (00:15) Unit 7. The Network Health Dashboard
- (00:30) Exercise 6. Using the Network Health Dashboard
- (01:30) Unit 8. Monitoring and polling network devices
- (00:00) Exercise 7. Configuring polling

Day 3

- (00:45) Unit 9. Understanding DNCIM
- (01:30) Unit 10. Customizing discovery
- (00:00) Exercise 8. Customizing discovery
- (01:00) Unit 11. The gateway and RCA
- (00:00) Exercise 9. The gateway and RCA

Day 4

- (00:30) Unit 12. Tivoli Network Manager scripts
- (00:30) Unit 13. Tivoli Network Manager failover
- (01:00) Unit 14. Installing Tivoli Network Manager
- (00:45) Unit 15. Tivoli Network Manager reports
- (00:00) Exercise 10. Tivoli Network Manager reports

Day 5

- (00:20) Unit 16. Event searches
- (00:45) Exercise 11. Event searches in Netcool Operations Insight
- (00:45) Unit 17. Log Analysis basics
- (01:00) Exercise 12. Log Analysis
- (00:20) Unit 18. Related events
- (00:30) Exercise 13. Find related events
- (00:20) Unit 19. Seasonal events
- (00:30) Exercise 14. Finding seasonal events
- (00:45) Exercise 15. Using Tivoli Netcool Configuration Manager
- (00:00) Unit 20. Optical RAN

Unit 1. Introduction to Netcool Operations Insight

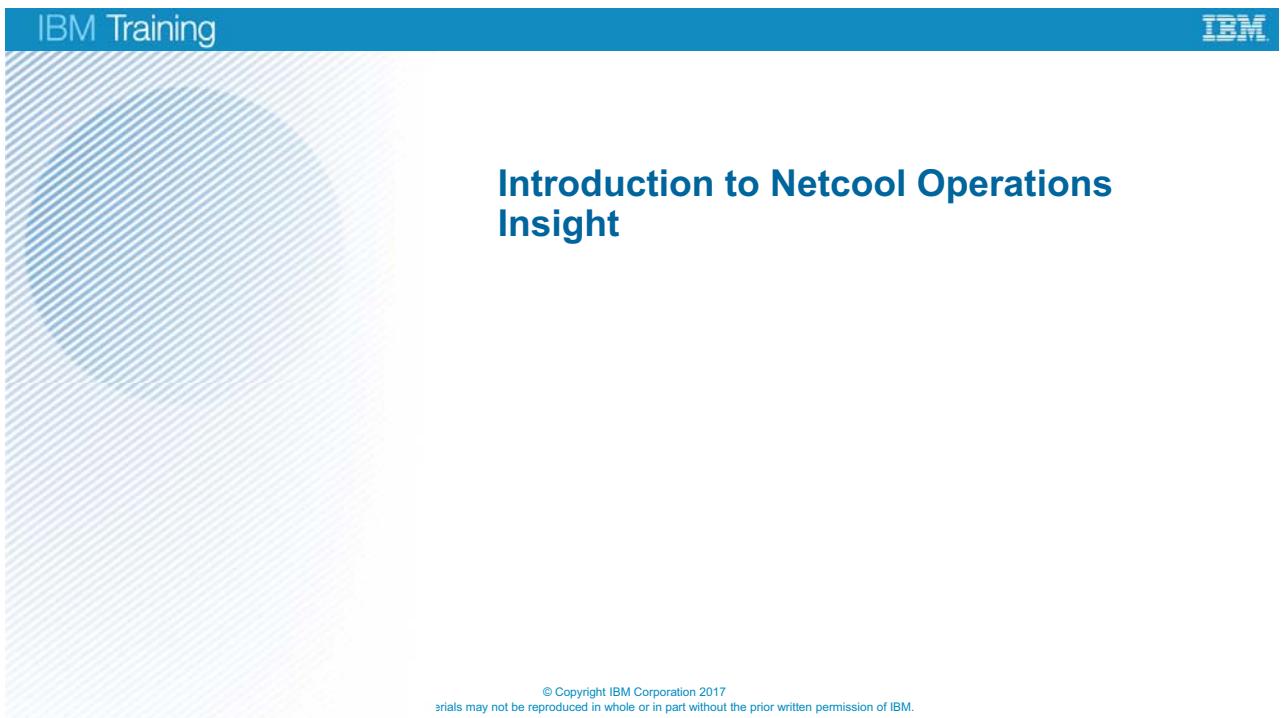


Figure 1-1. Introduction to Netcool Operations Insight

Estimated time

01:30

Overview

This unit describes the architecture and function of the components inside Netcool Operations Insight, and Tivoli Netcool Configuration Manager.

Unit objectives

- Identify the major components in a typical Netcool architecture
- Describe what functions each component of Netcool Operations Insight provides
- Describe how the Netcool solution relates to steps in the Information Technology Infrastructure Library (ITIL) incident management process
- View events that exist in the OMNIbus ObjectServer
- Create a custom event view
- Create a custom event filter

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-2. Unit objectives

Realize the benefits of Netcool Operations Insight



Within seconds, identify, diagnose, and resolve problems

- With Event Search, IT operations staff with no programming or report writing skills can search large volumes of events and run ad hoc queries within 30 seconds, instead of hours



30% reduction in overall event load and 10% faster MTTR

- A large North American bank identified opportunities to reduce their event load within OMNIbus by 30% based on one-day workshop with Netcool Operations Insight event search
- The customer estimates that this decrease reduces average time-to-repair per incident by more than 10%



10% reduction in number of events that are presented to operators

- By using Netcool Operations Insight Event Analytics for seasonal event identification, as found in customer environment testing

\$1,000 per trouble ticket **not opened!**

- Netcool Operations Insight with Event Analytics helps clients spot seasonal problems and reduce event volumes, leading to a reduction in trouble tickets*
- Trouble tickets can cost an organization an average of \$1,000 per trouble ticket*

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-3. Realize the benefits of Netcool Operations Insight

One of the primary advantages of Netcool Operations Insight is the wealth of included capabilities and the ease of use.

Netcool Operations Insight base features

- Event collection
 - Probes insert events at a high rate into a memory-resident active database event search
 - Applies the search and analysis capabilities of **Operations Analytics Log Analysis** to events in Tivoli Netcool/OMNibus
 - The Log Analysis engine imports OMNibus events and indexes them for searching
- Event Analytics
 - Netcool/Impact analyzes Tivoli Netcool/OMNibus historical event data
- IBM Connections integration
 - Netcool/Impact enables social collaboration through IBM Connections by automatically providing updates to key stakeholders
- Service Management dashboard

Figure 1-4. Netcool Operations Insight base features

1.1. Base solution components

IBM Training

IBM

Base solution components

- Tivoli Netcool/OMNIbus core V8.1.0.5
- Gateway for JDBC
 - Is used to populate event archive database
- Tivoli Netcool/OMNIbus Web GUI V8.1.0.4
 - With Netcool Operations Insight extensions
- Netcool/Impact V7.1.0.4
 - With Netcool Operations Insight extensions
- IBM Operations Analytics Log Analysis V1.3.2 Standard Edition
- Tivoli Netcool/OMNIbus Insight Pack V1.3.0.2 for IBM Operations Analytics Log Analysis
- Gateway for Message Bus V7.0
- Jazz for Service Management V1.1.2.1

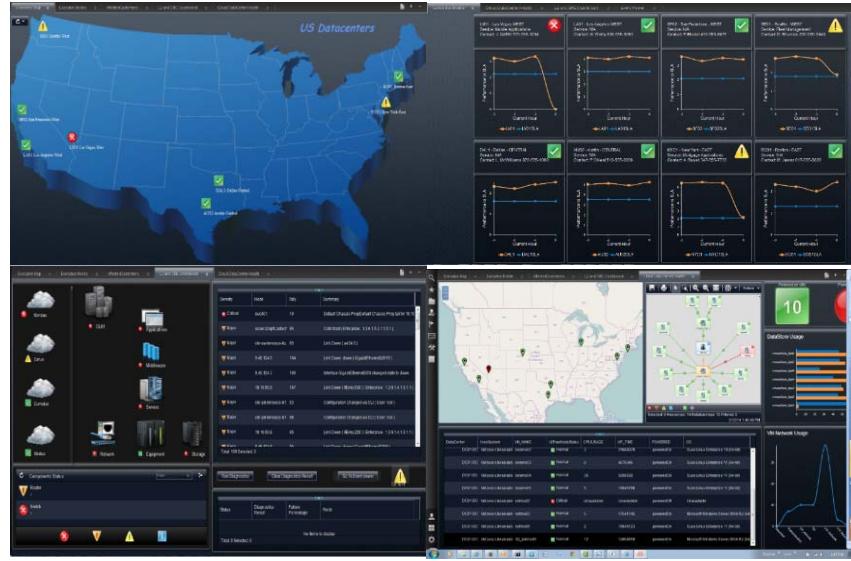
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-5. Base solution components

Comprehensive dashboards

- Modern widgets suitable for mobile devices
- 60% faster in creation of custom high-impact dashboards



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-6. Comprehensive dashboards

Dashboards are easier to create and suitable for access from mobile devices. Some sample dashboards are included with Netcool Operations Insight, but you must make custom dashboards.

Event Viewer in Netcool Operations Insight

- No longer uses Java applets
- No need for JREs on client workstations
- Renders pages faster
- 75% saving on cost of ownership

The screenshot shows the Event Viewer interface in Netcool Operations Insight. The main window displays a table of events. The columns are: Group, Count, Sev, Ack, Node, Alert Group, and Summary. The 'Group' column lists categories like All, AUSTIN, Cirrus_App, Cirrus_Infra, LASVEGAS, Nimbus_App, Nimbus_Infra, and Undefined. The 'Count' column shows the number of events for each group. The 'Sev' column indicates the severity level (e.g., Critical, Major, Minor). The 'Ack' column shows if the event has been acknowledged. The 'Node' column lists the hostnames of the systems where the events occurred. The 'Alert Group' column identifies the specific alert type. The 'Summary' column provides a brief description of the event. The interface includes a sidebar with various icons for search, star, file, and other system monitoring.

Group	Count	Sev	Ack	Node	Alert Group	Summary
All	5,514	critical	No	istpc101	Port to port Connect	The connection from host vio59501a port C952
AUSTIN	2	critical	No	istpc101	Port to port Connect	The connection from host vio59501a port C952
Cirrus_App	1,578	major	No	itmme02	ITM_KVM_DATASTORE	KVM_Datastore_Bad_Status Overall_Status<>
Cirrus_Infra	1,430	major	No	BRMS_01_IH	Omegamon_Base	This node has high JSP percent error count
LASVEGAS	1	critical	No	istpc101	Port to port Connect	The connection from host vio59501a port C952
Nimbus_App	317	critical	No	istpc101	Port to port Connect	The connection from host vio59501a port C952
Nimbus_Infra	1,752	critical	No	istpc101	Port to port Connect	The connection from host vio59501a port C952
Undefined	434	major	Yes	9.45.124.5	Generic Link Status	Link Down: down (GigabitEthernet2/0/19)
		major	No	istpc101	Port to port Connection-	The connection from switch isanb384a (F819)
		major	No	isibc06m1.adtech.internet.ll	Bridge Topology Chang	VLAN Port Transitioned or (Port: Gi1/0/9 Doma
		major	No	OPNPG_01_DPEP_022_1'	Omegamon_Base	Perc peak free memory very low - risk of crash
		major	No	sbr-sa-innovsys-bz.adtech.i	Generic Link Status	Link Down (ge-4/0/6)
		major	No	dpev373	ITM_NT_Processor	GIDC_cputil_3ntw_std2 (Processor=Total_Av
		major	No	vmesx_eve-vmesxfaces	ITM_KVM_SERVER_DA	KVM_Server_Datastore_Free_Low (Percent_Fr

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

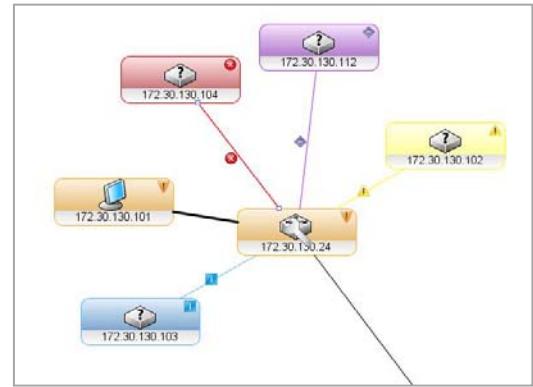
Figure 1-7. Event Viewer in Netcool Operations Insight

For existing Netcool customers, the Event Viewer means no longer needing to use the AEL and the Java plug-in. The Event Viewer does not use Java.

Optional network management components (1 of 5)

IBM Tivoli Network Manager V4.2

- Automatic discovery of network devices
- ICMP and SNMP polling of network devices
- Creation of network maps
- Populates network health dashboard
- Populates data that is used for asset and polling-based reports
- Creates Network Health Dashboard to show key metrics for specific network views
 - Device and interface availability for the selected view
 - Presents performance graphs, tables, and traces of KPI data for monitored devices and interfaces
 - Shows time correlation of network changes from Tivoli Configuration Manager with network events
- Network fault isolation (root cause analysis) to determine the real cause of network outages



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-8. Optional network management components (1 of 5)

Optional network management components (2 of 5)

Probes

- SNMP multithread probe (mtrapd)
- Syslog probe

Network Manager Insight Pack V1.3.0.0 for IBM Operations Analytics Log Analysis

- Analyzes topology records from Network Manager to enable topology-based searches
- Topology search
 - An extension of the Networks for Operations Insight feature
 - Applies the search and analysis capabilities of Operations Analytics Log Analysis to give insight into network diagnostics

Network Performance Insight

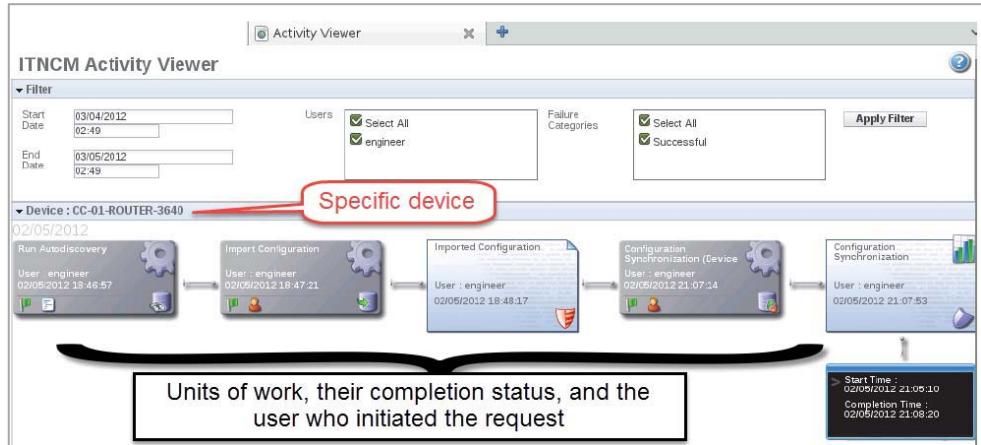
- Flow-based network traffic performance monitoring system
- Provides comprehensive and scalable visibility on network traffic with visualization and reporting of network performance data for complex, multivendor, multi-technology networks

Figure 1-9. Optional network management components (2 of 5)

Optional network management components (3 of 5)

- **IBM Tivoli Netcool Configuration Manager V6.4.2**

- Specify configuration changes in a browser interface
- Schedule those changes
- Have parties approve the changes
- View Unit of Work (UOW) results in **Activity Viewer**



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-10. Optional network management components (3 of 5)

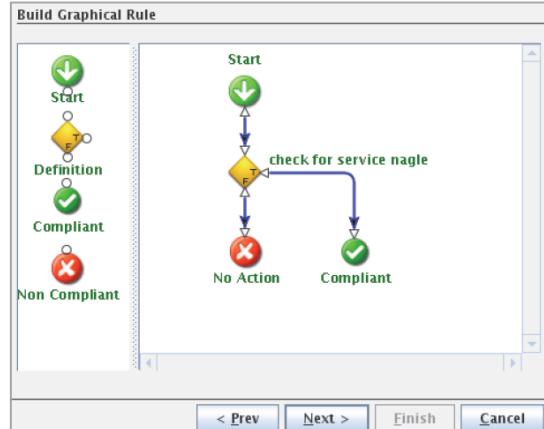
The main part of Tivoli Netcool Configuration Manager does these things:

- Imports inventory data from Tivoli Network Manager discovery
- Retrieves device configuration information into XML database
- Schedules units of work (UOW) to configure devices
- Records results of attempted configuration changes
- Enables rollback to previous configurations when errors occur
- Creates workflow and approval process to implement configuration changes
- Enables configuration of network devices without knowledge of vendor-specific syntax details
- Upgrades IOS

Optional network management components (4 of 5)

IBM Tivoli Netcool Configuration Manager V6.4.2

- The Tivoli Netcool Configuration Manager automatically checks for compliance to 17 security policies that the National Security Administration (NSA) designed to harden your network security
- You can use the Policy Editor to create additional policies to ensure that network devices are compliant with your business practice standards
 - It can then check to see whether network devices are compliant with the standards, and it alerts you to those devices that fail the compliance check
- You can also configure the policy editor to automatically make changes to enforce compliance for certain configuration items



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-11. Optional network management components (4 of 5)

The Policy Editor in Tivoli Network Configuration manager checks network devices to see whether they are compliant with 17 policies identified by the National Security Administration (NSA) as being essential to harden network devices to prevent intrusions into your network.

You can also create policies unique to your business.

Optional network management components (5 of 5)

- **IBM Runbook Automation**

- Investigate and delegate problems faster and more efficiently
- Diagnose and fix problems faster and build operational knowledge
- Easily create, publish, and manage runbooks and automations
- Keep score to track achievements and find opportunities for improvement

- **IBM Alert Notification**

- Provides instant notification of alerts for any critical IT issues across multiple monitoring tools
- Gives IT staff instant notification of alerts for any issues in your IT operations environment

Figure 1-12. Optional network management components (5 of 5)



Topology search (1 of 2)

The screenshot shows a list of network events in a table format. An event for node 172.20.1.3 is selected and has a context menu open. The menu items are: Acknowledge, De-acknowledge, Prioritize, Suppress/Escalate, Take ownership, User Assign, Group Assign, Delete, Ping, Event Search, Information..., Copy, and Show_Alerts_Between_Two_Nodes.

Sev	Ack	Node	Alert Group	Summary	First Occurrence	Last Occurrence	Count	Type	
!	No	172.20.1.3	Generic Link Status	Link Down, Administratively (Isi:1048583)	9/10/14 11:39:50 PM	9/10/14 11:39:50 PM	1	Problem	
!	No	172.20.1.3	Generic Link Status	Link Down, Administratively (Isi:1048580)	9/10/14 10:57:07 PM	9/10/14 10:57:07 PM	1	Problem	
!	No	172.20.1.3	Generic Link Status	Link Down, Administratively (Isi:1048581)	9/10/14 11:31:19 PM	9/10/14 11:31:24 PM	2	Problem	
!	No	172.20.1.3	Generic Link Status	Link Down, Administratively (Isi:1048587)	9/12/14 5:15:30 PM	9/12/14 5:15:30 PM	1	Problem	
!	No	172.20.1.3	Generic Link Status	Link Down, Administratively (Isi:1048586)	9/11/14 11:11:49 AM	9/11/14 11:11:49 AM	1	Problem	
!	No	172.20.1.3	Generic Link Status	Link Down, Administratively (Isi:1048576)	9/8/14 11:36:28 AM	9/8/14 11:36:28 AM	1	Problem	
!	No	172.20.1.3	Generic Link Status	Link Down, Administratively (Isi:1048578)	9/8/14 11:37:21 AM	9/8/14 11:37:21 AM	1	Problem	
!	No	172.20.1.3	Generic Link Status	Link Down, Administratively (Isi:1048577)	9/10/14 10:49:22 AM	9/10/14 10:49:22 AM	2	Problem	
!	No	172.20.1.3	Ping	Ping Status	Link Down, Administratively (Isi:1048582)	9/10/14 11:31:11 PM	9/10/14 11:31:11 PM	1	Problem
!	No	172.20.1.3	Ping	Ping Status	Link Down, Administratively (Isi:1048584)	9/11/14 12:16:32 AM	9/11/14 12:16:32 AM	1	Problem
!	No	172.20.1.3	Ping	User Assign	Label Switched Path Down, Topology Changed (LSP FEC: 9.198.131.0, Router ID: 172.20.1.3)	7/28/14 6:48:33 PM	7/28/14 6:48:39 PM	2	Problem
!	No	172.20.1.4	Ping	Group Assign	Link Down, Administratively (Isi:1048581)	9/11/14 12:38:46 AM	9/11/14 12:38:46 AM	1	Problem
!	No	172.20.1.4	Ping	Delete	Link Down, Administratively (Isi:1048582)	9/11/14 12:38:45 AM	9/11/14 12:38:45 AM	1	Problem
!	No	172.20.1.4	Ping	Ping	Link Down, Administratively (Isi:1048583)	9/11/14 12:38:45 AM	9/11/14 12:38:45 AM	1	Problem
!	No	172.20.1.4	Ping	Event Search	Show event dashboard by node	9/11/14 12:38:46 AM	9/11/14 12:38:46 AM	1	Problem
!	No	172.20.1.4	Ping	Information...	Search for similar events	9/11/14 12:38:46 AM	9/11/14 12:38:46 AM	1	Problem
!	No	172.20.1.5	Ping	Copy	Search for events by node	9/11/14 12:38:46 AM	9/11/14 12:38:46 AM	1	Problem
!	No	172.20.1.5	BGP Peer S...	Show keywords and event count	9/8/14 11:36:05 AM	9/8/14 11:36:05 AM	1	Problem	
!	No	172.20.1.5	BGP Peer S...	Show_Alerts_Between_Two_Nodes	Peer: 172.20.2.94)	8/23/14 11:40:28 PM	8/23/14 11:56:33 PM	5	Problem

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-13. Topology search (1 of 2)

You can right-click an event and select **Event Search**. The following options are available:

- **Show event dashboard by node**
- **Search for similar events**
- **Search for events by node**
- **Show keywords and event count:**
 - Shows log file entries and other events that have the same keywords and an event count by keyword
- **Show_Alerts_Between_Two_Nodes:**
 - Quickly identifies the routes between two devices from the network topology
 - Generates a graph that shows an event severity summary for each route and reveals problematic routes

Topology search (2 of 2)

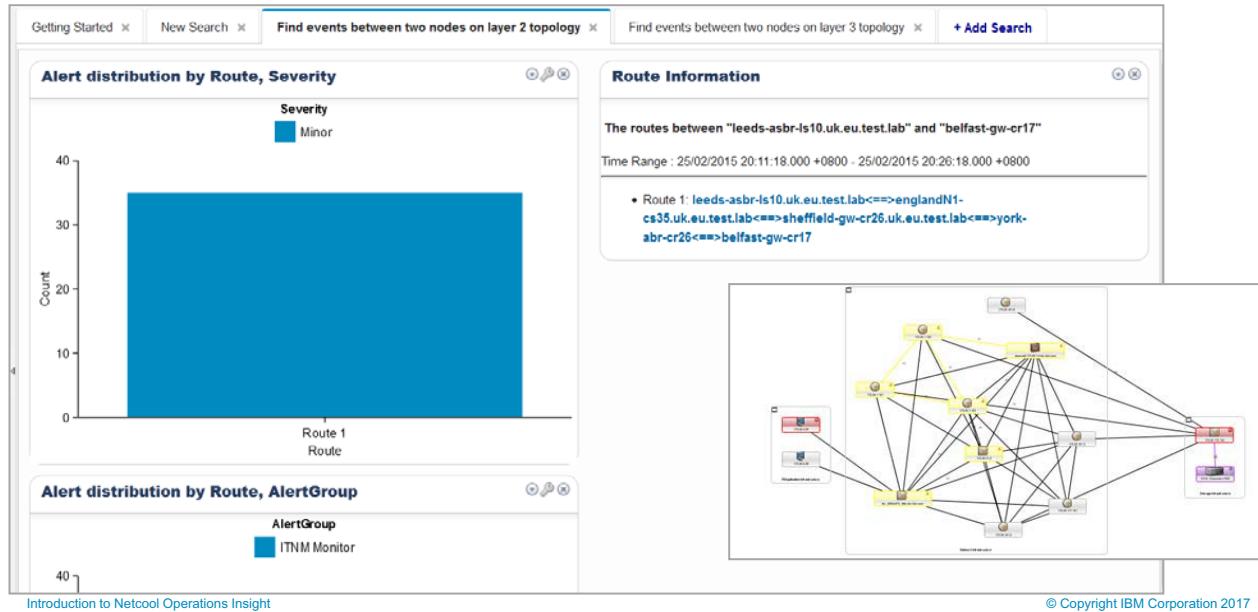


Figure 1-14. Topology search (2 of 2)

These tools can be started from the following locations:

- IBM Tivoli Network Manager topology view
- Netcool/OMNibus Web GUI Event Viewer
- Operations Analytics Log Analysis

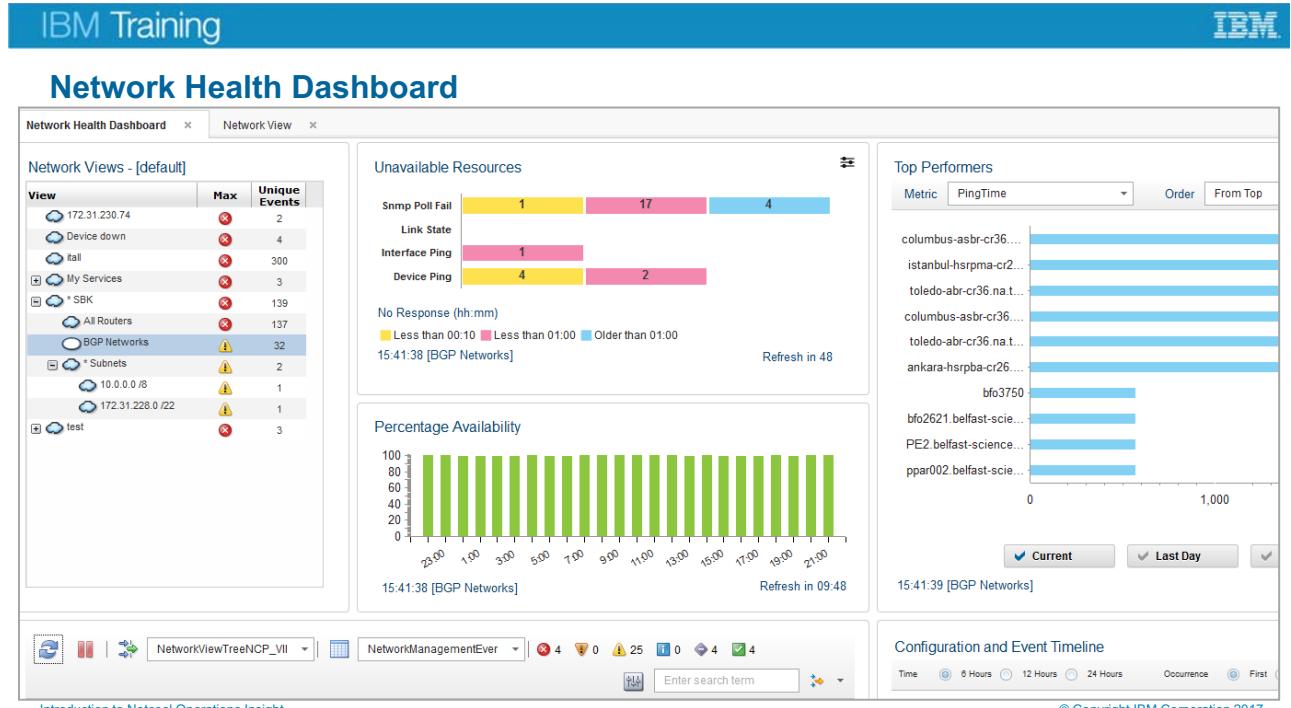


Figure 1-15. Network Health Dashboard

Select a network view, and all the other page components show data from the selected network view. Displayed objects can be clicked to update other frames in the view. The Network Health Dashboard includes the following features:

- List of bookmarked network views for the current user.
- Unavailable resources from the selected network view are shown based on one of the following types of events:
 - SNMP poll failure
 - Link state down traps (or poll results)
 - Interface ping failures
 - Main chassis device ping failures
- Percentage availability graph over time for devices in selected network view.
- Top performers chart or table for polling data where the poll data was stored. Summarization tables are updated at 15-minute intervals.
- Configuration and event timeline shows the correlation between configuration changes that are made by Tivoli Configuration Manager and events.
- Event view for devices in selected network view or for devices that are associated with a just-clicked object in one of the other frames.

Network Performance Insight

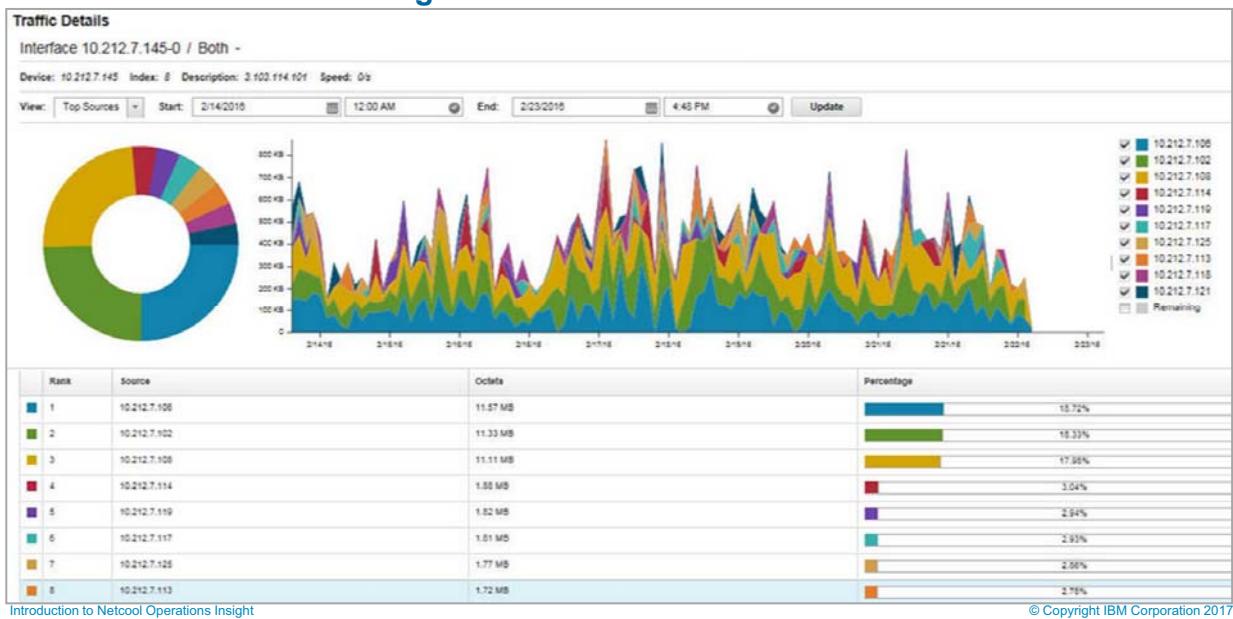


Figure 1-16. Network Performance Insight

The screenshot displays the IBM Agile Operations Management interface for Alert Notification. At the top, there's a navigation bar with links for Manage Notification Policies, Alert Viewer (which is selected), Manage API Keys, and Getting Started. Below the navigation is a header bar with tabs for Alert Viewer (selected), All Alerts, My Alerts, and Alert Notification.

The main area is divided into several sections:

- Alert Viewer:** Shows a table of alerts with columns for State, ID, and What Happened. An annotation box highlights the "SMS" option under "Notification options" for a specific alert, stating: "SMS can be used to acknowledge the alert by replying to the text".
- Alert History:** A table showing a log of notification state changes.
- Mobile App:** A preview of the mobile application interface showing an "Alerts Overview" screen with a list of notifications categorized as Notified, Unnotified, and Mobile App.
- Bottom Right:** Copyright information: © Copyright IBM Corporation 2017.

Figure 1-17. Alert Notification

Three types of runbook automation

1. Manual runbooks

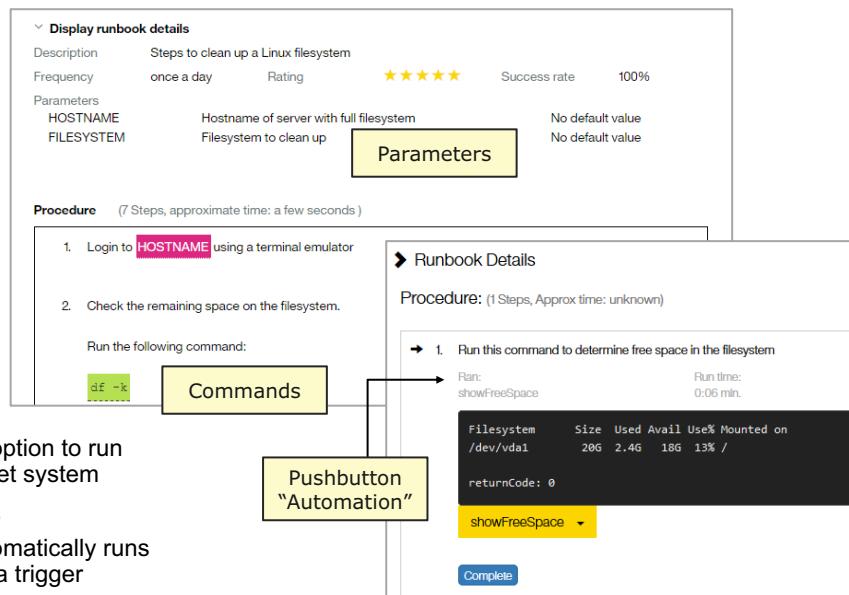
- Each step describes the exact manual procedure that an operator is to follow
- Uses provided network and host management tools that are accessible from the environment

2. Semi-automated runbooks

- Each step describes exactly what an operator is to do for a particular situation
- The operator can select an option to run an automated task on a target system

3. Fully automated runbooks

- The system selects and automatically runs a runbook as a response to a trigger



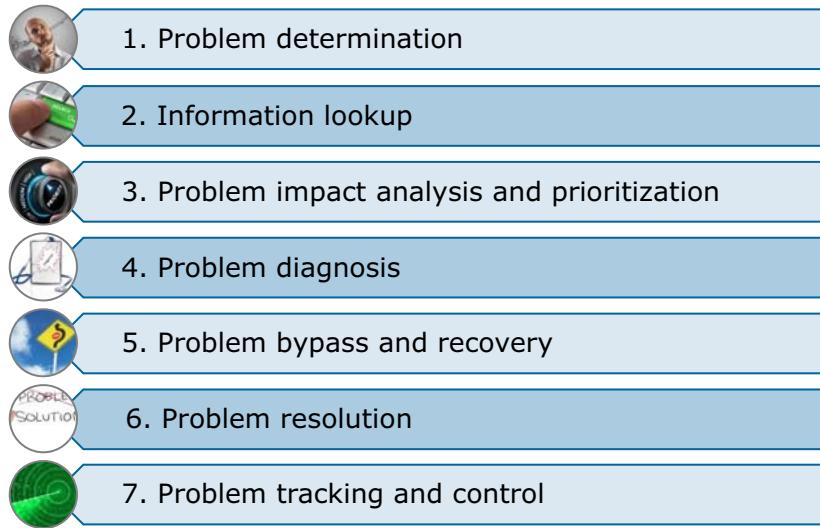
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-18. Three types of runbook automation

ITIL: How to handle technology problems

Information Technology Infrastructure Library



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-19. ITIL: How to handle technology problems

Follow these steps when you investigate any type of infrastructure issue:

- **Problem determination:** Identify whether a problem is real.
- **Information lookup:** Locate and retrieve additional information that can help identify or isolate a problem.
- **Problem impact analysis, prioritization, and SLA management:** Prioritize problems so that limited human resources focus on problems with the greatest impact.
- **Problem diagnosis:** Determine the exact nature of the problem.
- **Problem bypass and recovery:** Since the fix to a problem might take some time, personnel need to determine and implement a workaround plan to restore key services.
- **Problem resolution:** Investigate whether the problem is a recurring one or is the result of a recent change.
- **Problem tracking and control:** Maintain accurate problem and diagnosis records to provide a valuable resource for potential future issues.

The Tivoli Network Management solution is suited to facilitate the implementation and use of these steps.

Problem determination (ITIL Step 1)

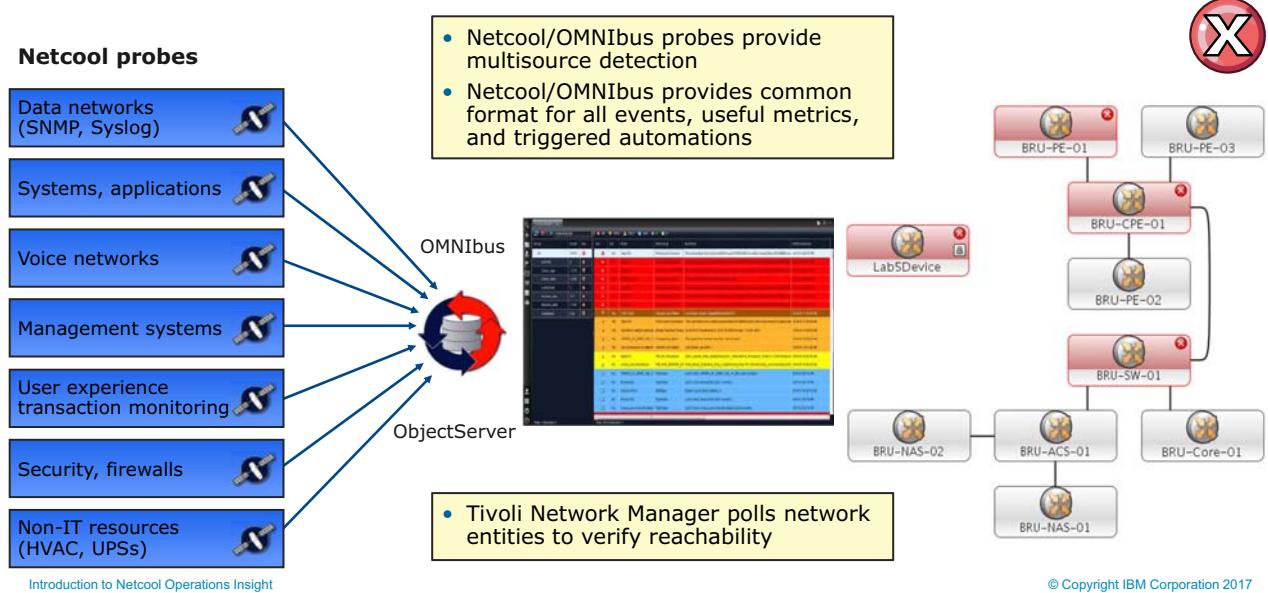


Figure 1-20. Problem determination (ITIL Step 1)

OMNIbus has a library of prebuilt probes that provide solutions for capturing infrastructure issues from various sources. These probes collect data and generate events in a common format. These events are passed to the ObjectServer.

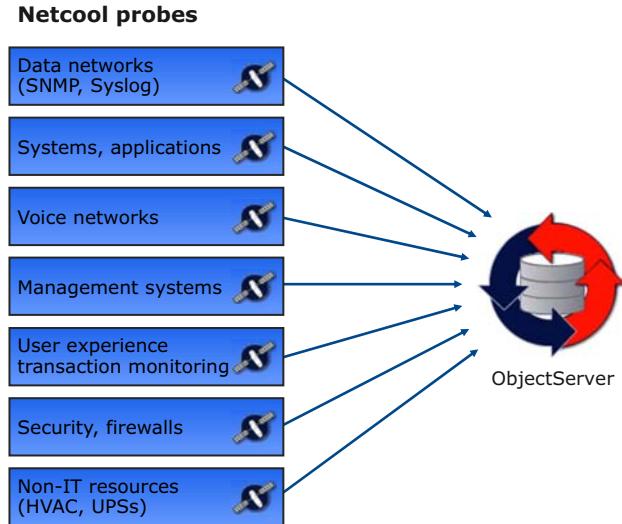
The ObjectServer provides several key features that are related to event processing:

- Data deduplication:** Reduces event volume due to repetitive alarms
- Correlation (generic clear):** Correlates problem and resolution events and automatically clears them, eliminating potential false alarms
- Triggers:** Automated processes that run within the ObjectServer can automatically run commands based on the arrival of specific events

IBM Tivoli Network Manager provides the option of active component polling. This feature is user-configurable and provides continuous feedback based on active polling of infrastructure components.

OMNIbus ObjectServer

- Memory-resident database
- Event repository
- Event data deduplication
- Automations
- Correlation
- Advanced filtering and viewing
- API interface: HTTP and OSLC
- VMware support
- 64-bit support
- Utilities: MIB Manager



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-21. OMNIbus ObjectServer

The ObjectServer is the Netcool/OMNIbus central store. It is a proprietary, memory-resident database designed for the near-term storage of event records. Many issues in your infrastructure can produce multiple events. These events are merely repetitions of the same issue. One of the key features of the ObjectServer is event data deduplication. This feature provides for significant event volume reduction by storing a single copy of an event regardless of how many times it repeats.

Because the ObjectServer runs entirely in memory, it is fast. Checkpoints are written to disk on regular intervals so that event data is recovered if a system failure occurs.

OMNIbus probe operation and integration

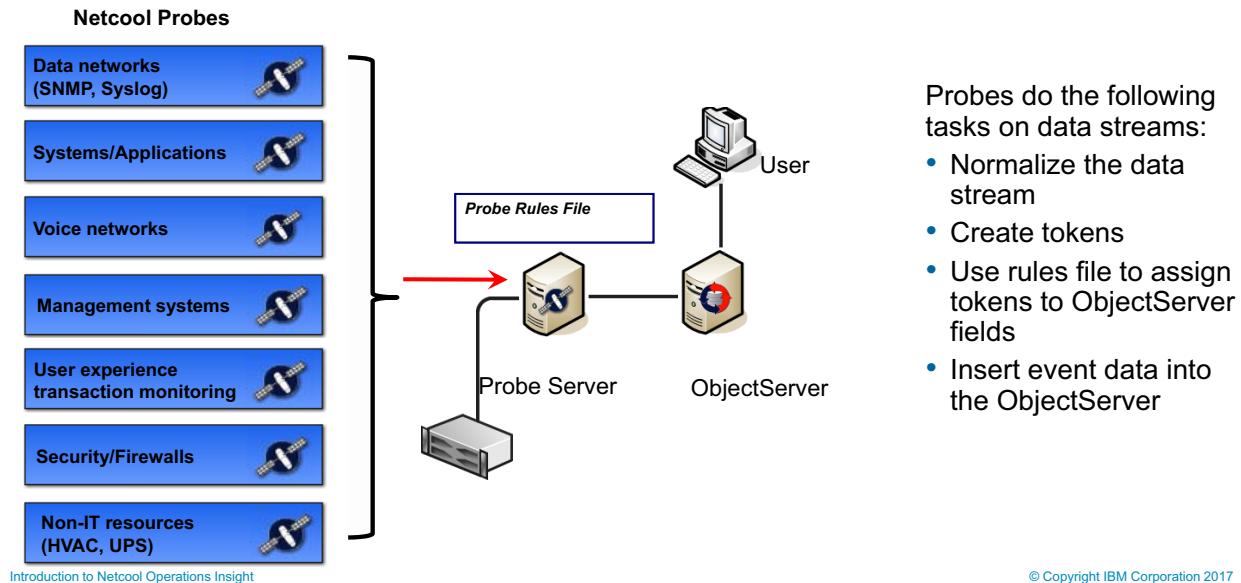


Figure 1-22. OMNIbus probe operation and integration

Probes are software components that are designed to collect data from a specific source. There are currently over 100 unique probes in the Netcool/OMNIbus library. Each one collects data from a specific source: a database table, log file, application output, and others.

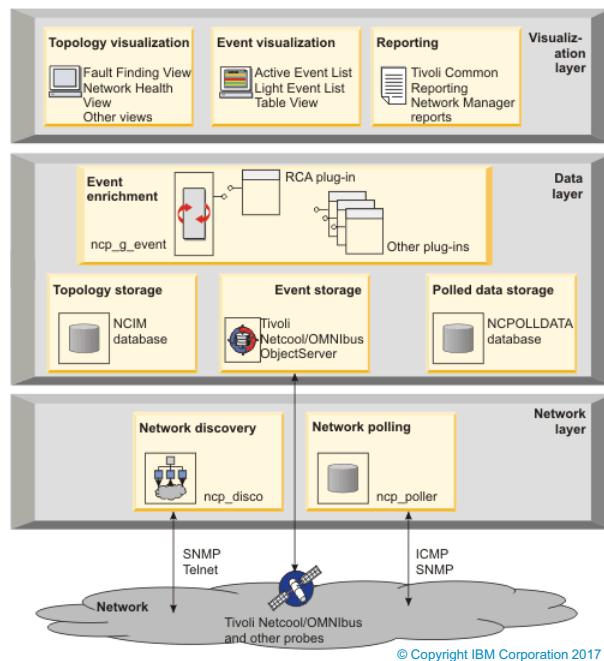
Probes are lightweight, fast, and resilient. Their role is to extract data from the source and format that data into a common format. That data is passed to the Netcool/OMNIbus central store (ObjectServer) through a TCP/IP connection. TCP ensures data delivery. If the connection between the probe and the ObjectServer is broken, the probe can capture data locally, and then forward the data to the ObjectServer when the connection is restored.

A probe rules file parses the information that the probe gathers from the devices that it monitors. The probe rules file normalizes the tokens from the data stream and then puts all messages in a standard format and sends them to the Netcool/OMNIbus ObjectServer.

Tivoli Network Manager

- Network autodiscovery
- Monitoring and polling of discovered devices
- Topology mapping
 - Device details in Device Structure browser
 - Network map views that include OSI Layers 1 – 3, IP subnets, converged topology, PIM, IPMRoute, microwave, and logical RAN
 - Discovery of networks with static network address translation (NAT)
 - Multiprotocol Label Switching (MPLS) virtual private network (VPN) views based on physical connectivity
 - Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), pseudowire, and virtual local area network (VLAN) views indicating how devices function with these protocols
- Network fault isolation in hierarchical and meshed networks

[Introduction to Netcool Operations Insight](#)



© Copyright IBM Corporation 2017

Figure 1-23. Tivoli Network Manager

Network discovery starts in one of these four ways:

- Manual start of a full discovery
- Manual start of a partial rediscovery
- Automatic start according to a predefined schedule
- Triggered start by a network event or other condition

One of the features of IBM Tivoli Network Manager 4.2 is the ability to pinpoint the location of root cause network faults. Tivoli Network Manager designates some events as root cause events and others as system events. This diagnostic process enables NOC personnel to focus on those problems that are affecting your business. This ability is commonly referred to as root cause analysis. However, the term **root cause analysis** is something of a misnomer. The only two root causes for network device faults are device misconfiguration and hardware failure. For that reason, this student guide usually refers to the ability to pinpoint devices that cause multiple network failures as **network fault isolation**. Your instructor might use the terms root cause analysis and network fault isolation interchangeably.

Previous versions of Tivoli Network Manager discovered devices at Layers 2 – 3 of the OSI model. Version 4.2 can also discover Layer 1 devices by running a discovery collector to read information from element managers or a CSV file.

Information lookup (ITIL Step 2)

Tivoli Netcool/Impact can retrieve business information from external sources of data

- It can then do these tasks:
 - Enrich events
 - Do automated diagnostics
 - Escalate events and notify personnel based on defined procedures
 - Use policies to process information and update dashboard views and external databases

Tivoli Network Manager can use database tables to retrieve or store business information during the discovery process

- It can then enrich events with this information, such as these examples:
 - Circuit IDs
 - Customer, region, department
 - Contact, location, and service level agreement (SLA) information

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

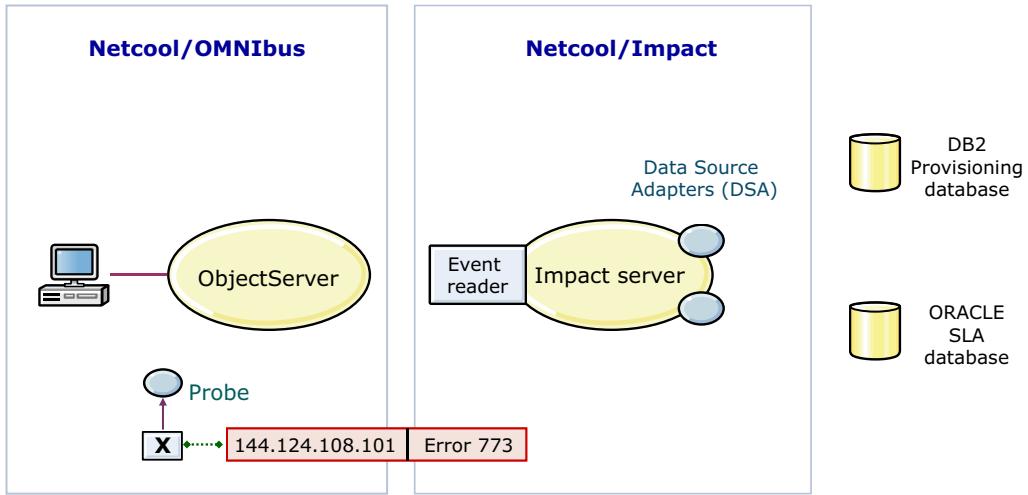
Figure 1-24. Information lookup (ITIL Step 2)

Many customers have a great deal of information that can benefit problem determination. Unfortunately, often this information is not readily accessible. In other cases, it is spread across multiple locations and might not even share a common user interface. Some data is in spreadsheets, some in HTML pages, and other data in database tables.

The Tivoli solution can retrieve this information from the disparate locations and make it immediately accessible. Two features are available to provide this information to personnel.

- **Event enrichment:** ObjectServer event records come with a standard format. Customers have the option of extending this format to add data. Information such as location, circuit ID, point of contact, hours of operation. Several Tivoli components possess the ability to populate these custom fields with customer-specific data, which is retrieved from customer data stores. With the additional data added to the event record, it is immediately available to all users with access to the events.
- **Operator Views:** Netcool/Impact can retrieve data from various electronic sources and publish that data to an HTML page. The page is created dynamically based on receipt of a specific event and data that was extracted from customer data stores. That page can contain basic information like Circuit ID, location, and address. It can also include standard operating procedures for resolving specific issues.

Basic event enrichment (1 of 5)



Introduction to Netcool Operations Insight

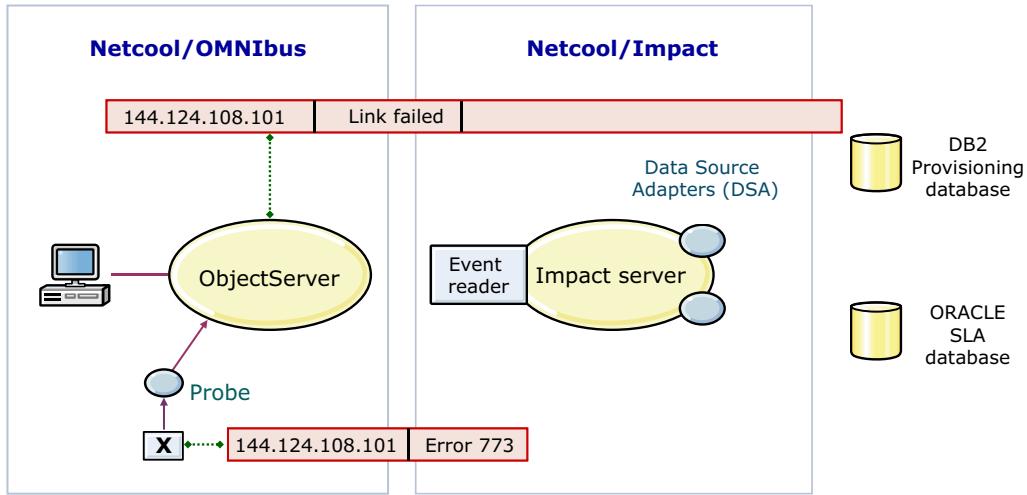
© Copyright IBM Corporation 2017

Figure 1-25. Basic event enrichment (1 of 5)

In this series of slides, you see how event fields from OMNIbus can be enhanced (enriched) by impact to add customer information to the event. Then, the resulting event provides the NOC staff with live information such as customer name, location, and contact details. The Netcool/Impact policies can also reprioritize events according to the SLA of the particular customer.

A probe is a software with a small footprint that passively listens to information that is coming from the devices to which the probe is assigned. In this example, the probe receives a message from a device that says **Error 773**.

Basic event enrichment (2 of 5)



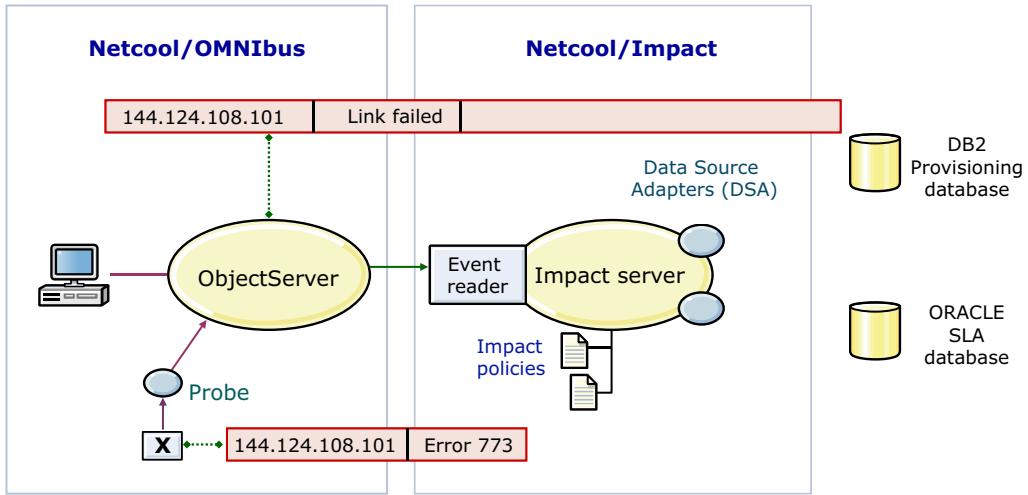
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-26. Basic event enrichment (2 of 5)

The probe rules file parses the **Error 773** message and creates an event in the Netcool Omnibus object server that says **Link failed**. The result is that more useful information is created for the network operations center (NOC) operator.

Basic event enrichment (3 of 5)



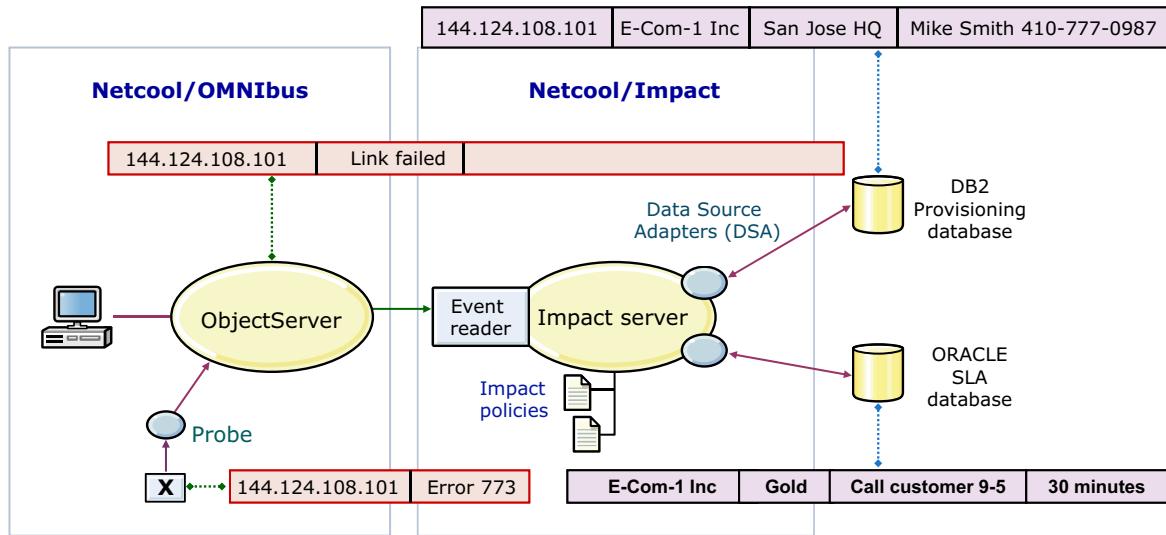
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-27. Basic event enrichment (3 of 5)

The Netcool/Impact server has an Event Reader that receives all events in the Netcool Omnibus server. It applies policies to each event based on filter criteria in the policy. The policy language tells the Impact server what to do with the event. In this example, the Impact server has policies that tell it to access external databases and enrich the event's useful business information.

Basic event enrichment (4 of 5)



Introduction to Netcool Operations Insight

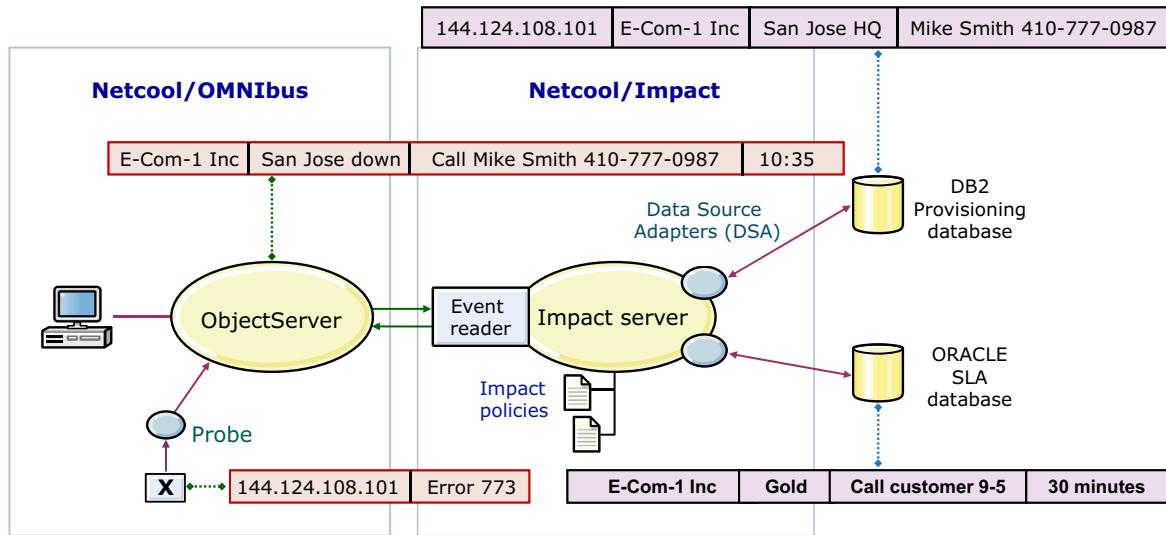
© Copyright IBM Corporation 2017

Figure 1-28. Basic event enrichment (4 of 5)

In this example, the Netcool/Impact server uses its data source adapters (DSA) to first access a DB2 provisioning database. Using the IP address, the policy does a lookup on the DB2 database and establishes the customer, location, and customer contact information associated with that device.

Another DSA accesses an Oracle database to look up the service level agreement (SLA) to determine the level of support in the service commitment for the customer associated with the device that has the error.

Basic event enrichment (5 of 5)



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-29. Basic event enrichment (5 of 5)

The Netcool/Impact server now enriches the event with the information from the external databases. The NOC operator now sees a message that plainly indicates that a specific customer with a specific location has a link that is down. They also know that they must call Mike Smith at the specified phone number by 10:35 in order to avoid an SLA penalty.

Tivoli Netcool/Impact can run much more complex procedures that can also write to external databases other than the Netcool/Omnibus database. However, this example shows how the raw event can be enriched to create actionable business data.

Problem diagnosis (ITIL Step 4)

- Tivoli Network Manager identifies the root cause of connectivity failures so operators can focus on the real problems
- Netcool/OMNibus triggers can run external diagnostic or resolution scripts
- Tivoli Netcool/Impact policies can run automated diagnostics
- The Network Health Dashboard shows whether someone made recent configuration changes that correlate in time to network and device incidents
- Tivoli Netcool Configuration Manager unit of work (UOW) histories show whether it attempted to implement changes recently and identify what was modified
- The Netcool Operations Insight engine looks for patterns of related events from log files and event history and can identify a root cause event whenever these patterns recur
- Netcool Operations Insight can search for related or similar events in a network and create on-demand reports in just 3 – 4 mouse clicks

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-30. Problem diagnosis (ITIL Step 4)

OMNibus and Impact both can run automated commands that are based on receipt of specific events. These commands might do some remedial actions, like rebooting a device, running more diagnostics, querying a device, and recording the result.

IBM Tivoli Network Manager provides a number of capabilities that allow a user to view portions of the network, view device configurations, and issue diagnostic commands.

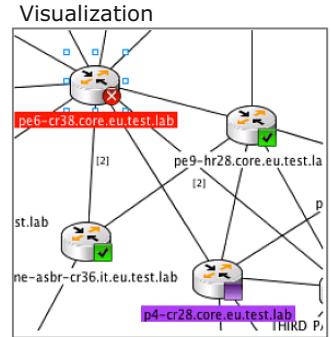
When Tivoli components are integrated with a ticketing system, most of the diagnostic results can be passed directly to the ticketing system.

Topological root cause analysis

- Tivoli Network Manager determines a root cause event when events indicate a loss of connectivity
- Operators can then focus on the real problem

Pre-RCA

NmosCauseType	Node	Summary	Count	NmosManagedStatus	NmosSerial	IfName
Unknown	172.20.1.4	Ping failure on p4-cr28.core.eu.test.lab[0 ...	1	Managed		
Unknown	172.20.1.6	Ping failure on pe6-cr38.core.eu.test.lab[...	1	Managed		
Unknown	172.20.1.6	%LINEPROTO-5-UPDOWN: Line protocol on ...	1	Managed		
Unknown	172.20.1.4	Link Down on node p4-cr28.core.eu.test.la...	1	Managed		
Unknown	172.20.1.4	%LINEPROTO-5-UPDOWN: Line protocol on ...	1	Managed		
Unknown	172.20.1.4	SNMP Poll : Link down on node p4-cr28.co...	1	Managed		
Unknown	172.20.1.6	SNMP Poll : Link down on node pe6-cr38.c...	1	Managed		
Unknown	172.20.1.6	Link Down on node pe6-cr38.core.eu.test.l...	1	Managed		



After RCA

NmosCauseType	Node	Summary	Count	NmosManagedStatus	NmosSerial	IfName
Root Cause	172.20.1.6	%LINEPROTO-5-UPDOWN: Line protocol on ...	1	Managed	0	Gi0/1
Symptom	172.20.1.4	Link Down on node p4-cr28.core.eu.test.la...	1	Managed	38795	Gi0/1
Symptom	172.20.1.4	Ping failure on p4-cr28.core.eu.test.lab[0 ...	1	Managed	38795	Gi0/1
Symptom	172.20.1.4	%LINEPROTO-5-UPDOWN: Line protocol on ...	1	Managed	38796	Gi0/1
Symptom	172.20.1.4	SNMP Poll : Link down on node p4-cr28.co...	1	Managed	38795	Gi0/1
Symptom	172.20.1.6	SNMP Poll : Link down on node pe6-cr38.c...	1	Managed	38799	Gi0/1
Symptom	172.20.1.6	Ping failure on pe6-cr38.core.eu.test.lab[...	1	Managed	38799	Gi0/1
Symptom	172.20.1.6	Link Down on node pe6-cr38.core.eu.test.l...	1	Managed	38799	Gi0/1

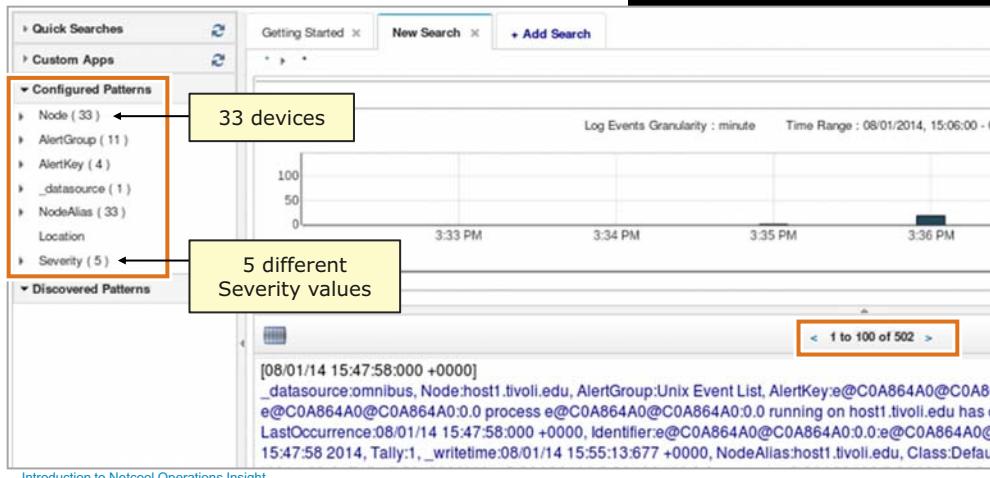
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-31. Topological root cause analysis

Event search

- Log Analysis tools launch from right-click menus of the **Event Viewer** and the **Active Event List**



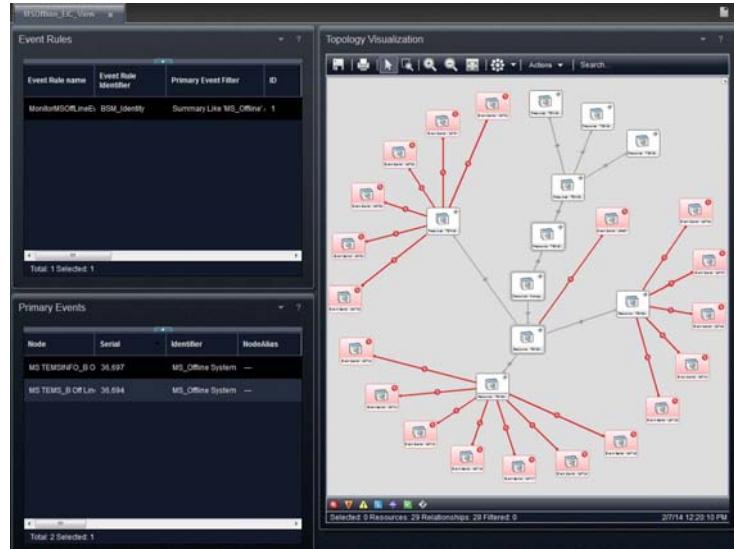
- The context of the event is passed to the Log Analysis search

© Copyright IBM Corporation 2017

Figure 1-32. Event search

Event isolation and correlation

- Correlate events based on known relationships in Service Component Repository
- Imported from Tivoli Business Service Manager, Tivoli Application Dependency Discovery Manager, Discovery Library Toolkit
- Help determine a probable root cause for a selected event
- Reduce mean time to repair



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-33. Event isolation and correlation

Event isolation and correlation determine cause and symptom events based on known relationships.

The known relationships are contained in the Service Component Repository.

The repository can record these relationships in a number of ways.

This graphical chart shows hierarchical relationships within a network. It is not necessary to be able to read the details in the image.

Event Analytics for seasonal event identification

- See related events that might be candidates for suppression
- Identify difficult-to-spot seasonal events that often result in regular periodic problems
- Use visualizations that help you quickly isolate these problems as being more severe or significant
- Provides opportunities for event reduction, thus improving operational efficiency



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-34. Event Analytics for seasonal event identification

No user configuration is necessary to enable the analytics engine to identify seasonal events.

Event Analytics: Seasonality

- Identifies seasonal patterns, such as when and how frequently events occur
- Seasonality analyses are published in reports and graphs so that you can find seasonal patterns
 - For example, an event that periodically occurs at an unscheduled specific time is highlighted
- Use the information from the seasonality reports to create network, device, or suppression rules to reduce the number of events

Node	Summary	Alert Group	Reviewed By	Confidence Level
ducttape.tivlab.rule	TDSBLD: ITM: The lsass process has started.	ITM NT Process		100%
s3w2k.tivlab.aust	TDSBLD: ITM: The lsass process has started.	ITM NT Process		100%
fw2w2k.tivlab.aust	TDSBLD: ITM: The lsass process has started.	ITM NT Process		100%
fw3w2k.tivlab.aust	TDSBLD: ITM: The lsass process has started.	ITM NT Process		100%
rtpbwin1.tivlab.rule	TDSBLD: ITM: The lsass process has started.	ITM NT Process		100%
rtpbwin3b.tivlab.ral	TDSBLD: ITM: The lsass process has started.	ITM NT Process		100%

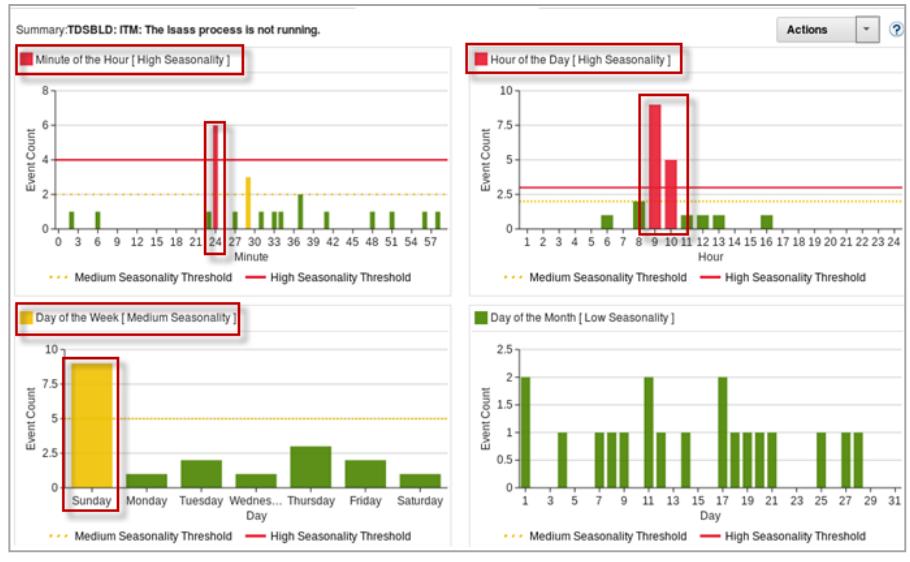
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-35. Event Analytics: Seasonality

Event Analytics for seasonal event identification

- See related events that might be candidates for suppression
- Identify difficult-to-spot seasonal events that often result in regular periodic problems
- Use visualizations that help you quickly isolate these problems as being more severe or significant
- Provides opportunities for event reduction, thus improving operational efficiency



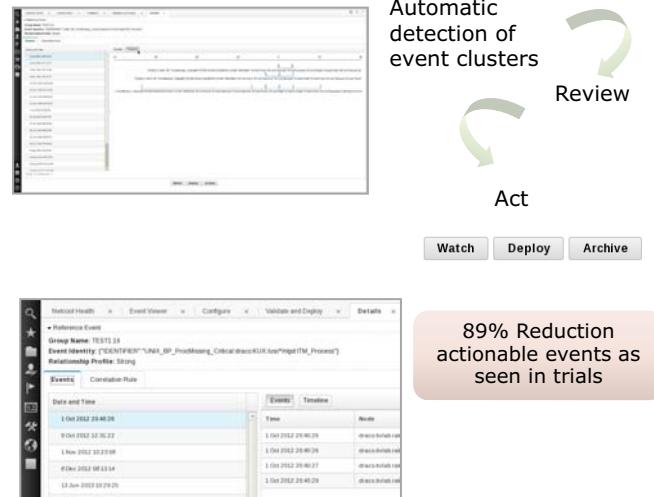
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-36. Event Analytics for seasonal event identification

Related events

- Analytics engine learns what events occur together and notifies the administrator about potential relationships
- The administrator can decide what to do with detected relationships
 - Deploy:** Confirm the pattern and have the analytics engine create a root cause alarm when the pattern recurs
 - Watch:** Look for similar patterns to confirm whether this pattern is deployed
 - Ignore**
- When the events recur, one event is identified as an actionable root cause event
- Uses cognitive learning to analyze historical event archive



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-37. Related events

- The analytics engine in Network Operations Insight identifies patterns and assigns a numerical value to each pattern. That number indicates the degree of certainty that these events are related. The value is derived in part from how many times similar groups of events occurred in the past.
- It notifies the administrator of event groupings and picks the one event that is most likely the root cause.
- The administrator reviews the event grouping. If another event seems more appropriate, the administrator can select a different root cause from the event grouping.
- The administrator then chooses one of the following actions:
 - Deploy** the grouping. If the grouping is deployed, Netcool Operations Insight monitors for the event group to recur. When it does, it creates a new synthetic event that points to a defined root cause. Other events in the grouping are marked as symptom events. When the operator looks at events in the **Related Events** view, the symptom events are indented underneath the root cause events.
 - Watch** the grouping. With this choice, the pattern is saved and the system looks for recurrences of this pattern. Statistics are maintained as to the number of times the event grouping recurs. The administrator can review this data later and decide whether to deploy the patterns.
 - Ignore it.**

The code for analyzing ObjectServer events and log files of all kinds is already included in Netcool Operations Insight. No coding needs to be done to implement the analytics functions. You need to tell the engine where to look for data sources.

Event Analytics: Related events

- Netcool/Impact evaluates historical event
- Determines which events have a statistical tendency to occur together
- Outputs the results on a scheduled basis as event groups
- You deploy valid event groups as Netcool/Impact correlation rules
- The rules act on the event data and show a single parent event from the event group, with all other events in the group as children

Severity	Node	ScopeID	TicketNumber	Summary
 	TLCO-TLCO/BPT-47495	TLCO-TLCO/BPT-47495	55784	INCIDENT: TLCO-TLCO/BPT-47495: 3 sites affected
 	RT0748	TLCO-TLCO/BPT-47495	55784	RT0748: CAUSE AND IMPACT: Operational Warning, inc running on ba
 	RT1297	TLCO-TLCO/BPT-47495	55784	RT1297: Service delivery reported as non-Functional caused by Perform
 	RT1297	TLCO-TLCO/BPT-47495	55784	7705 *~ LAPD FAILURE
 	RT1297	TLCO-TLCO/BPT-47495	55784	7705 *~ LAPD FAILURE
 	RT1297	TLCO-TLCO/BPT-47495	55784	7767 ***- BCCH MISSING
 	VC4282	TLCO-TLCO/BPT-47495	55784	VC4282: CAUSE AND IMPACT: Operational Warning, inc running on ba

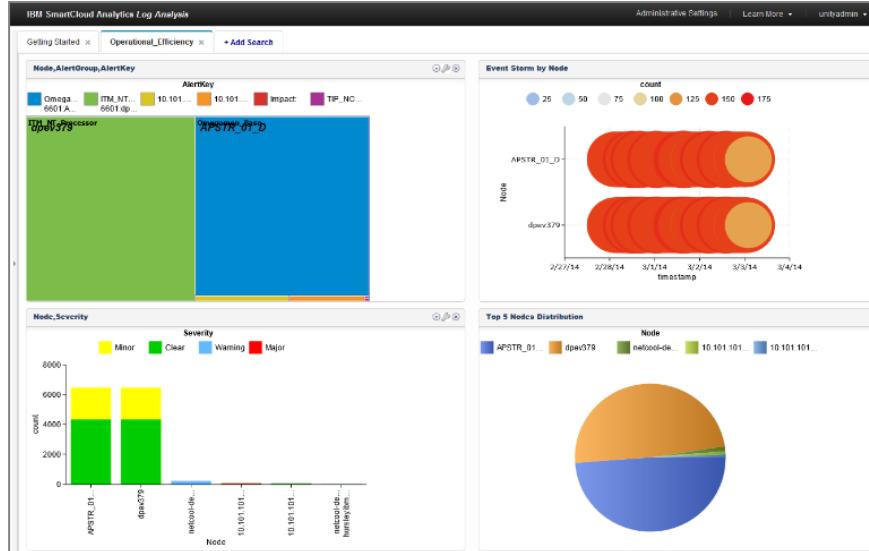
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-38. Event Analytics: Related events

Event Analytics dashboard

- Analytics abilities are included with product
- You can create custom color charts with three mouse clicks and see a graphical representation of tabular data
- Viewing colored charts can let you quickly infer the main problems in your network
- Click an area of the chart to identify the events associated with that area
- Reduces time to resolve the problem (MTTR)



[Introduction to Netcool Operations Insight](#)

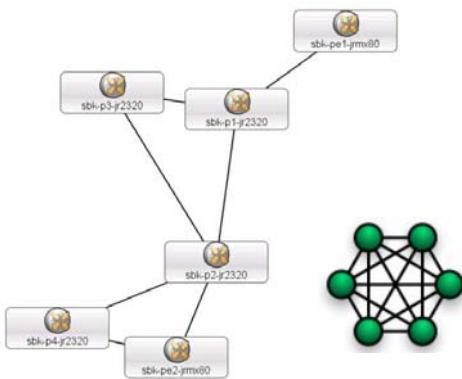
© Copyright IBM Corporation 2017

Figure 1-39. Event Analytics dashboard

One question that usually comes up is: What is the difference between Log Analysis event analysis and Tivoli Common Reporting? While they both accomplish the same thing, Log Analysis is faster, more flexible, and easier to use.

Problem bypass and recovery (ITIL Step 5)

- Reactive
 - Tivoli Network Manager aids in visualization of network structure
 - Use Tivoli Netcool Configuration Manager to implement essential routing changes, defined paths, or neighbor relationships to reroute traffic
 - Tivoli Network Manager polling and Netcool/OMNibus correlation triggers automatically clear problem events when they are resolved
- Proactive
 - Implement fault-tolerant meshed network architecture



Node	Alert Group	AlertKey	Summary	PowerDemand	Last Occurrence(+)	Count
National Grid	Trend Warning	demand	Warning: Demand has been rising for the last thirty minutes	31434	08/02/14 07:43:21	1
National Grid	Trend Warning	demand	Warning: Demand has been rising for the last thirty minutes	33163	08/02/14 08:13:22	1
National Grid	Trend Warning	demand	Warning: Demand has been rising for the last thirty minutes	34863	08/02/14 08:43:22	1
National Grid	Trend Warning	demand	Warning: Demand has been rising for the last thirty minutes	36622	08/02/14 09:13:23	1
National Grid	Trend Warning	demand	Warning: Demand has been rising for the last thirty minutes	37350	08/02/14 09:43:23	1
National Grid	Trend Warning	demand	Warning: Demand has been rising for the last thirty minutes	38338	08/02/14 10:13:24	1
National Grid	Trend Warning	demand	Warning: Demand has been rising for the last thirty minutes	38834	08/02/14 10:43:24	1
National Grid	Threshold Breach	frequency	Clear: Frequency now above 50Hz	39202	08/02/14 12:38:26	3
National Grid	Threshold Breach	frequency	Warning: Frequency below 50Hz	38441	08/02/14 13:28:27	48
National Grid	Trend Warning	demand	Information: Demand has been falling for the last thirty minutes	38328	08/02/14 13:33:27	1

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-40. Problem bypass and recovery (ITIL Step 5)

IBM Tivoli Network Manager network views can assist the user in determining whether alternative routes are available within their network to potentially bypass failed components.

OMNibus correlation (generic clear) automatically correlates problem and resolution events to reduce alarm volume and help eliminate false positives.

Problem resolution (ITIL Step 6)

- Use historical event reports to see how similar problems were handled in the past
- Use Netcool/Impact operator views to display standard resolutions
- Use the Netcool/Impact Maintenance Manager to schedule one-time and repeating maintenance windows to suppress alarms from portions of the network where maintenance is taking place
- Use Tivoli Netcool Configuration Manager to roll back a configuration change if necessary



Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-41. Problem resolution (ITIL Step 6)

Event records archived to a database (like Tivoli Data Warehouse) provide a valuable resource for problem resolution. Often times infrastructure issues repeat. Being able to retrieve historical events might provide an answer to a current problem. The historical event data is retrieved by running a Tivoli Common Reporting report. If the integration to a ticketing system is configured, then the historical event can contain a ticket ID. This data field enables the operator to quickly retrieve the ticket that was associated with that particular event.

Tivoli Netcool/Impact Maintenance Manager

Logged in as user - newbols
23 July 2012 12:16:58

Maintenance Manager

ID	Status	Window Type	Mode	Match Type	Start Time	End Time	Occurs	Node	Alert Group
30	●	One-off	Suppression	EQUALS	20/07/2012 09:00	20/07/2012 10:00	N/A	testz	NULL
37	●	One-off	Suppression	EQUALS	22/07/2012 00:22	22/07/2012 00:30	N/A	Testq	NULL
39	●	One-off	Suppression	EQUALS	23/07/2012 12:00	23/07/2012 12:20	N/A	IMPACT61	NULL
40	●	Days of the week	Suppression	EQUALS	11:00	11:20	Sun Mon Sat	IMPACT61	Databases
41	●	One-off	Suppression	EQUALS	23/07/2012 09:30	23/07/2012 14:00	N/A	NULL	Network
42	●	Day of the month	Maintenance	REGEXP	11:00	14:30	23rd	IMPACT61	Databases
43	●	One-off	Suppression	REGEXP	24/07/2012 09:00	24/07/2012 12:00	N/A	OMNIBus731	ConnectionWatch
44	●	One-off	Suppression	REGEXP	02/07/2012 15:00	02/07/2012 15:00	N/A	Node0001	NULL
45	●	One-off	Maintenance	REGEXP	23/07/2012 12:12	23/07/2012 12:45	N/A	Network0002	NULL
									NULL
									Etc/Greenwich

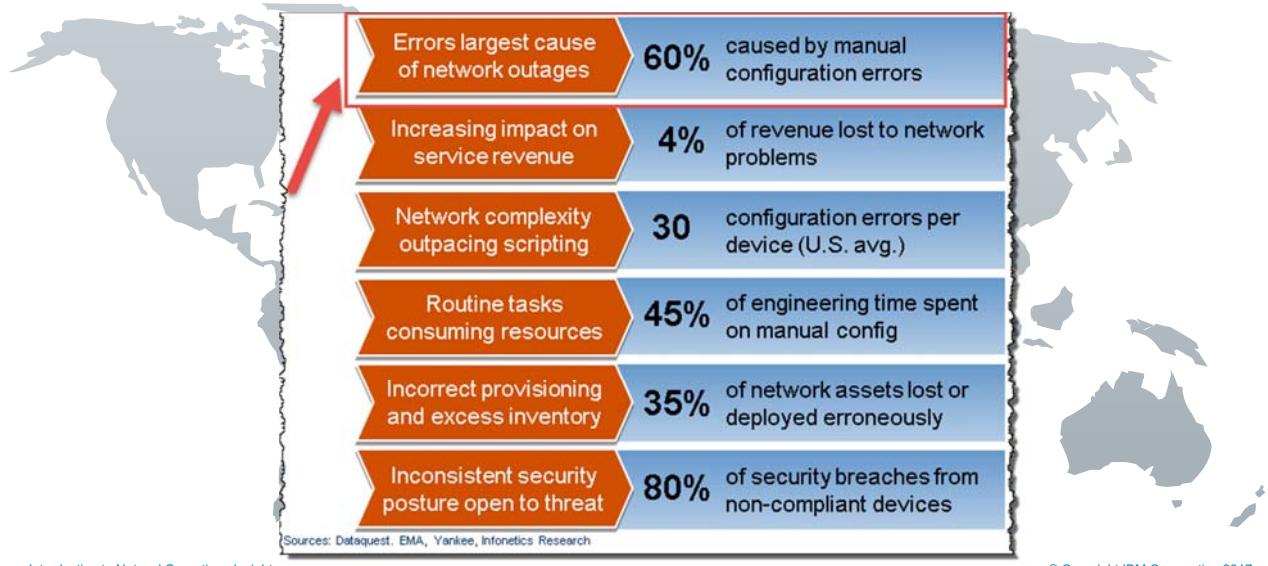
- Schedule one-time or repeating maintenance windows for different portions of your network
 - During a maintenance window, alarms from that part of the network can be suppressed from operator views (an administrator can still view them)
 - When the maintenance window is over, the event suppression is removed
 - If the maintenance operations caused the alarms, the ObjectServer clears most or all of the alarms that were suppressed
 - Impact runs automated diagnostics on some of the events to determine whether they can be cleared automatically
- Events that remain are then treated as other events in your network

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-42. Tivoli Netcool/Impact Maintenance Manager

The need for Tivoli Netcool Configuration Manager



© Copyright IBM Corporation 2017

Figure 1-43. The need for Tivoli Netcool Configuration Manager

Enterprise Management Associates research confirms this data. They report that average problem resolution times are 1 – 4 hours with some problems that last for 2 days. Between 30 minutes and 2 hours are dedicated to detecting and identifying configuration changes that caused problems on systems or infrastructure devices.

IBM Training



View configuration reports to see recent changes and their success or failure

Last "n" device changes, who made the changes and when?

Providing a view of the configured detail

Device terminal audit log – giving a view of the device interaction

Device Unit of Work (UOW) summary

Tivoli Netcool Configuration Manager provides reports that display the following kinds of information:

- For the last **N** device changes, identify who made the changes and when they were made
- Configuration detail view for each device
- Device terminal audit log – shows when out-of-band changes were made and what those changes were
- Device Unit of Work (UOW) summary – shows what schedule changes were completed successfully or failed

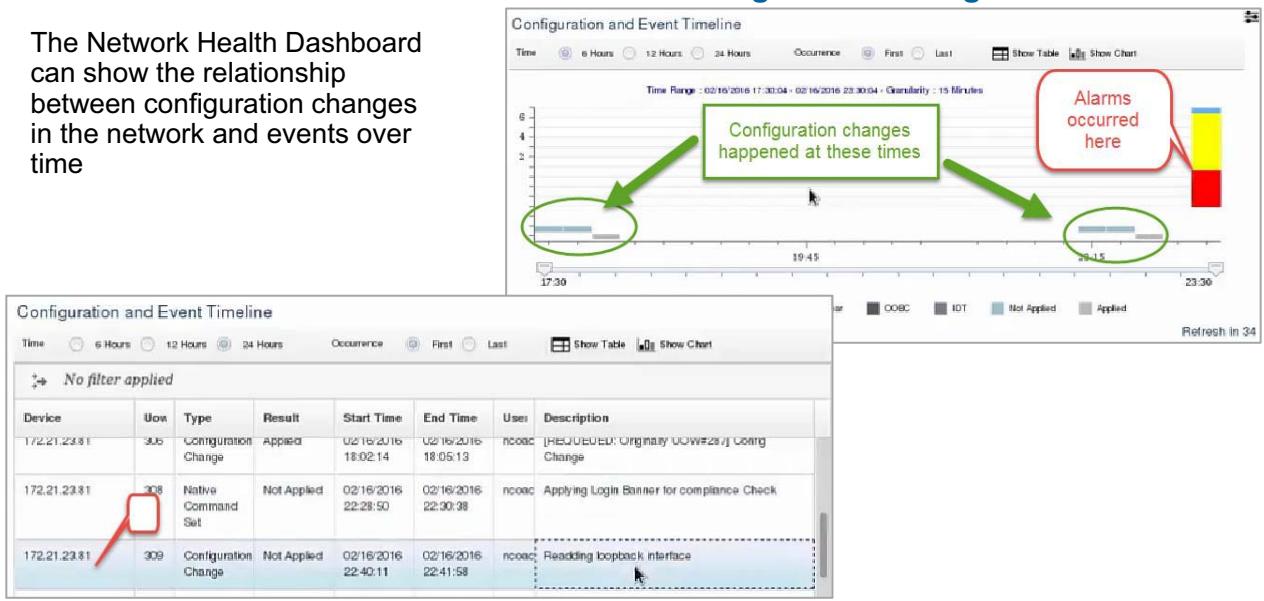
Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-44. View configuration reports to see recent changes and their success or failure

Network Health Dashboard shows time of configuration changes and alarms

The Network Health Dashboard can show the relationship between configuration changes in the network and events over time



© Copyright IBM Corporation 2017

Figure 1-45. Network Health Dashboard shows time of configuration changes and alarms

Use Runbook automations for consistent resolutions

- Manual runbooks give the operator a set of manual steps to follow to resolve a problem
- Semi-automated runbooks prompt the operator do certain steps in an orderly fashion to resolve particular situations
 - The operator can select options to run automated tasks on a target system
- Fully automated runbooks cause a series of actions to run automatically in response to selected event triggers

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-46. Use Runbook automations for consistent resolutions

Problem tracking and control (ITIL Step 7)

Managing of Notification Policy

- Under Manage Notification Policies, select the Create Notification Policy button.
- Here is where a user can customize which recipients receive notifications for what events by entering a Notification Policy Name, predefined and custom rules, and recipients.

The screenshot shows the 'Create a Notification Policy' interface. On the left, there's a sidebar with a red header 'Create a Notification Policy'. Below it, there are two sections: 'Critical DB Issues' (selected) and 'Send notifications to DB2 Administrators'. On the right, there are several configuration options:

- Severity of the alert is Critical or above**: Includes 'DB issues' and a checked checkbox for 'Match all rules'.
- Delay notifications**: Includes a checkbox for 'until [3] identical alerts occur within [60] seconds'.
- Notify these users and groups**: Includes an email address 'Send Me Notifications (sendmenotification@us.ibm.com)' and two radio button options: 'Each time the rules are matched' (selected) and 'First match over an eight hour period'.
- Escalate if the rule is not acknowledged**: Includes a '+' icon for 'Add Escalation'.

At the bottom right of the interface is a copyright notice: © Copyright IBM Corporation 2017.

Figure 1-47. Problem tracking and control (ITIL Step 7)

Many customers use their ticketing systems as the basis for their problem management workflow. As specific operations are done, the ticket is continuously updated with information and can report a different status that indicates the current state of the problem.

When OMNIBus is integrated to the ticketing system, trouble ticket information is returned to the ObjectServer and the event record is updated. All users with access to the event record can immediately see the current state, as related to the ticket.

IBM Training

IBM Connections integration

- IBM Connections is a social networking utility that allows participating stakeholders to receive updates on operational status of network and system components
- Automations or operators can cause certain types of events to be sent to a group of individuals responsible for those devices or that have an interest in the services provided by those devices
 - Automatically update stakeholders with Operations Status
- Notification is driven from the Event List tool or automation

The screenshot shows the IBM Connections integration interface. On the left, there's a dashboard with various icons representing network components like Network, Cloud, Compute, Status, and Storage. To the right of the dashboard is a summary table of events with columns for Priority, Node, and Summary. Below the summary table is a forum titled 'Forums' with tabs for 'Forums' and 'Topics'. It displays three topics from different forums: 'Network Monitoring Agent reported a Problem for site: US', 'Network Monitoring Agent reported a Problem for site: UK', and 'Network Monitoring Agent reported a Problem for site: France'. Each topic has a small icon, the start date ('Started by Frank Adams'), and the number of replies ('0'). At the bottom of the forum section, there are buttons for 'Show: 10 | 25 | 50 | 100 items per page' and a 'Feed for these topics' button.

Introduction to Netcool Operations Insight

© Copyright IBM Corporation 2017

Figure 1-48. IBM Connections integration

Exercise: Review of Netcool Operations Insight architecture

[Introduction to Netcool Operations Insight](#)

© Copyright IBM Corporation

2017

Figure 1-49. Exercise: Review of Netcool Operations Insight architecture

Complete the exercise for this unit.

Unit summary

- Identify the major components in a typical Netcool architecture
- Describe what functions each component of Netcool Operations Insight provides
- Describe how the Netcool solution relates to steps in the Information Technology Infrastructure Library (ITIL) incident management process
- View events that exist in the OMNIbus ObjectServer
- Create a custom event view
- Create a custom event filter

[Introduction to Netcool Operations Insight](#)

© Copyright IBM Corporation 2017

Figure 1-50. Unit summary

Unit 2. Tivoli Network Manager 4.2: Introduction and architecture

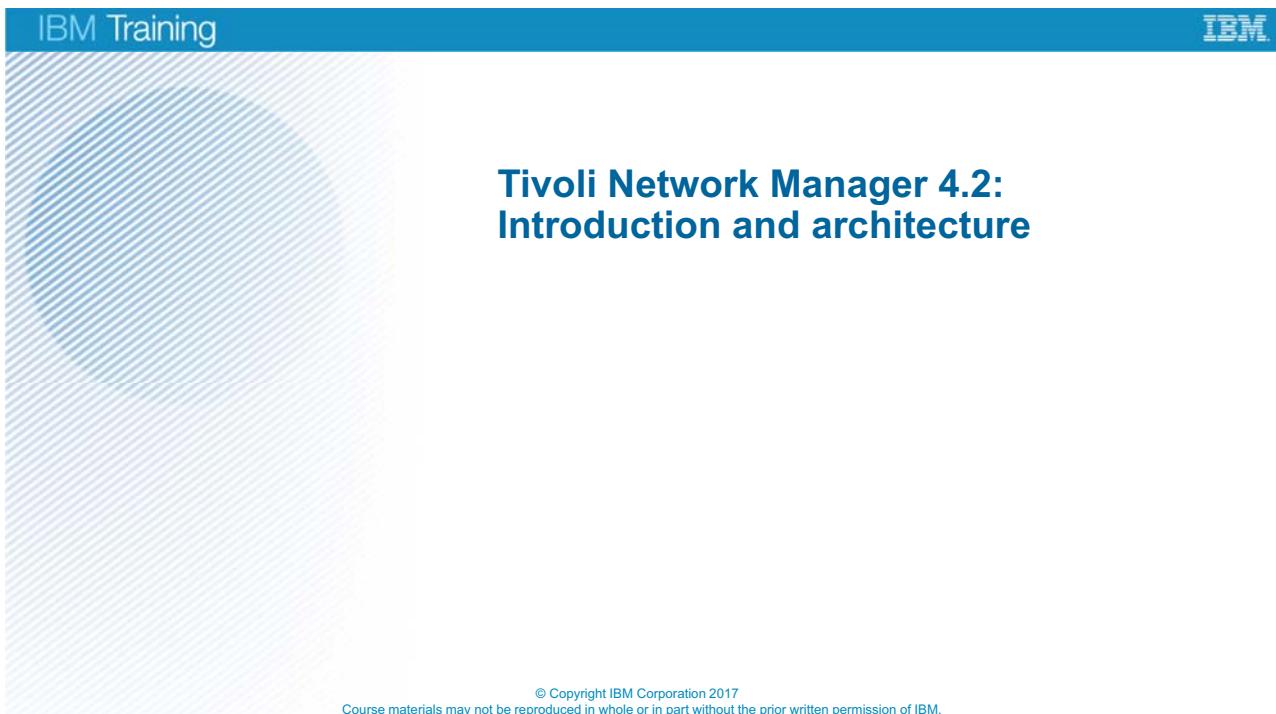


Figure 2-1. *Tivoli Network Manager 4.2 Introduction and architecture*

Estimated time

01:30

Overview

This unit describes the architecture of Tivoli Network Manager 4.2. It then gives generic sizing and configuration guidelines and teaches how to use a key diagnostic tool to find problems in a Tivoli Network Manager installation.

How you will check your progress

Complete student exercises.

Unit objectives

- Explain the key functions of Tivoli Network Manager
- Diagram the key components of Tivoli Network Manager
- List implementation prerequisites
- Determine sizing guidelines for deployment of Tivoli Network Manager

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-2. Unit objectives

2.1. Features overview

This lesson introduces the main features of Tivoli Network Manager and its four functions.

IBM Training



Features overview

After you complete this section, you can do the following tasks:

- List the four functions of Tivoli Network Manager
- Know what script to run to properly set environment variables

Figure 2-3. Features overview

Key functions

- Network autodiscovery
- Monitoring and polling of discovered devices
- Topology mapping
 - Device details in Device Structure browser
 - Network map such as OSI Layers 1 – 3, IP Subnets, Converged Topology, PIM, IPMRoute, Microwave, and Logical RAN
 - Discovery of networks with static network address translation (NAT)
 - Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) views based on physical connectivity
 - Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), pseudowire, and virtual local area network (VLAN) views that indicate how devices function with these protocols
- Network fault isolation in hierarchical and meshed networks

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-4. Key functions

You can run a network discovery by one of the four following methods:

- A full discovery is started manually.
- A partial discovery is started manually.
- Discovery starts according to a predefined schedule.
- Discovery starts when a network event or other condition triggers it.

One of the features of IBM Tivoli Network Manager 4.2 is the ability to pinpoint the location of root cause network faults. Tivoli Network Manager designates some events as root cause events and others as system events. This diagnostic process enables NOC personnel to focus on those problems that are impacting your business. This ability is commonly referred to as root cause analysis. However, the term “root cause analysis” is something of a misnomer. Network errors have two primary root causes.

Network device faults are caused primarily for two reasons:

- Configuration errors
- Hardware failure

Tivoli Network Manager does network fault isolation. It determines what device in your network failed and caused a loss of connectivity to other devices. While network fault isolation is the more correct term, this course uses the terms root cause analysis, RCA, and network fault isolation interchangeably.

OSI model

A seven-layer model of protocol layers, defining interoperability between network devices and software

7 Application

6 Presentation

5 Session

4 Transport

3 Network

2 Data link

1 Physical

- Tivoli Network Manager 4.2 discovery agents interrogate devices at Layers 2 – 3 of the OSI model
- Discovery collectors gather information from supported element management systems (EMS) and comma-separated value (CSV) files about Layer 1 devices

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-5. OSI model

The open systems interconnection (OSI) model describes seven layers of protocols that define interoperability between network devices and software. Tivoli Network Manager IP Edition discovers Layers 2 and 3 of this model with finders and agents. These layers correspond to routers, switches, and the devices that plug into them.

Tivoli Network Manager 4.2 can use data that from an EMS or a comma-separated value (CSV) file to discover Layer 1 devices such as microwave, optical, and RAN devices. Tivoli Network Manager cannot interrogate Layer 1 devices directly.

New features in 4.2 (1 of 2)

- Discovery
 - The DNCIM database is now stored in `$ITNMHOME/embeddedDb/sqlite/ncp_disco.domainName`
 - A new discovery finder **ncp_df_dbentry** reads a database to retrieve a list of devices to find on the network
- Polling and reporting
 - Tivoli Data Warehouse is no longer used to provide polling data summarization
 - Apache Storm is installed with Tivoli Network Manager and is automatically configured to produce summarization data tables for stored polling data
 - Two instances of the poller are configured on installation with binary names **ncp_poller_admin** and **ncp_poller_default**
- The SNMP MIB Grapher now displays only real-time data
 - Historical data can be seen by right-clicking a device in the topology view and selecting **Polling > MIB Info > Show Performance**
- Installation uses the **IBM Installation Manager**

[Tivoli Network Manager 4.2: Introduction and architecture](#)

© Copyright IBM Corporation 2017

Figure 2-6. New features in 4.2 (1 of 2)

Network Manager now uses functions of the Dashboard Application Services Hub within Jazz for Service Management to do the following functions: administer pages, folders, views, widgets, and console preference profiles.

The integration between Network Manager V4.2 and Tivoli Netcool/OMNIbus Web GUI V8.1 is simpler because Dashboard Application Services Hub now hosts both products. The Network Management Integration for Tivoli Netcool/OMNIbus V8.1 is no longer necessary. For more information, see the *IBM Tivoli Network Manager IP Edition Network Visualization Setup Guide*.

Tivoli Network Manager 4.2 features some changes to the polling functions of previous versions.

- The **ncp_poller_admin** binary monitors the size and age of the poll data database tables, updates caches, drops partitions, and supports the MIB Grapher.
- The **ncp_poller_default** binary is used for SNMP and ping polling. In previous versions, the administrator had to configure an administrative poller, but this step is no longer necessary. (If you use more than the default two pollers, some manual configuration is necessary.)

New features in 4.2 (2 of 2)

- Topology display
 - Network Manager now uses functions of the Dashboard Application Services Hub within Jazz for Service Management to administer pages, folders, views, widgets, and console preference profiles
 - Tivoli Integrated Portal is no longer needed and the Network Manager GUI no longer uses Java applets
 - Network Health Dashboard enables Netcool Operations Insight users to monitor and troubleshoot network health
- Reporting
 - Reports previously formatted in the Business Intelligence Reporting Tool (BIRT) format are now supplied in the Cognos format to facilitate editing
 - Some reports are no longer available in version 4.2

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-7. New features in 4.2 (2 of 2)

Platform and browser support

Operating systems

- Intel or AMD Platform Support (64 bit only)
 - Red Hat Enterprise Linux 6 (x86-64)
 - Red Hat Enterprise Linux 7 (x86-64)
 - SuSE Linux Enterprise Server (SLES) 11.0 (x86-64) SP2 and SP3
- IBM PowerPC systems
 - AIX 6.1 iSeries and pSeries
 - AIX 7.1 iSeries and pSeries

Browser support

- Mozilla Firefox 38 (ESR)
- Internet Explorer 10 and 11

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-8. Platform and browser support



Requirements

On AIX, you must meet these installation prerequisites:

- Ensure that either the **unzip** or **GNU tar** utility is installed so that you can extract the installation file. To install these utilities, see the document at <http://tinyurl.com/d44hm3>.
- Ensure that GTK2 is installed. To install GTK2, see the following technote:
<http://tinyurl.com/y6wu3tut>



Requirements

On Red Hat, make sure that these prerequisite modules are installed:

- **libstdc++-4.4.4-13+**
- **libstdc++-4.8.2+**
- **pam-1.1.1**



Requirements

On SuSE Linux servers, you must have the **libstdc++-46-4.6.1+** and **pam-64bit** packages installed.

Database support

- DB2 10.5 ESE (included with Tivoli Network Manager 4.2)
- DB2 10.5 WSE
- DB2 10.1 ESE
- Oracle 12c
- Oracle 11g with partitioning option (only in version 4.2 interim fix 2 and later versions)

Figure 2-9. Database support

Prerequisite software for integration

Core:

- Netcool/OMNibus
 - 8.1 Fix Pack 5 or later
 - 7.4
 - 7.3.1
- Python 2.6 or 2.7 (Linux) or 2.7.5 (AIX) for Apache Storm

GUI:

- Jazz for Service Management 1.1.2.1 or later
- Web GUI 8.1 Fix Pack 4 or later
 - Web GUI requires DASH 3.1.2.1 fix pack 2 or later
- Network Manager 4.2 works with DASH 3.1.2.1 fix pack 2 and 3
- WebSphere Application Server 8.5.5.7 (included with Tivoli Network Manager 4.2) is essential for FIPS 140-2 compliance
- Compatible with WebSphere Application Server 8.5.5.5 or later, including Java Development Kit 7

Tivoli Common Reporting Services:

- 3.1.2.1 or later

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-10. Prerequisite software for integration



Information

Tivoli Network Manager 4.2 can be integrated with Netcool/OMNibus for Windows.

Default environment variables in \$NCHOME/env.sh

This script sets important variables:

```
NC_RULES_HOME=/opt/IBM/tivoli/netcool/omnibus/../etc/rules  
PRECISION_HOME=/opt/IBM/tivoli/netcool/precision  
PWD=/opt/IBM/tivoli/netcool  
LANG=en_US  
ITNMHOME=/opt/IBM/tivoli/netcool/precision  
NCHOME=/opt/IBM/tivoli/netcool  
OMNIHOME=/opt/IBM/tivoli/netcool/omnibus
```

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-11. Default environment variables in \$NCHOME/env.sh

The **ITNMHOME** variable replaces the **PRECISION_HOME** variable that is found in previous versions of Tivoli Network Manager. If you have custom scripts that use the **PRECISION_HOME** variable, add **PRECISION_HOME** to these other variables. Assign it the same value as the **ITNMHOME** variable.

```
PRECISION_HOME=$ITNMHOME; export ITNMHOME
```

The `$NCHOME/env.sh` script is not available until after the product is installed.

Topic summary

After you complete this section, you can do the following tasks:

- List the four functions of Tivoli Network Manager
- Know what script to run to properly set environment variables

Figure 2-12. Topic summary

2.2. Architectural overview

This lesson explains the three layers of Tivoli Network Manager architecture, its internal and external components, and the message broker that enables component communications.

IBM Training



Architectural overview

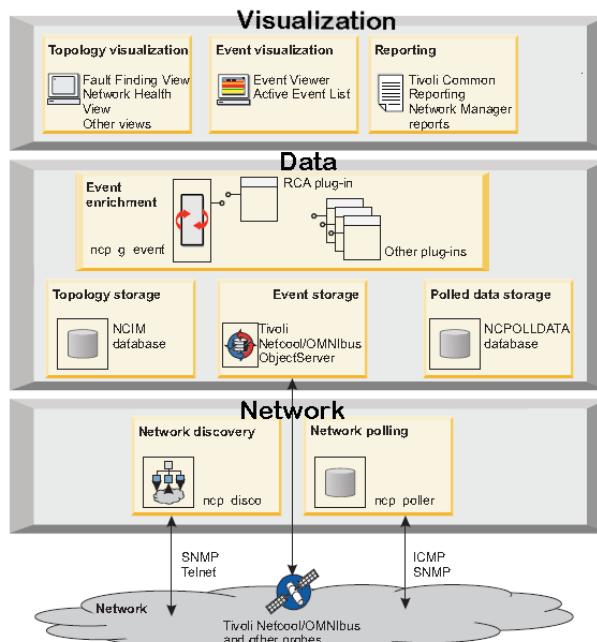
After completing this section, you should be able to do the following tasks:

- Identify internal and external components of Tivoli Network Manager
- List the three layers of the Tivoli Network Manager architecture

Figure 2-13. Architectural overview

Three architecture layers

- Network layer
- Data layer
- Visualization layer
- Communications between various Network Manager core components pass information to other components with the **Really Small Message Broker** (RSMB), which is started automatically



Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-14. Three architecture layers

The three layers are as follows:

- The network layer consists of processes that interact directly with network devices (network discovery and polling).
- The data layer consists of processes that enrich events, do network fault isolation, or store data that is related to topology, events, and polling.
- The visualization layer consists of processes that provide visualization and reporting of topology and event data.

IBM Tivoli Network Manager 4.2 uses the Really Small Message Broker (RSMB) or **nano-broker** to facilitate communications between components. It is part of IBM WebSphere MQ with a small footprint. It uses MQTT protocol (<http://www.mqtt.org>). Publishers send messages to the broker, which then distributes the messages to the subscribers that requested to receive those messages. RSMB is started automatically. If it is stopped, run a discovery to cause the RSMB to automatically restart. The **ncp_brokerlaunch** binary runs internally to start the broker and read its configuration file. You never need to run this program manually. The message broker binary is **ncp_brokerd**. It starts automatically or restarts if any ncp process needs it. The log file \$NCHOME/log/precision/ncp_brokerd_1883.log regenerates if deleted. You can stop the broker with the following commands:

```
cd $ITNMHOME/scripts/perl/scripts
./stop_broker.pl - domain domainName .
```

If you stop the broker without using this script, it restarts automatically.

The configuration file is `$NCHOME/etc/precision/Precision.broker.cfg` and looks similar to this example:

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

Another configuration file has the running port number as part of the file name: `$NCHOME/etc/precision/broker_1883.cfg`. Port 1883 is the default port. (The port number can be changed, but this change is usually unnecessary.)

```
bind_address 127.0.0.1  
port 1883  
trace_level 30  
max_queued_messages 10000
```

Internal components (1 of 3)

- **Internal components** are components within the Tivoli Network Manager core processes that communicate with each other
- **ncp_ctrl**: This control process starts other processes, manages process dependency, and passes command-line parameters to each process as it starts
 - Processes are monitored and restarted when necessary
- **ncp_oql**: IBM Tivoli Network Manager Object Query Language
 - Queries internal Tivoli Network Manager services like CTRL, DISCO, and MODEL
 - Queries DNCIM and NCIM databases with the **-dbId** option
 - Runs with the **-service** command-line option
 - Administrators use this process to query and update data in the Network Manager databases
- **ncp_store**: This process writes critical data to files in the `$NCHOME/var/precision` directory
- **ncp_model**: This process keeps a memory-resident copy of the discovered topology and sends topology data to the Network Connectivity and Inventory Model (NCIM) database

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-15. Internal components (1 of 3)

The operational characteristics of `ncp_ctrl` are defined in the `$NCHOME/etc/precision/CtrlServices.domainName.cfg` file. This file specifies:

- Which processes are started
- The logging levels that are assigned to each process
- The order in which processes are started (a process cannot start until the processes on which it depends are started)

Run the `ncp_oql` command to run queries or do database operations. Student exercises for this course demonstrate the use of several useful operations with Object Query Language.

Internal components (2 of 3)

- **ncp_config:** This process takes user configuration input from the user GUI and configures the necessary files and processes to implement the user's configuration instructions
- **ncp_mib:** Use this administration utility to update your Management Information Base (MIB) data for use within the SNMP MIB Browser
- **ncp_virtualdomain:** This process configures communication between primary and backup Tivoli Network Manager domains to ensure that both servers have identical copies of topology
- **ncp_poller_admin:** This process manages interactions with the polling tables and supports the Real-time MIB Grapher
- **ncp_poller_default:** This process polls network devices with SNMP and ICMP
- **ncp_df_finder-name:** These discovery finders determine a list of extant devices that are subject to further querying during the discovery process:
 - **ncp_df_ping**
 - **ncp_df_file**
 - **ncp_df_collector**
 - **ncp_df_dbentry**

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-16. Internal components (2 of 3)

Internal components (3 of 3)

- **ncp_d_helperserv:** The Helper Server manages the discovery helpers and stores information that is retrieved from the network
 - This discovery helper server receives all requests by discovery finders and agents to query network devices
 - The Helper Server can service the requests directly with cached data or pass on the request to the appropriate helper

Figure 2-17. Internal components (3 of 3)

Helper server components

- **ncp_dh_helper-name:** The discovery agents send queries to the discovery helper server, which uses these helpers to interrogate devices
 - **ncp_dh_snmp:** Conducts Simple Network Management Protocol (SNMP) queries
 - **ncp_dh_xmlrpc:** The collector agent sends XML Remote Procedure Call (RPC) requests to an element management system (EMS) capable of responding to XML queries
 - **ncp_dh_ping:** Receives requests from the ping finder and returns information to it
 - **ncp_dh_telnet:** Receives requests from agents that require telnet or SSH access to devices and returns data to those agents
 - **ncp_dh_dns:** Resolves IP addresses to host names by communicating with host files or Domain Name Services (DNS)
 - **ncp_dh_arp:** Resolves IP addresses to MAC addresses with Address Resolution Protocol (ARP)

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-18. Helper server components

The discovery helper server uses a number of protocol-specific helpers to interrogate network devices. The discovery helper server is the only part of Tivoli Network Manager that interacts with your network devices during the network discovery.

Finders and agents send requests for information to the helper server. The helper server interrogates the device and then sends results back to the finder or agent.

External components (1 of 2)

- **External components** are those Tivoli Network Manager components that communicate with other products such as Tivoli Netcool software, databases, or third-party products
- **ncp_class**: This component sets a **ClassName** for each discovered entity
 - **ClassName** determines the icon representation in the GUI and filtering in polling policies
- **ncp_g_event**: The event gateway provides a bidirectional interface between Network Manager and Tivoli Netcool/OMNibus and forwards events to plug-ins for event enrichment, root cause analysis, or other actions
- **ncp_webtool**: This process makes topology tools available in the topology hop and network partition views
- **ncp_dla**: The Discovery Library Adapter (DLA) collects data on network resources and relationships from Network Manager for import into the Tivoli Change and Configuration Management Database (CCMDB)

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-19. External components (1 of 2)

The process name `ncp_g_event` indicates that the process is an event gateway.

External components (2 of 2)

- **nco_p_ncpmonitor**

- Probe for Tivoli Netcool/OMNIbus
- The poller in Tivoli Network Manager poller sends notification of threshold violations by way of this probe to the Tivoli Netcool/OMNIbus ObjectServer
- The **nco_p_ncpmonitor** process converts these events into ObjectServer format

- **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

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-20. External components (2 of 2)

The process name `nco_p_ncpmonitor` indicates its function:

- The `nco` indicates Netcool/OMNIbus.
- The `_p` indicates a probe.
- The `_ncpmonitor` indicates a Netcool Precision (Tivoli Network Manager) monitor (which gathers polling information).

This probe reports on threshold violations that the Tivoli Network Manager polling engine detects, and it sends these violations as alarms to Netcool/OMNIbus.

The `ncp_trapmux` process is a trap multiplexer.

- Devices can send traps to a destination where this process is running.
- The `ncp_trapmux` process then forwards trap information on to multiple destinations.

Normally, all devices are configured to send traps to locations where the `nco_p_mttrapd` probe is running. However, some customers have other software that is deployed to which they want to send raw SNMP traps. The trap multiplexer can receive the traps and send them to both the `nco_p_mttrapd` probe and to other destinations. This feature is not commonly used. A configuration file specifies the trap destinations.

Topic summary

After you complete this section, you can do the following tasks:

- Identify internal and external components of Tivoli Network Manager
- List the three layers of the Tivoli Network Manager architecture

2.3. Troubleshooting with log and trace files

This lesson shows several methods for dynamically increasing logging levels on running processes. Every Tivoli Network Manager process generates a log file and trace file. By increasing logging levels, you get more detailed information to use in troubleshooting problems with Tivoli Network Manager.

IBM Training



Troubleshooting with log and trace files

After completing this section, you should be able to:

- Increase the logging level for key Tivoli Network Manager processes
- Name the location for Tivoli Network Manager log files
- Identify the configuration file that is used to define process control

Figure 2-22. Troubleshooting with log and trace files

Troubleshooting with log files

- Each process has two log files in `$NCHOME/log/precision`
 - `*.log` contains general information about the process that is starting and stopping
 - `*.trace` contains all information in `*.log` and information about what the process is doing, what data is being processed, and any errors that are encountered during that data processing
- For example, if you have a process that is called `ncp_model` and runs in the ITNM4GO domain, you have the following log files that are associated with that process:
 - `$NCHOME/log/precision/ncp_model.ITNM4GO.trace`
 - `$NCHOME/log/precision/ncp_model.ITNM4GO.log`
- You can use three methods to adjust the amount of detail that goes into these two files for each process:
 1. Change the startup parameters for specific processes in `CtrlServices.cfg`
 2. Send a `SIGUSR` to dynamically update logging detail level
 3. Use an Object Query Language (OQL) statement to update `services.inTray` (in the CTRL service)

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-23. Troubleshooting with log files

Each Tivoli Network Manager component creates two log files when it starts. These log files can be found in the `$NCHOME/log/precision` directory.

The file name of each file includes the process name and the domain name. For example, if you have a process that is called `ncp_model` running in the ITNM4GO domain, you have the following log files that are associated with that process:

- `$NCHOME/log/precision/ncp_model.ITNM4GO.trace`
- `$NCHOME/log/precision/ncp_model.ITNM4GO.log`

The `.log` file contains general messages that indicate processes are starting or stopping. These files have little diagnostic use beyond that information.

The `*.trace` file contains all of the information found in the `.log` file. It also contains information about how that process is handling data and reports on errors it encounters. The `*.trace` file is where you find the most useful diagnostic information.



Note

Some processes limit trace file size and roll previous messages into a separate file. For example, `ncp_mib_0.trace` contains current poller data while `ncp_mib_1.trace` contains previous poll data rolled into an archive file.

Set logging levels in CtrlServices.cfg

- When **ncp_ctrl** starts, the level of detail for logging (to ***.log** and ***.trace** files) is set in **\$NCHOME/etc/precision/CtrlServices.domainName.cfg**
 - `["-domain", "$PRECISION_DOMAIN", "-latency", "150000", "-debug", "0", "-messagelevel", "warn"]`
- The **-debug** option affects the level of detail in the ***.trace** file
 - It accepts values of: **0, 1, 2, 3, 4**
- The **-messagelevel** option affects the ***.log** file
 - It accepts values of **warn, info, debug, error**
- The settings in the **CtrlServices.cfg** file determine the levels of processes when they start
 - This file cannot be modified dynamically
- Inside this file:
 - Process names that are inserted into the **services.intray** table are entered into process control
 - If the process stops for some reason, the **ncp_ctrl** binary actively monitors and automatically restarts them
 - If any processes are started in this file that are not inserted into **services.intray**, they start but are not put into process control
 - If such a process stops, the **ncp_ctrl** binary does not attempt to restart it

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-24. Set logging levels in CtrlServices.cfg

Changes made to the **CtrlServices.domainName.cfg** file are persistent. The changes become effective only when Tivoli Network Manager is restarted. The easiest way to restart Tivoli Network Manager is to use the following commands in order:

- `itnm_stop -domain domainName`
- `itnm_start -domain domainName`

Dynamic updating of logging levels with a signal

- Dynamically change the detail level of these files with the process ID number of the process
 - `kill -USR1 process_ID` (changes the level of the `*.log` file)
 - `kill -USR2 process_ID` (changes the level of the `*.trace` file)
- The change in detail level is reflected in the log or trace file for an individual process
 - `kill -USR1 1319` (each repetition increases debug level until it goes back to the original level)


```
Information: SIGUSR1 received: Logging level changed to 'info'
Information: SIGUSR1 received: Logging level changed to 'debug'
Information: SIGUSR1 received: Logging level changed to 'error'
Information: SIGUSR1 received: Logging level changed to 'warn'
```
- Summary
 - Simplest and most useful method for quick troubleshooting

Figure 2-25. Dynamic updating of logging levels with a signal

The level of detail that is captured in trace or log files can be adjusted dynamically by sending a signal to the process ID for the specific process. For example, if you want more detailed information about the `ncp_disco` process that is running under process ID 1962, run the following commands:

```
kill -USR1 1962
kill -USR2 1962
```

- The USR1 signal increments the detail of the `ncp_disco.domainName.log` file. Each time that you run the `kill -USR1 process_ID` command, you increment the level of logging to the next level. When you reach the highest level of detail, the next use of the command causes the logging level to return to the lowest level. Available levels include: **warn, info, debug, error**.
- The USR2 signal increments the detail of the `ncp_disco.domainName.trace` file. Each time that you use the `kill -USR2 process_ID` command, the process trace level increments to the next available level. When you reach the highest level of detail, the next use of the command causes the logging level to return to the lowest level. Available levels include: **0, 1, 2, 3, 4**.
- This ability to dynamically change logging level does not require you to stop, reconfigure, and restart the software. This method of changing logging levels is the simplest and most useful method for quick troubleshooting.

Example of dynamic logging level updates

```
kill -USR2 8254 (each repetition increases debug level until it goes back to the original level)
```

```
Received SIGUSR2: Switched to debug level 1
Received SIGUSR2: Switched to debug level 2

Received SIGUSR2: Switched to debug level 3
RDScanForAgents starting
RDFindDeletedAgents starting
RDScanForAgents ending

Received SIGUSR2: Switched to debug level 4
2011-07 17:32:22: itnm39server:8254 -> CRivHeartbeatClb::OnTimer mem usage:
 50597888 bytes
2011-07 17:32:22: itnm39server:8254 -> CRivHeartbeatClb::OnTimer mem usage:
 50597888 bytes
```

Figure 2-26. Example of dynamic logging level updates



Hint

If you dynamically update **services.inTray** to get a specific level of log or trace files, wait at least 5 seconds for the process to update to that level. Avoid being impatient and sending the signal again.

Debug level 3 is the level of debugging that is most useful for onsite personnel and is usually sufficient for technical support.

Updating services.inTray to set levels

- Step 1
 - `ncp_oql -domain ITNM4GO -service ctrl`

- Step 2: Setting trace level
 - `update services.inTray set traceLevel=4 where serviceName='ncp_model'`

- Step 3: Setting log level
 - `update services.inTray set LogLevel='error' where serviceName='ncp_model'`

- Summary
 - Not useful because of complexity

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-27. Updating services.inTray to set levels

services.inTray logging detail parameters

```
select * from services.inTray where serviceName='ncp_model'
{
    serviceName='ncp_model';
    servicePath='$PRECISION_HOME/platform/$PLATFORM/bin';
    domainName='SBTN2010';
    argList=['-domain','$PRECISION_DOMAIN','-latency','60000',
        '-debug','0','-messagelevel','warn'];
    dependsOn=['ncp_config','ncp_store','ncp_class'];
    retryCount=5;
    serviceId=3;
    traceLevel=4;
    logLevel='error';
    serviceState=4;
    interval=9;
    processId=30418;
}
```

- The **ncp_model** service started with the **trace** at level 0, and the **messagelevel** is set to **warn**
- These query results show that the original logging levels **changed**

Figure 2-28. services.inTray logging detail parameters

Topic summary

After you complete this section, you can do the following tasks:

- Increase the logging level for key Tivoli Network Manager processes
- Know the location for Tivoli Network Manager log files
- Identify the configuration file that is used to define process control

Figure 2-29. Topic summary

2.4. Architectural consideration

This unit explains the reasons necessary to create more network domains. It also gives several examples of sizing considerations to evaluate when implementing Tivoli Network Manager.

IBM Training



Architectural consideration

After completing this section, you should be able to:

- List the four components that must be present for a complete installation of Tivoli Network Manager
- List reasons for creating more Tivoli Network Manager domains
- List protocols that are necessary to do a network discovery
- Describe how the ObjectServer differentiates between events from different Tivoli Network Manager domains

Installation objective

- During a Tivoli Network Manager installation, the following four components are installed:
 - Network Manager core processes (discovery, polling, root cause analysis, and event enrichment)
 - NCIM database for storing topology data
 - Tivoli Netcool/OMNibus (event management software)
 - The Jazz for Service Management dashboard and Web GUI provide the user interface framework and web applications
- The objective of the installation is to place these components on one or more servers

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-31. Installation objective

You can configure Tivoli Network Manager architecture in single-server and multiple-server architectures.

- The software can run on a single server or multiple servers.
- It can connect to a single ObjectServer for all domains, or it can connect to an ObjectServer for each domain.
- It can run in a stand-alone mode or in a failover architecture.
- Monitoring and polling can occur from a single point, or these tasks can be distributed among several computers.

Whatever architectural configuration you use, these same four components must be a part of every installation:

- Tivoli Network Manager core components
- The Network Connectivity and Inventory Model (NCIM) database for storing topology information
- A Tivoli Netcool/OMNibus ObjectServer
- The Jazz for Service Management dashboard and Web GUI

Sizing considerations

- The following are typical Network Manager deployment configurations:
 - Single-server deployment
 - Distributed deployment: two or more servers
- These factors might require an increased number of servers in a distributed deployment:
 - Active event rates
 - Amount and rate of stored polling data
 - Device status polling rates and number of polling targets
 - Network response times for polled targets
 - Discovery frequency
 - Size of the network

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-32. Sizing considerations

A number of factors determine the deployment architecture and hardware requirements for a Tivoli Network Manager installation. If you have a complex network environment, you might want to engage professional services to properly design a solution to fit your needs.

Multiple domains

- A **domain** is a collection of processes that are associated with Tivoli Network Manager
- In some circumstances, it can be necessary to configure Tivoli Network Manager IP Edition services to manage more than one domain
 - Overlapping IP addresses cannot occur within the same domain
 - Some companies require separation of topologies into separate domains for management reasons
 - Customers can separate discovered devices into separate domains to reflect business, logical, or geographic divisions
 - You can put devices into separate domains to reduce the time that is necessary for discovery and network fault isolation
 - Network fault isolation occurs only within a contiguous map inside of a single domain

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-33. *Multiple domains*

A domain is a collection of processes that are associated with Tivoli Network Manager. Each domain that runs on a server uses a separate collection of processes. For example, if you have two domains that are running on your system, two copies of each Tivoli Network Manager process run on the server.

Other architectural considerations

- Tivoli Netcool/OMNibus is the event manager for collecting alerts from network devices
 - The volume of events in your environment can determine the necessary number of OMNibus ObjectServers
 - Event volume, geographic concerns, and business objectives can all influence whether you have a tiered OMNibus architecture

- The Tivoli Network Manager server requires access to the network devices you want to discover
 - The Tivoli Network Manager server must be included in the access control lists (ACLs) for network devices
 - Tivoli Network Manager must be able to authenticate to element management systems (EMS) to gather information with a collector
 - Discovering network devices on the other side of firewalls requires the firewall to pass certain protocols through the firewall: DNS, ARP, SNMP, Telnet, or Secure Shell (SSH), ICMP

[Tivoli Network Manager 4.2: Introduction and architecture](#)

© Copyright IBM Corporation 2017

Figure 2-34. Other architectural considerations

When you design a Tivoli Network Manager architecture for your environment, you need to consider access and security as a part of the architecture.

- Network devices with access control lists (ACLs) must include the Tivoli Network Manager to enable the server to interrogate the devices.
- If you are discovering devices on the other side of the firewall, you need to coordinate installation with your firewall administrator.

For discovery to work properly, you need the following protocols to be enabled through the firewall:

- DNS
- ARP
- SNMP
- Telnet or Secure Shell (SSH)
- ICMP

Planning deployments

- Domains
 - What is a domain?
 - When do you need separate domains?
 - Sizing issues
 - Overlapping IP addresses
 - Organizational needs
 - Implications of multiple domains on root cause analysis
- Location
 - Access lists, firewalls
- File and directory structure
- To see a sizing table, classification of networks, and sample architectures, see *IBM Tivoli Network Manager IP Edition: Product Overview*
 - This document and other Tivoli Network Management 4.2 documents at: <http://tinyurl.com/z4hqc6x>



Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-35. Planning deployments

When you plan a deployment, coordinate your efforts with personnel familiar with the configuration of network device access lists and firewalls. A successful discovery requires access to the devices on your network so they can be properly interrogated.

Planning and deployment also include determining the number of discovery domains that are used. Several conditions might indicate a need for more domains:

- Large networks can require long discovery times. You can shorten your discovery times by partitioning your network into multiple domains. Each domain can then be discovered in a fraction of the time.
- You can run the network discovery from more than one location. One set of core domain processes must exist on the hard disk of each server.
- Putting devices in separate domains can also reduce the time that is necessary for network fault isolation.
- Your network exceeds a certain size. See product documentation for the optimum number of network entities per domain.
- Your network contains overlapping IP addresses. Because the IP address is a key field within the topology database, you cannot have two devices with the same IP address in the same domain.
- A managed service provider (MSP) can easily have two customers with the same IP address space. In this case, the MSP has two alternatives:
 - Put each customer in a separate domain.
 - Discover one of the customer networks from the public side of the firewall and use network address translation (NAT) for one of the customers. This method causes each customer's network devices to be represented in the topology database with a unique IP address. The

actual IP addresses are also stored in the topology database, but the unique translated IP addresses are the key field.

- Operational boundaries dictate the need for multiple domains. Examples of operational boundaries include geographical boundaries and security boundaries. One group in an organization can specify that they want to see only devices in their network map for which they are responsible. Security considerations can require that key network devices are visible only to authorized personnel. These types of considerations are referred to as Layer 8 issues because they are beyond the seven layers of the OSI model.

Putting network devices in separate domains can affect network fault isolation.

- By default, when Tivoli Network Manager is determining whether events are root cause or symptom events, it considers only devices within a single domain.
- Root cause analysis does not consider devices in one domain to be causes or symptoms of events in another domain.
- Some customers choose to include key devices on their core network in each separate domain they discover. This configuration enables Tivoli Network Manager to properly identify whether problems in the core network are causing problems in one or more edge networks.

In the following pages, general guidelines for deploying Tivoli Network Manager are provided for certain categories of networks. Tivoli Network Manager has a flexible architecture that enables it to be installed to fit your specific environment.

When determining the number of domains needed, the actual number of domains that are necessary varies depending on various factors, including the agents that are used in the discovery. For example, the **Entity** agent discovers more entities that are represented in the database. An increased number of entities might require more domains to handle the greater amount of data efficiently.

Gather the following data:

- Number of devices in the network
- Average number of interfaces per device



Note

The actual interface count on a specific device can vary considerably from the average interface count. An example of this interface count disparity is found in MPLS networks. Typically, the number of interfaces per core device is high. However, in edge devices, there might be as few as 2 – 3 interfaces per device.

- Apply the following equation to determine an approximate number of network entities:

$$\text{Number of network entities} = \text{Number of devices} * \text{Average interface count} * \text{multiplier}$$



Information

Use a multiplier of 2 for a routed network and a multiplier of 0.5 for a switched network.

Switched networks tend to generate more network entities because they contain VLANs, which contain multiple entities. Apply the following equation to determine the suggested number of network domains:

$$\text{Number of domains necessary} = (\text{Number of network entities}) / 250,000$$



Information

In previous versions of Tivoli Network Manager, it was a good idea to limit the number of network entities to 250,000 for a domain. With Tivoli Network Manager 4.2, plan on 400,000 entities or fewer per network domain to maintain optimum performance. The following examples assume 250 K devices per domain.

Router-centric customer example



Example

A customer has a network with the following characteristics:

Number of devices in the network: 15,000

Average number of interfaces per device: 20

This customer network has approximately 600,000 network entities:

$$\text{Number of network entities} = 15,000 * 20 * 2 = 600,000$$

Based on the following calculation, this network requires three network domains:

$$\text{Number of domains necessary} = 600,000 / 250,000 = 2.4$$

Switch-centric customer example



Example

A customer has a network with the following characteristics:

Number of devices in the network: 1,000

Average number of interfaces per device: 24

This customer network has approximately 84,000 network entities:

$$\text{Number of network entities} = 1,000 * 24 * 3.5 = 84,000$$

Based on the following calculation, this network requires one network domain:

$$\text{Number of domains necessary} = 84,000 / 250,000 < 1$$

Creating new domains

You can add another domain to a server by one of the following methods:

Run the `$ITNMHOME/scripts/perl/scripts/domain_create.pl` script.

Start another instance of **ncp_ctrl** and use the `-domain` option to specify the new domain name.

Demonstration or educational deployment (1 of 2)

- Approximately 25 network devices and key servers combined
 - All devices are in the same location as Tivoli Network Manager server
 - Flat network architecture
 - Multivendor network devices on LAN with fast Ethernet
 - For demonstration purposes, have some devices that are configured to use SNMPv3 and others with IPv6 addresses
- Three GUI clients
- Chassis ping polling and some SNMP polling
- No major Tivoli products are integrated with the system, other than Tivoli Netcool/OMNibus
- Performance reports are needed for short data collection periods (typically five days) to match the length of the training course

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-36. Demonstration or educational deployment (1 of 2)

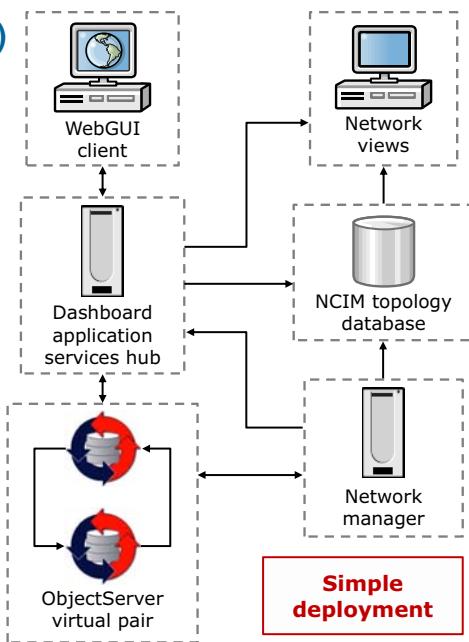
A demonstration or educational deployment can be used for:

- A development lab
- A training classroom
- A customer demonstration center or proof of technology
- Small to medium customer networks with 1000 or fewer network devices

A single server is adequate for this type of deployment.

Demonstration or educational deployment (2 of 2)

- Single-server deployments are appropriate for small demonstration or educational systems, and for systems to support small to medium customer networks with 1000 or fewer network devices
- A single-server deployment must meet the following minimum specification:
 - 2 or more processors of current speeds
 - 3 GHz or better for processors from the Intel product line
 - 4 GB RAM



© Copyright IBM Corporation 2017

Tivoli Network Manager 4.2: Introduction and architecture

Figure 2-37. Demonstration or educational deployment (2 of 2)

A limitation of a single-server deployment is that Tivoli Netcool/OMNibus and Network Manager must be upgraded at the same time. All upgrades to major releases must be done at the same time to avoid compatibility issues. A major release is a release that has its own documentation set. For example, Tivoli Netcool/OMNibus V8.1.0 fix packs can be applied at different times.

Small network description

- Network
 - 150 – 330 network devices and key servers
 - 20 – 30 virtual local area networks (VLANs)
 - Devices are in the same location as Tivoli Network Manager server
 - Network changes occur monthly and a full discovery run
- 3 active GUI clients
- Polling is set to these parameters:
 - Chassis ping polling at 2-minute intervals
 - SNMP polling at 30-minute intervals with 3 – 6 polled MIB values
- No major integration to Tivoli products other than Tivoli Netcool/OMNibus

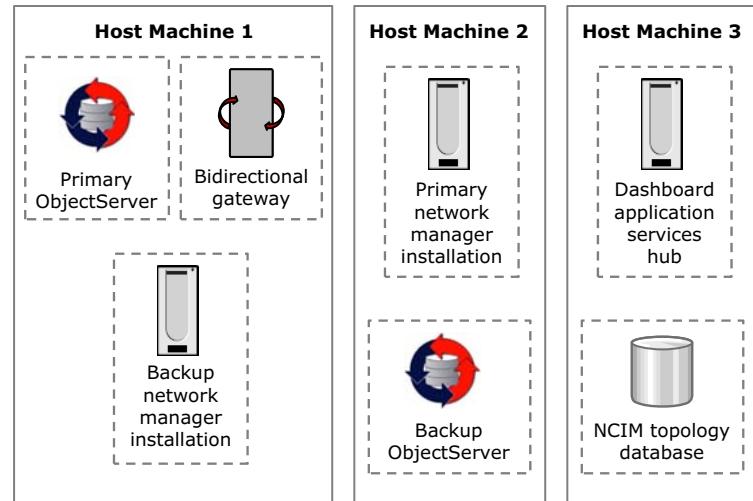
Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-38. Small network description

Small network deployment with redundancy

- Servers need the following hardware characteristics:
 - 4 – 8 GB of RAM
 - Two or more processors
 - RAID 5 disk array
- Configure high availability between ObjectServers and Tivoli Network Manager servers
- The poller runs on the acting primary Tivoli Network Manager server



Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-39. Small network deployment with redundancy

In a small network deployment, a single server with enough memory and more than one processor is sufficient.

- A RAID 5 disk array is not required here. But you need some form of disk redundancy so that persistent data stores are not lost with the failure of a single hard disk.
- Because of the small size of this network, a single poller instance is all that is necessary.

Medium network description

- Network
 - 250 – 1000 network devices and key servers
 - Workstations are not managed
- Devices and servers are distributed among central and remote sites
 - Remote site connects by a wide area network (WAN) connection
 - Local network has Fast Ethernet or Gigabit Ethernet connections
- Polling:
 - Chassis ping polling at 2 – 5 minute intervals
 - SNMP polling at 5 – 15 minute intervals
- Other:
 - 5 – 20 active GUI clients
 - Integration with IBM Tivoli Monitoring with Tivoli Data Warehouse running DB2 to support performance reports
 - Performance reports are necessary for data collection periods on the order of 31 days

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-40. Medium network description

Typically, Tivoli Network Manager is used only to monitor key network devices and servers. While it can poll workstations, most customers choose not to do this type of polling. Default polling policies are used to determine when a loss of connectivity to devices on the network occurs. You can add other custom polls for SNMP variables that are useful to you.

However, Tivoli Network Manager is not designed for robust performance analysis and trending. Other products with the Tivoli suite are better suited to that task. Performance data within Tivoli Network Manager is useful for short-term diagnostic analysis.

Medium network deployment

- Single or dual servers, which are based on adequate memory, polling targets, polling rates, and event rates
 - Single server: 8 processors, 32 GB RAM, RAID 5 disk array
 - Dual server: Four processors for Network Manager server, four processors for Tivoli Netcool/OMNibus and Dashboard Application Services Hub (DASH) server, 8 GB RAM, RAID 5 disk array
- Supported database server:
 - If you have a single database server, the database can be installed on a dedicated server or on one of the others servers that has the lowest processor and memory load
- Two domains
- Polling:
 - Single server: One polling engine
 - Dual server: One poller for chassis pings, two or more pollers for SNMP polls

[Tivoli Network Manager 4.2: Introduction and architecture](#)

© Copyright IBM Corporation 2017

Figure 2-41. Medium network deployment

As the size of the network grows, so do the hardware requirements for Tivoli Network Manager to retain peak efficiency. For example, to store a large amount of historical polling data, consider Tivoli Data Warehouse to store polling data. The NCIM network inventory remains in the default DB2 database. Moving polling data to Tivoli Data Warehouse reduces the load on the DB2 NCIM database. This data distribution enables network maps and root cause analysis to operate at peak performance. (Tivoli Network Manager 4.2 does not require Tivoli Data Warehouse for polling data summarization as it uses Apache Storm for this purpose. However, Tivoli Data Warehouse still might provide some performance and management benefits with large amounts of data.)

The following information describes two-server and three-server deployment:

- Two-server deployment:
 - Server 1: Network Manager core components, Tivoli Netcool/OMNibus, and the NCIM database. The core components are the network discovery, polling, root cause analysis, and event enrichment components.
 - Server 2: Dashboard Application Services Hub with associated Network Manager web applications.
- Three-server deployment:
 - Server 1: Network Manager core components.
 - Server 2: Tivoli Netcool/OMNibus.
 - Server 3: Dashboard Application Services Hub with associated Network Manager web applications, together with the NCIM database.

Large network description

- Large enterprise company with global network
- Network:
 - Contains Multiprotocol Label Switching (MPLS) core networks
 - 1000 – 12,000 devices with 30 or more ports per device
 - Network operations are done from central location with operations staff that monitor the core network
- Polling:
 - Chassis polling at 2 – 5 minute intervals
 - SNMP polling at 10 – 15 minutes
 - SNMPv3 polling of key devices
- Integration:
 - IBM Tivoli Monitoring with Tivoli Data Warehouse* running DB2 to facilitate performance reporting
 - IBM Tivoli Business Service Manager
 - IBM Tivoli Application Dependency Discovery Manager

***Note:** Tivoli Network Manager 4.2 does not require Tivoli Data Warehouse for report data aggregation. It uses Apache Storm for data aggregation.

Figure 2-42. Large network description

- Multiprotocol Label Switching (MPLS) is often used within large enterprise networks and telecommunications companies. Extra discovery time is necessary to gather MPLS information. The additional data from MPLS discovery is substantial because many devices have 30 or more ports.
- Deployments of Tivoli Network Manager in these environments often integrate with other IBM Tivoli products for the following functions:
 - Performance monitoring and reporting
 - Defining, monitoring, and reporting on the status of business services and service level agreements
 - Discovering application dependencies and component relationships

Large network deployment

For 1000 devices, a single server or dual server deployment can work

- For 12,000 devices, you need a distributed multiple server deployment

For the larger of these scenarios, consider the following configuration:

- Deploy two domains with two servers for each domain
- Dedicated database server with HADR
- Each server needs four processors, 8 GB RAM, redundant disk array
 - Server 1: Tivoli Network Manager
 - Server 2: Tivoli/Netcool OMNIbus and Dashboard Application Services Hub (DASH)
 - Server 3: Dedicated RDBMS supporting both domains
- Configure three pollers:
 - Master poller is used for SNMP real-time MIB graphing only
 - Chassis and interface ping poller operates in a non-master mode
 - SNMP poller operates in non-master mode

Tivoli Network Manager 4.2: Introduction and architecture

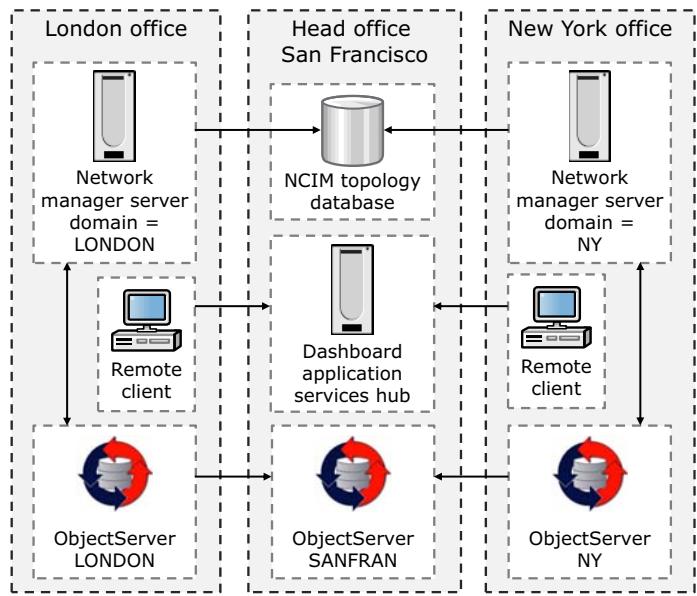
© Copyright IBM Corporation 2017

Figure 2-43. Large network deployment

The hardware requirements that are listed in this image are general guidelines. Servers can have more CPUs and memory.

Large network deployment example

- One ObjectServer and one Network Manager installation in both London and New York
- London and New York domains send events and topology to San Francisco
- One ObjectServer and one Dashboard Application Services Hub (DASH) installation in San Francisco for event consolidation
- Topology of either domain can be viewed, but discovery data is not consolidated



Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-44. Large network deployment example

Domains are discovered separately. The data from each domain can be written to the same database server. The data from each domain is stored separately. Data from each domain can be viewed from the GUI.

This example deployment consists of the following servers:

- One ObjectServer and one Network Manager installation in London. The London domain sends events and topology to San Francisco.
- One ObjectServer and one Network Manager installation in New York. The New York domain also sends events and topology to San Francisco.
- One ObjectServer and one Dashboard Application Services Hub installation in San Francisco.

Each server provides separate functions:

- The ObjectServer in San Francisco consolidates the events from London and New York.
- The Dashboard Application Services Hub server in San Francisco can access topology from both London and New York, but does not consolidate the topologies.
- Clients anywhere in the world can connect to the Dashboard Application Services Hub server, and view topology from London and New York.

Very large network description

- Large global enterprise company with 5 – 20 active GUI clients
- Network:
 - Over 12,000 network devices and key servers combined
 - Network infrastructure devices have approximately 30 interfaces per device
 - Other managed devices have two interfaces each
- Polling:
 - Chassis polling at 2 – 5 minute intervals
 - SNMP polling at 15 minutes or longer
 - SNMPv1 data collection
- Integration:
 - Tivoli Data Warehouse running DB2
 - Tivoli Business Service Manager
 - Tivoli Application Dependency Discovery Manager
 - Tivoli Netcool/OMNIbus

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-45. Very large network description

Very large network deployment

- Professional services are helpful in this type of deployment
- Break very large networks into multiple domains
 - For example, separate 15,000 devices into two domains
- Within each domain, configure servers for these functions:
 - Server 1: Network Manager with 36 GB of memory
 - Server 2: Tivoli Netcool/OMNibus and Dashboard Application Services Hub with up to 12 GB of memory
 - Server 3: A customer-selected RDBMS (DB2 or Oracle) with up to 84 GB of memory
- Three polling engines per domain:
 - Master real-time MIB graphing poller
 - Ping poller
 - SNMP poller

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-46. Very large network deployment

Setting up a high availability configuration would require more servers.

Single-domain ObjectServers

- Each domain has its own ObjectServer
 - Each domain has a collection ObjectServer that collects events from the probes in that domain
 - An aggregation ObjectServer gathers events from each of the collection ObjectServers
 - This approach scales well because new ObjectServers can be added as new domains are necessary; however, this approach requires multiple ObjectServers
- The Tivoli Network Manager IP Edition domains are independent
 - Can have one up and one down for maintenance
 - Scopes of the discovery can overlap
- The ObjectServer **ServerName** is used to identify from which domain an event originated

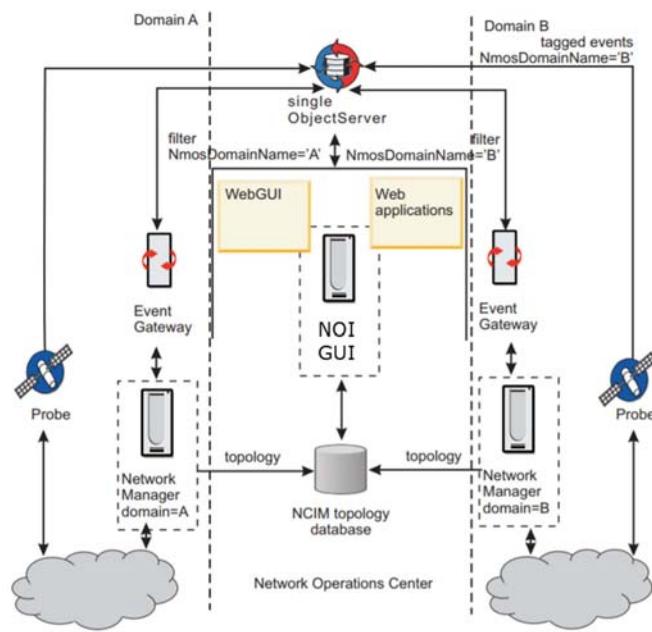
Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-47. Single-domain ObjectServers

Having a separate ObjectServer for each domain is a solution that scales well. This approach allows for more efficient event processing as your network grows. An aggregation layer consolidates the events into a single ObjectServer from which the operator views are derived.

Multiple-domain ObjectServers



© Copyright IBM Corporation 2017

Figure 2-48. Multiple-domain ObjectServers

In some cases, the event load does not necessitate multiple ObjectServers. Events are tagged with a domain tag to designate the domain from which they originated. When the operator right-clicks an event to view the topology, the domain tag specifies which set of data to show to the operator.

To implement this method, you need to do the following configuration activities:

- Configure the IBM Tivoli Netcool/OMNIbus probes to assign a **NmosDomainName** field to hold the domain name. For more information, see the *IBM Tivoli Network Manager IP Edition: Event Management Guide*.
- Add the **NmosDomainName** field to the Event Gateway filter for each domain. This field ensures that each gateway receives only events for its own domain.

NmosDomainName

- The ObjectServer schema for the **alerts.status** table includes the **NmosDomainName** field
 - For older versions of the ObjectServer that do not contain this field, the `$PRECISION_HOME/scripts/ncp_configure_omnibus.sql` script adds the **NmosDomainName** field to the ObjectServer
- Configure OMNIbus probes to tag domain-specific events with **NmosDomainName**
- Topoviz uses this tag to be certain that the correct device is shown from the correct domain
 - This tag is useful to create filtered views in an environment with multiple domains
- Add the **NmosDomainName** field to the Event Gateway filter for each domain
- For more information, see the *IBM Tivoli Network Manager IP Edition: Event Management Guide*

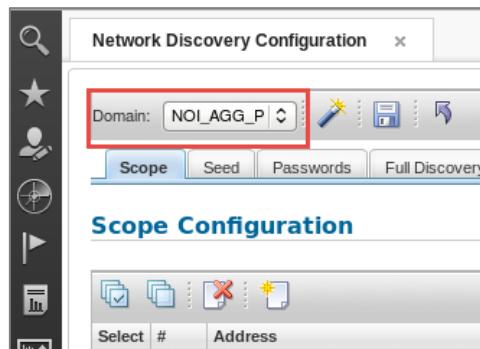
Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-49. NmosDomainName

Select the correct domain name

- To configure discovery of a domain or to view discovered devices, select the correct domain name from the GUI menu
- Verify that you have the correct domain name to prevent you from modifying the discovery or monitoring the configuration of the wrong domain



Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-50. Select the correct domain name

Operators must select the correct domain name of the topology that they want to view.

Topic summary

After you complete this section, you can do the following tasks:

- List the four components that must be present for a complete installation of Tivoli Network Manager
- List reasons for creating extra Tivoli Network Manager domains
- List protocols that are necessary to do a network discovery
- Describe how the ObjectServer differentiates between events from different Tivoli Network Manager domains

Figure 2-51. Topic summary

Review questions

1. What are the four main functions of Tivoli Network Manager?
2. At which of the three Tivoli Network Manager layers does the gateway operate?
3. Which of these two log file formats has the most useful information for troubleshooting?
 - a. *.log
 - b. *.trace
4. With multiple Tivoli Network Manager domains and a single ObjectServer, what ObjectServer field is used to designate the domain of origin for each event?

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-52. Review questions

Write your answers here:

Unit summary

- Explain the key functions of Tivoli Network Manager
- Diagram the key components of Tivoli Network Manager
- List implementation prerequisites
- Determine sizing guidelines for deployment of Tivoli Network Manager

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-53. Unit summary

Review answers

1. The four main functions of Tivoli Network Manager are:
 - Autodiscovery of a network
 - Monitoring and polling of network devices
 - Network fault isolation (root cause analysis)
 - Topology mapping
2. The Tivoli Network Manager gateway operates at the data layer.
3. Which of these two log file formats has the most useful information for troubleshooting?
 - a. *.log
 - b. *.trace
4. The NmosDomain field is used to designate the domain of origin for each event.

Tivoli Network Manager 4.2: Introduction and architecture

© Copyright IBM Corporation 2017

Figure 2-54. Review answers

Unit 3. Discovery basics

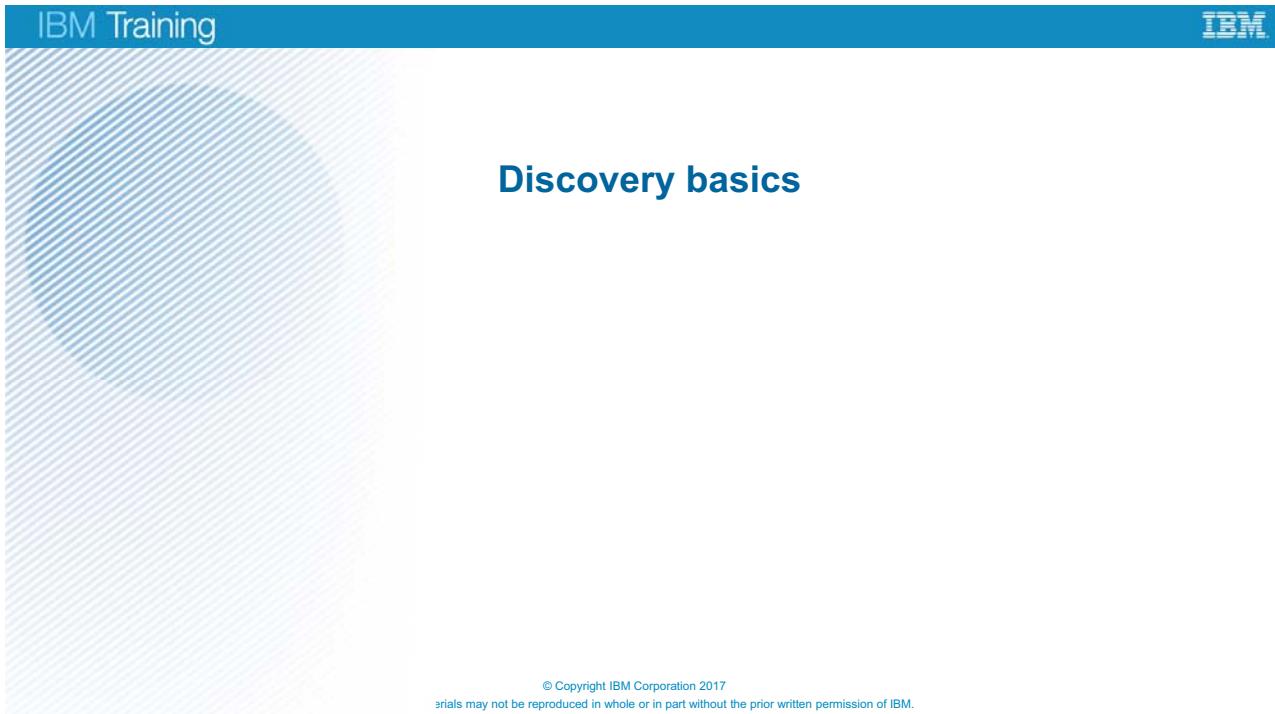


Figure 3-1. Discovery basics

Estimated time

01:30

Overview

In this unit, students learn to configure and run a network discovery by using Tivoli Network Manager.

How you will check your progress

- Complete checkpoint questions
- Student exercises

Unit objectives

- Configure a basic network discovery
- Describe the three fundamental tasks of discovery
- Schedule a regular full discovery of a network
- Use ping and file finders
- Run a network discovery
- Explain the function of the feedback mechanism during discovery

Figure 3-2. Unit objectives

3.1. How discovery works

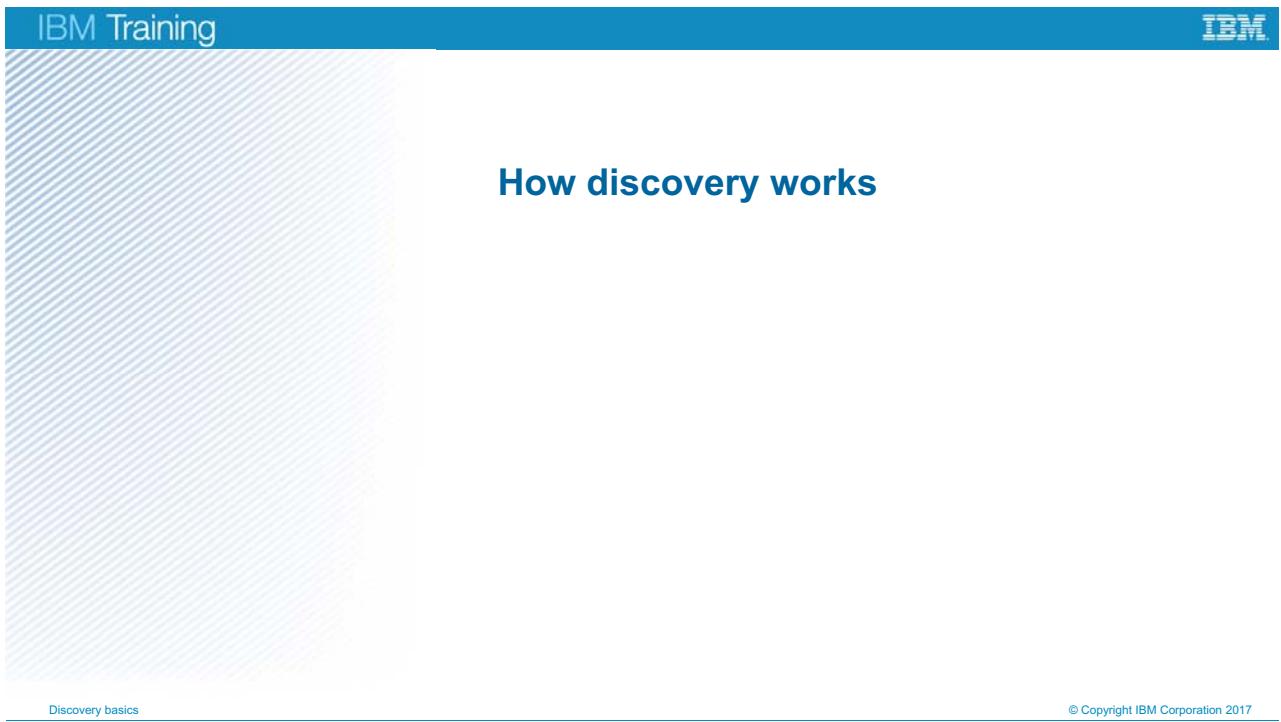


Figure 3-3. How discovery works

Discovery basics

After completing this topic, you should be able to do the following tasks:

- Configure a basic discovery of Layers 2 – 3 network devices
- List the protocols that the discovery process uses
- List the four major components of Tivoli Network Manager discovery
- Explain the three basic steps of discovery
- Identify which agent runs first

Figure 3-4. Discovery basics

Discovery is the key

- Monitoring and accurate network fault isolation (root cause analysis, or RCA) depend upon an accurate discovery
- Discovery also builds a detailed topology database that provides information to Tivoli Common Reporting

Figure 3-5. Discovery is the key

What you can discover

- Layer 3 devices
 - If they are IP
- Layer 2 devices
 - If they are Ethernet or other certified devices
- Frame Relay, asynchronous transfer mode (ATM) devices
 - If they have an IP management interface
- Devices (including Layer 1 devices) for which entity and connectivity information is available from these sources:
 - An element management system (EMS)
 - A comma-separated value (CSV) file
 - A supported database table

Figure 3-6. What you can discover

Discovery protocols

Enable the following protocols:

- Internet Control Message Protocol (ICMP)
- Simple Network Management Protocol (SNMP)
- Domain Name System (DNS)
- Address Resolution Protocol (ARP)
- Telnet or Secure Shell (SSH)

Figure 3-7. Discovery protocols

Tivoli Network Manager uses five key protocols. If you are discovering through firewalls, you must enable the following protocols to pass through the firewalls.

- Internet Control Message Protocol (ICMP)
- Simple Network Management Protocol (SNMP)
- Domain Name System (DNS)
- Address Resolution Protocol (ARP)
- Telnet or Secure Shell (SSH)

Some network devices might have access control lists (ACLs), which give access permission to individual devices or management subnets. The Tivoli Network Manager server must be included in the ACL to query the devices.

Discovery terminology and architecture

- **Finders** learn which devices exist
 - Populate database tables
 - Database schema in configuration file; can be queried
- **Agents** learn how the devices are connected
 - Populate database tables with entity and layer (connectivity) data
- **Stitchers** consolidate entity and connectivity information into a cohesive topology
 - Moves data from one discovery table to another
 - Builds entities first, then layers (connectivity)
- **Helpers** interrogate and poll network devices
 - Used by agents and polling process to do actual interrogation and polling of devices

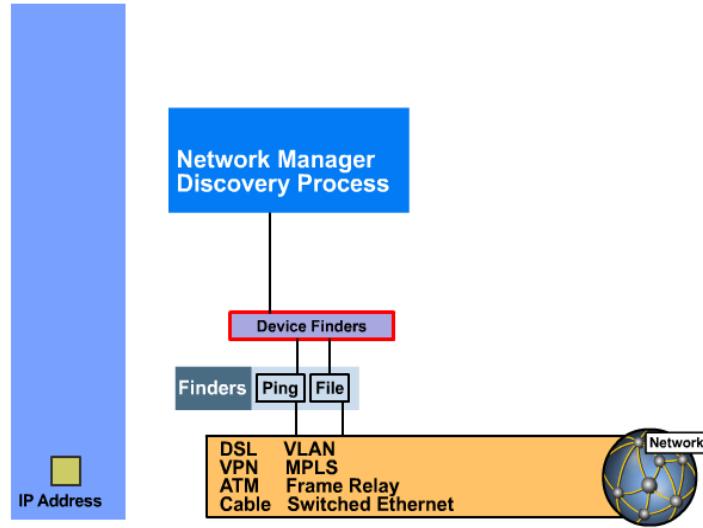
Figure 3-8. Discovery terminology and architecture

Tivoli Network Manager 4.2 uses four types of finders during the first phase of discovery:

- Ping finder
- File finder
- Collector finder (reads from EMS or CSV file)
- Database finder (reads device information from supported database)

IBM Training IBM

Step 1: Finders



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-9. Step 1: Finders

When discovery begins, the discovery is in **Phase 0** until it can load all of the configuration files and then find the first device. The ping finder finds a device when the device responds to a ping request. The file finder finds a device when it reads the IP address and host name in a seed file.

On the **Advanced** tab of the discovery configuration wizard, you can select an option to have the ping finder verify the existence of seed file entries.

- If this option is selected, the ping finder pings each entry in the hosts file that is supplied to the file finder. When a device responds to a ping, Tivoli Network Manager continues further processing.
- If this option is not selected, the device exists because it is in the seed file. It is instantiated in the network discovery even if the device cannot be interrogated.

Tivoli Network Manager also has two other ways of finding devices:

- A new discovery finder **ncp_df_dbentry** reads a database to retrieve a list of devices to find on the network.
- Discovery collector finders read a supported element manager or comma-separated value (CSV) file to find network entities. Collector agents gather detailed device information (if available) from the respective data repository.

For the purposes of giving a high-level overview of the discovery process, only the ping and file finders are explained in this section.

Finder options with either ping finder or file finder

Ping finder only

- Discovery ping sweeps subnets that are explicitly specified during discovery configuration
- If **feedback** is enabled, discovery ping sweeps other subnets that are attached to interfaces on discovered devices

File finder only

- Ping verification disabled:* Discovery finds those devices that are listed in the seed file if they exist
- Ping verification enabled:* Discovery finds only those devices that are listed in the seed file that respond to a verification ping
 - Service providers use this method to find only devices that they are paid to manage

Figure 3-10. Finder options with either ping finder or file finder

You can select the combination of ping and file finder options in Tivoli Network Manager.

Only the ping finder is enabled

In this situation, the Discovery ping sweeps specified subnets and any discovered subnets within the scope of discovery.

Only the file finder is enabled

In this case, if *ping verification is disabled*, discovery finds devices that are listed in the seed file whether they exist or not.

If ping verification is *enabled*, discovery finds only the devices that are listed in the seed file that respond to a verification ping. Managed service providers often use this method to find only those devices they are paid to manage.

When a discovery agent interrogates a devices routing table, it finds attached devices or subnets:

- If feedback is *enabled*, routing information passes to the ping finder. The ping finder then pings the device or ping sweeps the subnet if the device or subnet is included in the discovery scope. All devices that respond are then passed to the agents for interrogation.
- If feedback is *disabled*, information from device routing tables is ignored for the purposes of expanding discovery.

Tivoli Network Manager does not ping any device outside the configured scope.

Finder options with both ping finder and file finder

Ping and file finder

- **With feedback enabled:** Discovery begins with the seeded devices and subnets that are configured in the ping and file finders
 - Discovery reads interface information and then continues discovery by ping sweeping all in-scope subnets that are attached to a device
- **With feedback disabled:** Discovery finds all devices that are listed in specified seed files
 - The ping finder also pings all devices or subnets that are specified in the configuration
 - The ping finder does not ping sweep any subnets that are not explicitly specified in the discovery configuration

Feedback is the process whereby a list of subnets to which a device is attached is returned to the ping finder

- If those subnets are included in the discovery scope, the ping finder sweeps those subnets to find other devices in the network

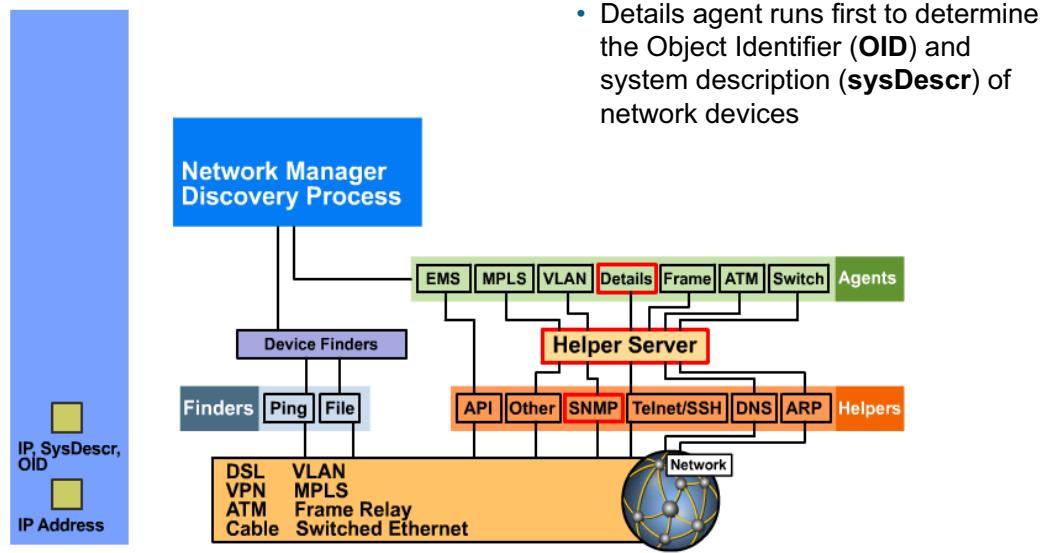
Figure 3-11. Finder options with both ping finder and file finder

Feedback is the process whereby a list of subnets to which a device is attached is returned to the ping finder. If those subnets are included in the discovery scope, the ping finder sweeps those subnets to find other devices in the network.

- If feedback is enabled, discovery begins with the seeded devices and subnets that are configured in the ping and file finders. Discovery reads interface information and then continues discovery by ping sweeping all in-scope subnets of which the device is a member.
- With feedback disabled, discovery finds all devices that are listed in specified seed files. The ping finder also pings all devices or subnets that are specified in the configuration. No other subnets are provided to the ping finder for processing.



Step 2a: Details agent



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-12. Step 2a: Details agent

The **Details** agent is always the first to run. It ascertains the system description and object ID of the device. This information is needed to determine which other agents must run to properly interrogate the device. You can have two Cisco routers of the same model with different versions of operating system. The difference in the operating system can require a different agent to interrogate each router.

No agent directly queries any devices. All agents send queries, whether SNMP, Telnet, SSH, DNS, or ARP, to the helper server, which queries the network devices.

Some agents require Telnet or SSH to interrogate older switches that do not report downstream connectivity or attached devices in response to an SNMP query. Telnet or SSH can also be necessary to gather Multiprotocol Label Switching (MPLS) configuration information.



Step 2b: Other agents

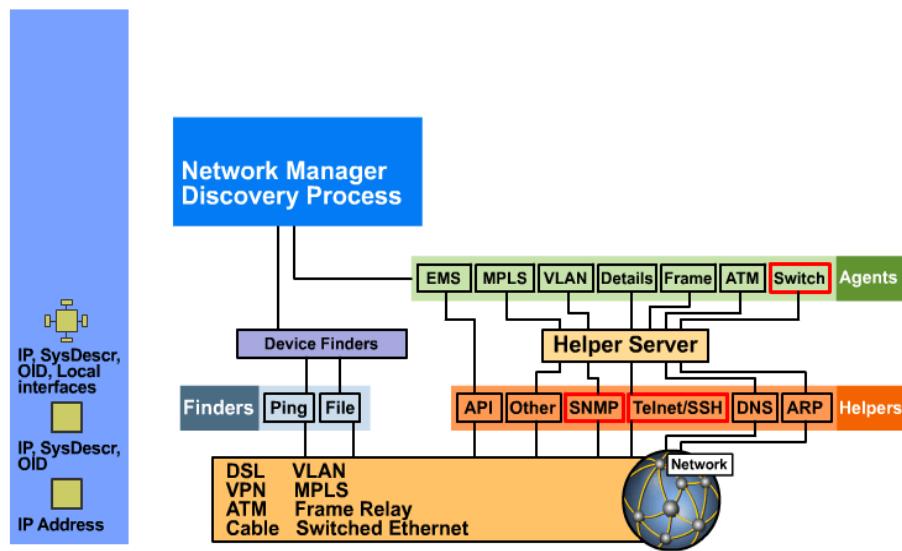


Figure 3-13. Step 2b: Other agents

© Copyright IBM Corporation 2017

Depending on the information found by the **Details** agent, other agents appropriate to the device are told to query the device.

The **AssocAddress** agent always runs immediately after the **Details** agent. This agent maintains a lookup table to determine whether Tivoli Network Manager already found the device through a different interface.

- If discovery found the device previously, no further action is taken. This feature prevents double-querying a device.
- If discovery did not interrogate the device previously, the device information is then sent to other agents for further interrogation.

Most agents have filters to limit the types of devices that they query. The filters are in the device filter section of the *.agt file in the \$ITNMHOME/disco/agents directory.

Although most agents use SNMP to query devices, some agents use other protocols. For the most efficient discovery, use only the agents necessary to get the level of discovery detail that you require. After you run a discovery, run the

`$ITNMHOME/scripts/perl/scripts/discoAgentsUsed.pl` script. That script shows you the agents that are used during discovery. You can then disable the unused agents to free up memory during subsequent discoveries.

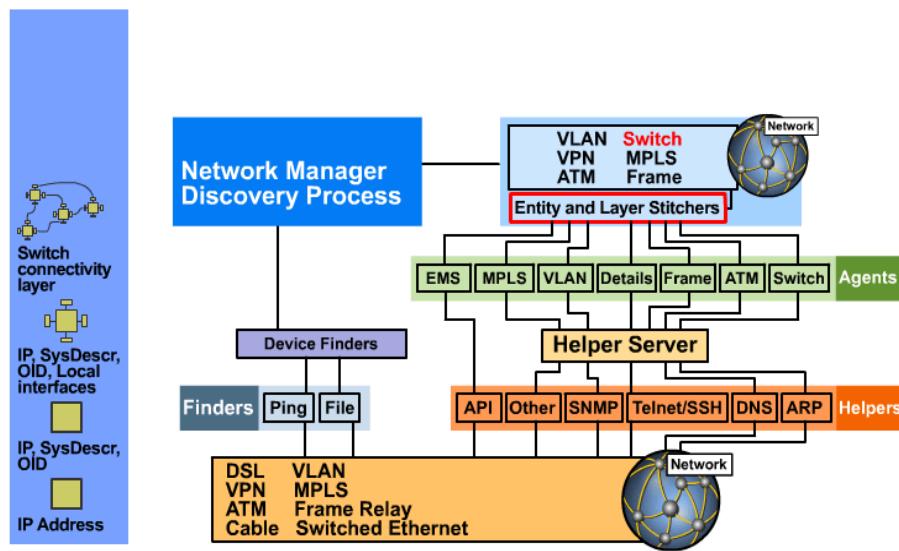


Important

Environment variables such as `$ITNMHOME` are set in the `/opt/IBM/tivoli/env.sh` file. (The `$ITNMHOME` variable is the equivalent of the older `$PRECISION_HOME` variable set to `$SNCHOME/precision`.) Each user who runs Tivoli Network Manager processes or scripts needs to first source the environment variables in that file. You can call this script in your user profile or source it manually with the following command:

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

Step 3: Stitchers (1 of 2)



Discovery basics

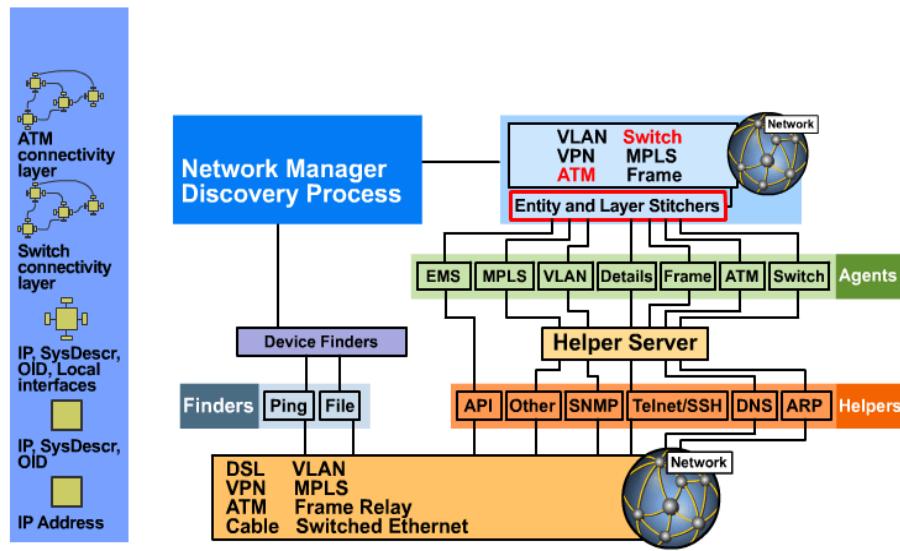
© Copyright IBM Corporation 2017

Figure 3-14. Step 3: Stitchers (1 of 2)

As the name implies, a stitcher sews information together. Stitchers create entity records for chassis and interfaces, and then layer records or connections between interfaces.

Every phase of the discovery process moves data by using stitchers. In this diagram, the switch stitchers build connectivity between interfaces on the switches.

Step 3: Stitchers (2 of 2)



Discovery basics

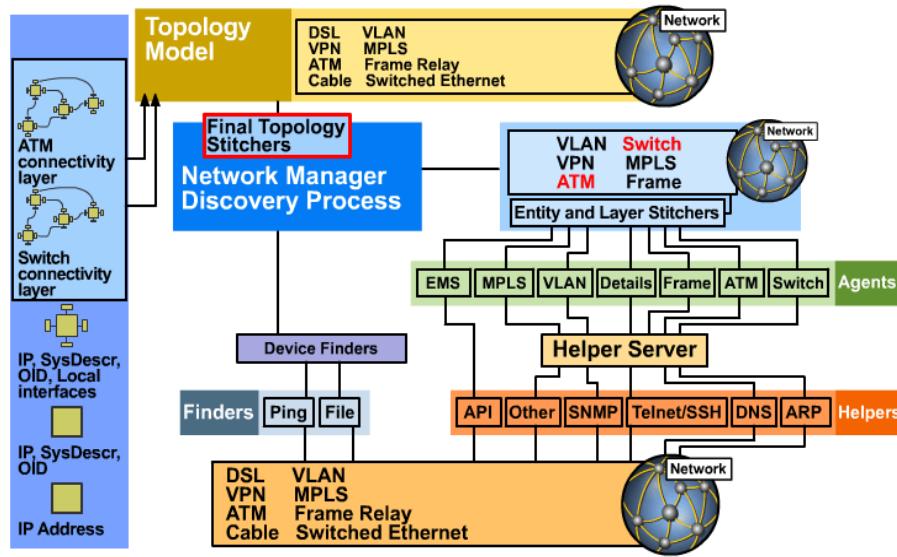
© Copyright IBM Corporation 2017

Figure 3-15. Step 3: Stitchers (2 of 2)

Entity stitchers and layer stitchers have the following characteristics:

- **Entity** stitchers build a list of chassis, interfaces, logical interfaces, and other entity types.
- **Layer** stitchers record the connectivity between devices. Each connectivity type, such as ATM or Frame Relay, is analyzed separately and then merged together.
- The entity and layer stitchers are run during the final phase of discovery.

Step 4: Stitchers send topology to Model



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-16. Step 4: Stitchers send topology to Model

CreateAndSendTopology.stch takes all the layers of connectivity and creates a fully stitched topology that contains all discovered devices and their connections.

Discovery Phases 0 and 1

Phase 0

- Initialization phase, ends when the first device is found

Phase 1

- The identification of all devices on the network by the finders
- Phase 1 finishes when the find rate drops beneath a certain level or nothing is found within a specific time period
 - The discovery service locks the **finders.processing** table after no new devices are found for 90 seconds (configurable)
 - This **m_NothingFindPeriod** stops finding new devices temporarily in order for discovery to complete more quickly
- Any devices that are found after this blackout period are sent to the **finders.pending** table
 - This state is called the **BlackoutState**
 - Agents complete their work and final-phase stitching is begun

Figure 3-17. Discovery Phases 0 and 1

Phase 0 and Phase 1 are described as follows:

- Phase 0 is the initialization phase. It ends when the finders discover the first device.
- During Phase 1, the finders identify all devices on the network. Phase 1 finishes when the find rate drops beneath a certain level or nothing is found within a specific time period. The discovery service locks the **finders.processing** table after no new devices are found for 90 seconds. You can configure this time period.
- The **m_NothingFindPeriod** stops finding new devices temporarily in order for discovery to complete more quickly. Any devices that are found after this blackout period are sent to the **finders.pending** table. The discovery is now in a **BlackoutState**. Agents complete their work, and final-phase stitching is begun.

The BlackoutState and Phase 1

- **Phase 1**

`m_BlackoutState=0`

- Device IP addresses are sent to **finders.processing**

`m_BlackoutState=1`

- Device IP addresses are sent to **finders.pending**

- All agents that can return data to the ping finder are run in Phase 1

- For example, the **IPRoutingTable** agent returns subnet ranges that must be passed to the ping finder

Figure 3-18. The BlackoutState and Phase 1

The **BlackoutState** for Phase 1 can be set to either 0 or 1.

- If the **BlackoutState** is set to **0**, devices are sent to **finders.processing**:

`m_BlackoutState= 0`

- If the **BlackoutState** is set to **1**, devices are sent to **finders.pending**:

`m_BlackoutState=1`

Some agents can return data to the ping finder if the feedback mechanism is enabled. These agents run in Phase 1. For example, the **IPRoutingTable** agent returns subnet ranges to the ping finder so that the finder can ping sweep the attached subnets.

Discovery Phases 2 – 4

- **Phase 2**

- Phase 2 agents resolve IP addresses to Media Access Control (MAC) addresses
- Phase 2 agents finish when each device in the dispatch table has an entry in the returns table

- **Phase 3**

- Switch agents download forward database tables and ping helper pings connected devices

- **Phase 4**

- Phase 4 agents do final-phase stitching
- After completion of Phase 4, discovery processes entries in **finders.pending** and adds devices to the topology database

Figure 3-19. Discovery Phases 2 – 4

Phases 2, 3, and 4 are described as follows:

- In Phase 2, agents do IP to Media Access Control (MAC) address resolution. Phase 2 agents complete when each device in the dispatch table has an entry in the returns table.
- In Phase 3, switch agents download forward database tables and the ping helper pings connected devices.
- In Phase 4, agents do final stitching. These final stitchers build entities and layers and combine into a topology. After completion of Phase 4, discovery processes any entries in **finders.pending** and adds those devices to the topology database.

Topic summary

Having completed this topic, you should be able to do the following tasks:

- Configure a basic discovery of Layer 2 – 3 network devices
- List the protocols that the discovery process uses
- List the four major components of Tivoli Network Manager discovery
- Explain the three basic steps of discovery
- Identify which agent runs first

Figure 3-20. Topic summary

3.2. The discovery wizard



Figure 3-21. The discovery wizard

Most Tivoli Network Manager administrators use the discovery wizard on the first discovery only. After that, they configure discovery by selecting the tab that corresponds to the configuration task and make changes there. However, it is instructive to go through the process with the discovery wizard one time to learn the logical sequence of configuration steps to take when you configure a discovery.

Using the discovery wizard

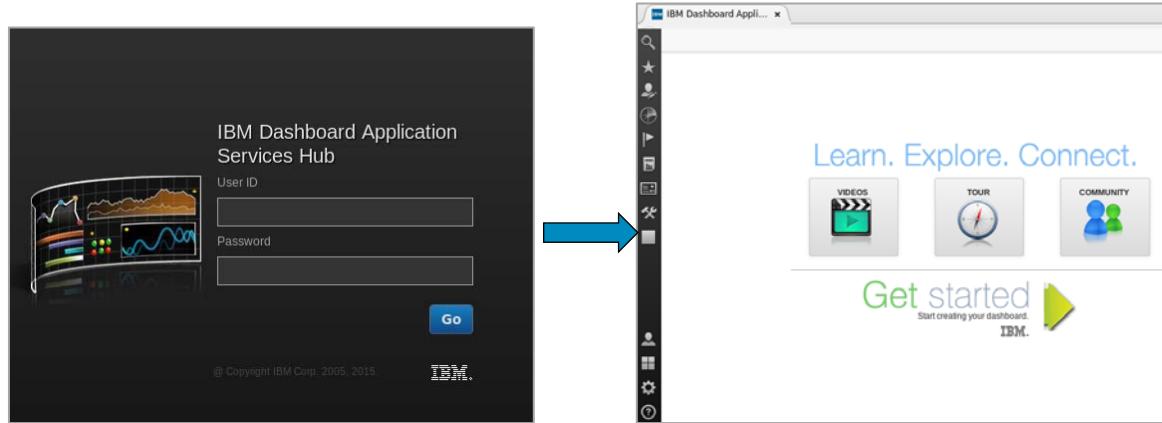
After completing this topic, you should be able to do the following tasks:

- Configure a discovery with the discovery wizard
- Know the essential steps to prepare for a discovery
- Know the importance of setting a discovery scope
- Explain VLAN modeling

Figure 3-22. Using the discovery wizard

Log on to IBM Dashboard Application Services Hub

- The IBM Dashboard Application Services Hub (DASH) replaces Tivoli Integrated Portal and eliminates the use of Java runtime engines (JRE) in Tivoli Network Manager

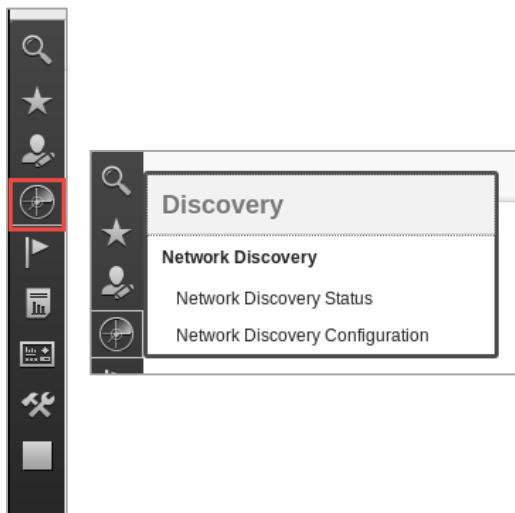


Discovery basics

© Copyright IBM Corporation 2017

Figure 3-23. Log on to IBM Dashboard Application Services Hub

Configuring discovery



- The Tivoli Network Manager server provides these methods for configuring discovery:
 - Configure the discovery with the Discovery Configuration Wizard
 - Configure the discovery with the Discovery Configuration tabs
 - Configure the discovery by manually editing the component configuration files and inserting values into the various component databases
- Select the radar icon and then **Network Discovery Configuration** DASH menu

Discovery basics

© Copyright IBM Corporation 2017

Figure 3-24. Configuring discovery

Normally, users access the wizard for the initial discovery configuration only. Editing configuration files manually is an advanced technique that is used only occasionally by technical support. Most often, you go to the Discovery Configuration GUI and configure only those tabs necessary to implement your discovery properly.

Preparing for discovery

- Determine the scope of discovery
 - Is the Network Manager server positioned to be able to interrogate all the devices that you want included in discovery?
 - Determine whether parts of the network need to be excluded from discovery
- Verify that the server can access network devices
 - Gather relevant SNMP community strings and Telnet/SSH passwords
 - Configure routes on the Tivoli Network Manager server and confirm that the server can reach devices on each subnet included in the discovery
 - Configure access control lists (ACLs) or position the Tivoli Network Manager server on a management subnet so that the server can query network devices
- Discovery must include all devices on which you want network fault isolation

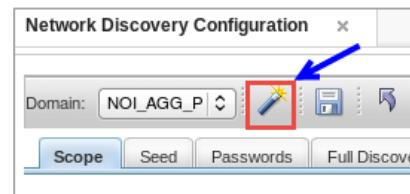
Figure 3-25. Preparing for discovery

Before you configure and run a discovery, verify the following items:

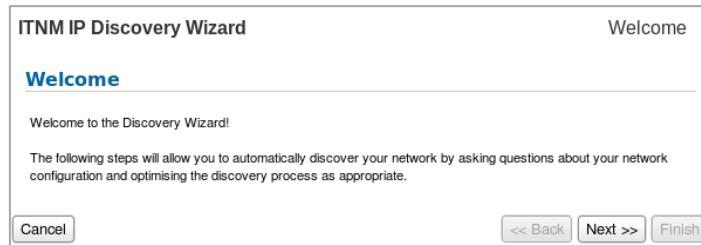
- **Scope of discovery:** Is the Network Manager server positioned to interrogate all devices that need to be included in your topology? Consider all necessary networks and subnetworks. Determine the associated netmasks. Determine whether any parts of the network need to be excluded. Gather all relevant SNMP community strings for the devices within the scope.
- **Routing:** Ensure that each of the networks and subnetworks to be discovered can be reached from the Tivoli Network Manager IP Edition server with an ICMP ping. If necessary, add routes to the Tivoli Network Manager IP Edition server with the `route add` command.
- **Access control lists (ACL):** Tivoli Network Manager IP Edition uses five protocols that might need to pass through firewalls. These protocols are ICMP, SNMP, DNS, ARP, and Telnet. Coordinate your efforts with the firewall administrator to remove obstacles to discovery of your network.
- **Root cause analysis (RCA):** Root cause analysis can run only on devices within the topology. Discovery must include all devices that are actively polled or send alerts to the ObjectServer. These include systems that run either the Network Manager poller or Tivoli Netcool/OMNibus probes. For more information about root cause analysis, see the *IBM Tivoli Network Manager IP Edition Monitoring and RCA Guide*.

Starting the discovery configuration wizard

- From the **Icon** menu, select the **wizard** option



- Click **Next** when you see the welcome message



[Discovery basics](#)

© Copyright IBM Corporation 2017

Figure 3-26. Starting the discovery configuration wizard

The Discovery Configuration GUI wizard helps in the rapid configuration and deployment of Tivoli Network Manager IP Edition within a proof of concept (POC) environment. It is useful to establish the basic discovery configuration on which subsequent extended discoveries are based.

The GUI creates and populates the appropriate domain-specific discovery configuration files found within `$NCHOME/etc/precision`. To configure discovery, the Tivoli Network Manager IP Edition **CONFIG** component must be running. The **ncp_ctrl** process starts this component.

Changes made to discovery files during discovery configuration automatically cause domain-specific files to be saved. The file names contain the domain name such as `DiscoAgents.ITNM4GO.cfg`.

Always specify Tivoli Network Manager IP Edition domain names in all uppercase letters. This practice makes it easier to recognize which files belong to which domain.



Note

Do not use a text editor (such as **vi** or **gedit**) to edit the underlying disco configuration files in `$NCHOME/etc/precision` while also using the discovery configuration GUI. The GUI is not aware of changes that are made to those files unless it makes the changes. To make the GUI aware of changes that are made through manual editing, force it to read the configuration files again by restarting the discovery process.

Choosing a scoped or unscoped discovery

- Avoid selecting an unscoped discovery
- If invalid static route entries exist on network devices, it can cause problems with default gateways

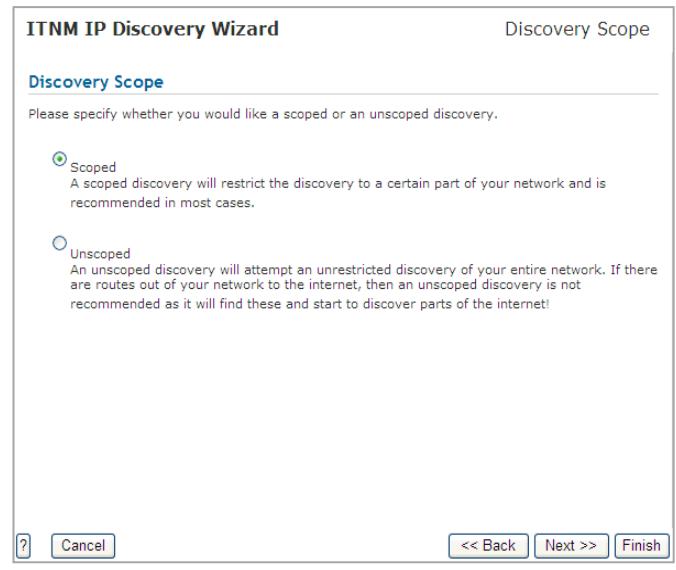


Figure 3-27. Choosing a scoped or unscoped discovery

The safest way to run a discovery is to ensure that the discovery engine interrogates and discovers only particular areas of your network.

Selecting **Scoped** forces the Tivoli Network Manager IP Edition discovery to find and interrogate only those devices within the specified scope of addresses and subnets. Always use a scoped discovery.

Selecting **Unscoped** configures the discovery engine to keep discovering while onward routes are discovered and accessible. Unscoped discoveries are sometimes used when a customer does not even know what subnets are in their network. Such discoveries have potential problems.

- If firewalls or devices on the network of the ISP are not configured properly, the discovery can extend and discover devices outside the intended network.
- Network devices might contain static routes that are incorrect. Attempts to discover these unavailable devices and subnets might cause traffic to go to the default gateway on the network. Excessive numbers of these queries can cause a default gateway to have severe performance problems. It can interfere with normal gateway traffic.

Scope limits the extent of discovery

- No addresses outside the scope are pinged or queried with SNMP

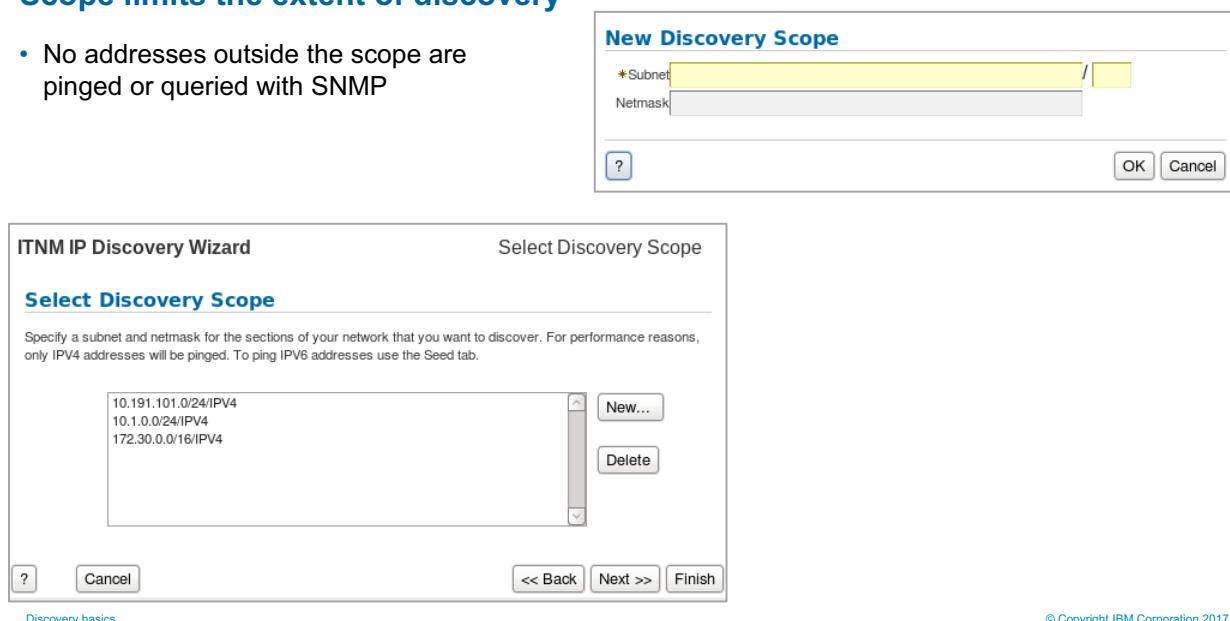


Figure 3-28. Scope limits the extent of discovery

After you select **Scoped** from the previous screen, enter the network addresses and masks for those areas to be included within your discovery. All subnets and addresses not within the scope are ignored. You can see the results within

`$NCHOME/etc/precision/DiscoScope.domainName.cfg`:

```
Insert into scope.zones
(
  m_Protocol,
  m_Action,
  m_Zones
)
values
(
  1,
  1,
  [ {m_Subnet='192.168.1.0',m_NetMask=24} ]
);
```

By default, the ping finder is configured to use scopes. It checks all device addresses sent to it from the discovery agents. The default behavior prevents pinging any addresses that are not included in the scope. No ICMP, SNMP, Telnet, SSH, ARP, RARP, or other discovery protocol packets are sent to devices that are not included in the scope.

For performance reasons, only IPV4 addresses are pinged. To ping IPV6 addresses, use the Seed tab in the Discovery Configuration GUI.

Discovery filters and exclusions

Discovery configuration begins by determining the scope of discovery

- If network devices contain invalid static routes in the network, an unscoped discovery can cause heavy traffic on the default gateway
 - Use an unscoped discovery with caution
- The scope must include the Tivoli Network Manager server
- The scope limits the breadth of discovery; nothing outside of the scope is pinged or interrogated

You can create exclusions to discovery

- Exclusions override inclusions
 - If you specify a Class B network for discovery but exclude one subnet within that address range, that subnet is not discovered
- Filtering excludes devices that fail to match the filter criteria

Figure 3-29. Discovery filters and exclusions

You can create an exclusion zone that is a subset of an inclusion zone. You can create an exclusion that is a subset of the inclusion. The ping finder does not ping excluded addresses. Discovery agents do not interrogate excluded devices. An exclusion always overrides an inclusion.

Prediscovery filters prevent interrogation of devices. You can configure a filter that is based on a field in the **Details.returns** table to eliminate some devices from further interrogation by other agents. A typical prediscovery filter, for example, can eliminate workstations and printers from the discovery.

Sometimes you need to eliminate interfaces from a device in your discovery. Its principal use is to exclude an interface from a device. For example, you might want to eliminate a **dsc0** interface from a Cisco 5500 or a **fpx0** foreign exchange interface from a Juniper box. Some boxes have a **discard** interface. These types of interfaces sometimes represent themselves as having virtual connections that do not represent physical connectivity. An **interface filter** eliminates unwanted interfaces from discovery. Interface filters are discussed later in this course.



Information

Tivoli Network Manager 4.2 still has a post-discovery filter that can eliminate unwanted interfaces. However, this filter is disabled by default as its use is mutually exclusive with the DNCIM database. Moreover, the post-discovery filter eliminates devices only after all agents and most of the stitchers complete their work. Agent-level interface filtering deprecates the usefulness of the post-discovery filter. Interface filters can act either on all agents or on specific agents. They eliminate certain interfaces from the discovery. Because these filters act at the agent level, no stitchers ever process connectivity information. For this reason, interface filters are more efficient than a post-discovery filter.

Out-of-scope interfaces

- Your network can contain devices that are within the discovery scope but can contain interfaces that are out of scope
- By default, the Layer 3 discovery agents do the following tasks:
 - Download the device interface table
 - Discover all the device interfaces
 - Instantiate all interfaces
- For out-of-scope interfaces, the **IsActive** flag is set to a value of 2
 - This setting prevents ping monitoring of these interfaces without a change of the scope for the ping monitor

Figure 3-30. Out-of-scope interfaces

Your network might contain devices that are within the discovery scope, but the devices themselves contain interfaces that are out of scope.

You can also modify the **scope.instantiateFilter** so that the out-of-scope interfaces are not instantiated in the discovery.

SNMP community strings

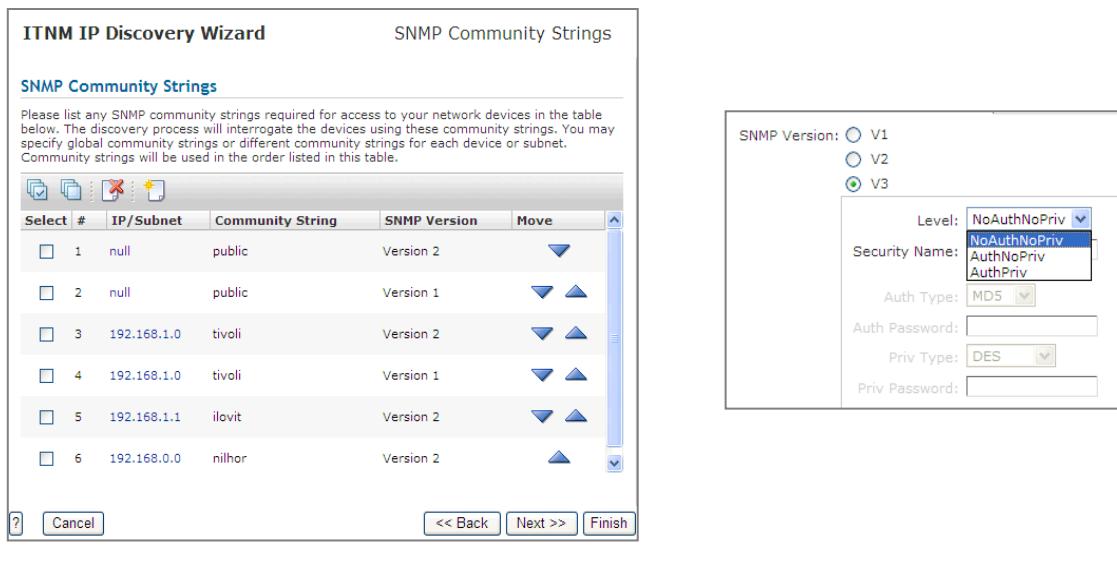


Figure 3-31. SNMP community strings

© Copyright IBM Corporation 2017

To interrogate devices within the network, it is important to provide the appropriate Simple Network Management Protocol (SNMP) community string access information. Tivoli Network Manager IP Edition requires the read-only SNMP community string because the product does not affect the device configuration. SNMP versions 1, 2, and 3 are supported. By default, a global community string of **public** is defined.

Entries can be specified in the following ways:

- Per device
- Per subnet
- Globally

Tivoli Network Manager processes community strings by looking for the most specific match first. For community strings at an equivalent level, they are processed in the order in which they are entered.

Where possible, specify the correct SNMP version. This practice enhances performance. If you are unsure which SNMP version a device uses, create two entries. Always specify the version 2 community string first so that it is processed before the version 1 community string. This practice ensures that Tivoli Network Manager IP Edition can use the **GetBulk** command for increased performance of the querying process.

In this example, when Tivoli Network Manager sends SNMP queries to the device at 192.168.1.1, it attempts the community strings in the following order:

1. ilovit (version 2)
2. tivoli (version 2)
3. tivoli (version 1)

4. nilhor (version 2)
5. public (version 2)
6. public (version 1)

Always enter each community string twice. Specify the version 2 string before the version 1 string. Tivoli Network Manager attempts the version 2 SNMP GET first. If the device responds, then the discovery helper uses this more efficient polling to reduce the chatter of version 1 polling. If you put the version 1 string first in the list, then it uses version 1 polling, which results in reduced performance.

Tivoli Network Manager also supports SNMP version 3.



Reminder

When querying new devices, Tivoli Network Manager first applies the most specific match from the list of configured SNMP community strings.

- At each level of subnet masking, Tivoli Network Manager processes from top to bottom.
 - When rediscovering nodes that exist in the topology, Tivoli Network Manager uses the SNMP community string that succeeded when it first found the device.
-

Processing SNMP community strings

- Tivoli Network Manager attempts the most specific match first
- If that fails, it looks for other matches at the same level of specificity and attempts them in the order that they are entered in the discovery configuration GUI
- If those strings fail, it looks for matches at the next lower level of specificity and processes them in the order that they are entered in the GUI
- It continues this process until it finds a match
 - Global community strings are the last to be attempted
- See the example in your *Student Guide*

Figure 3-32. Processing SNMP community strings

Tivoli Network Manager IP Edition processes SNMP community strings in the following order, which also applies to Telnet and SSH passwords:

1. The most specific match
2. The next closest match until all applicable matches are exhausted
3. The global community strings in the order in which they are listed in the SNMP community string table in the GUI

SNMP password properties

The dialog box is titled "SNMP Password Properties". It contains the following fields:

- Community String:** A required field.
- Apply To:** Set to "All Devices".
- IP Address:** Input fields for four octets.
- Subnet:** Input fields for four octets followed by a slash and another input field.
- Netmask:** Input fields for four octets.
- SNMP Version:** Set to "V1". Other options are "V2" and "V3".
- Security Settings:**
 - Level:** NoAuthNoPriv.
 - Security Name:** Input field.
 - Auth Type:** MD5.
 - Auth Password:** Input field.
 - Priv Password:** Input field.
- Connection Parameters:**
 - SNMP Port:** 161.
 - Timeout:** 0 ms.
 - Retries:** 0.

Discovery basics

© Copyright IBM Corporation 2017

Figure 3-33. SNMP password properties

You can add new SNMP community strings. If possible, avoid specifying all strings as global strings. Adding more specificity increases the speed of discovery and causes fewer SNMP authentication failures during the initial discovery process. Subsequent discoveries use cached SNMP authentication information, reducing authentication failures until devices on the network undergo a change of community strings.

SNMP v3 configuration shows you three security options for device access:

- Supplying only a group name
- Supplying a group name and *authentication* password
- Supplying a group name, authentication password, and *privileged* password

Telnet access configuration

Tivoli Network Manager supports SSH versions 1, 1.5, and 2

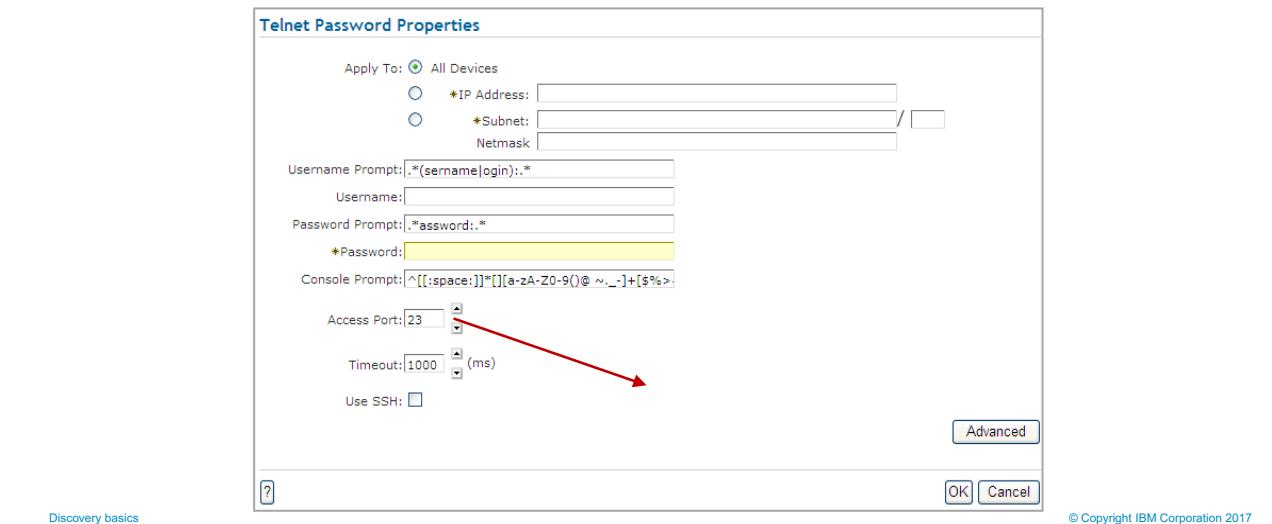


Figure 3-34. Telnet access configuration

Telnet access is often necessary to get configuration for Multiprotocol Label Switching (MPLS). It is also needed to garner information from some older devices that do not reveal forward connectivity information in response to an SNMP query. Older Ethernet switches might require Telnet access to get downstream Media Access Control (MAC) addresses from content-addressable memory (CAM).

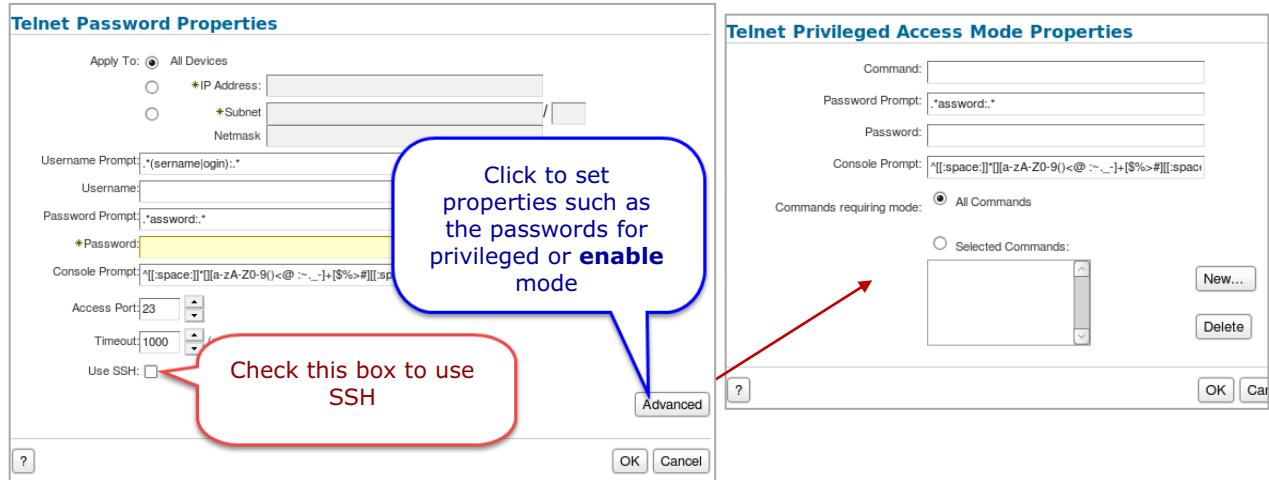
The Telnet Discovery Helper uses two device access and password configuration files to gain access to devices on the network:

- `TelnetStackSchema.cfg` defines the `telnetStack` database. You do not need to alter this file, as it defines the schema only.
- `TelnetStackPasswords.domainName.cfg` contains any necessary inserts into the `telnetStack` database. All the insertions that are given in this section are contained within this file in encrypted form.

By editing the `TelnetStackPasswords.domainName.cfg` file, you can manually specify OQL inserts to configure access parameters for every IP or subnet address.

It is also possible to specify an SSH connection when you configure Telnet device access. Versions 1.5 and 2 of SSH are supported. Tivoli Network Manager IP Edition currently supports password-based authentication or no authentication for SSH. It does not support Rivest-Shamir-Adleman (RSA) signature authentication.

Telnet passwords



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-35. Telnet passwords

From the Telnet Password Properties window, you can configure the following fields:

- The appropriate user name
- Password
- Scope of device addresses to which this access configuration is applied

You can set Telnet privileged access mode properties with the **Advanced** option. In the **Command** field, specify the command to enter privileged access mode. This mode is called **Enable**.

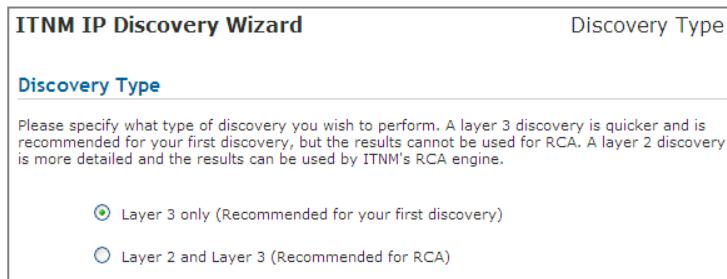
IBM Tivoli Network Manager IP Edition uses several commands that require the **Enable** mode for Cisco devices.

```
show run
show mac-address-table
show ip nat translation
```

None of these commands change the configuration of the device. However, these commands work only in the **Enable** (or **privileged**) mode.

Discovery agents

- **Layer 3 only** discoveries complete quickly
 - The results of this discovery indicate whether configuration problems exist with the network discovery
- Normal discoveries use the **Layer 2 and Layer 3** option



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-36. Discovery agents

The discovery wizard here provides a choice between a **Layer 3 only** discovery and a **Layer 2 and Layer 3** discovery. The wizard preselects certain agents based on this choice. These agents have their **m_Valid** field set to **1** in the `$NCHOME/etc/precision/DiscoAgents.domainName.cfg` file.



Note

It is usually a good idea for the first discovery in a large network to be a **Layer 3 only** discovery. Discovery completes quickly. Doing a Layer 3 discovery first confirms whether you have problems that hinder a full Layer 2 and Layer 3 discovery. Typical problems include the following items:

- Access control lists (ACLs) block access to devices
- Firewalls block access to devices
- Wrong SNMP community strings prevent device interrogation by discovery agents

After you eliminate any detected problems here, you can do a more detailed discovery by choosing the applicable Layer 2 agents.

If you select a Layer 2 discovery in the discovery wizard, you are asked if you want to model virtual local area networks (VLANs). Selecting this option affects the Network Manager downstream suppression. If you select a Layer 3 discovery only, you do not see this option.

VLAN modeling

- The discovery wizard asks this question only if you selected a Layer 2 discovery on the preceding screen
- It determines whether VLANs are used in determining the root cause of a network fault



Figure 3-37. VLAN modeling

In a typical discovery, Layer 2 switches might contain virtual local area network (VLAN) information. If you want the Tivoli Network Manager root cause analysis (RCA) to disregard downstream VLANs as part of its calculations, then set this parameter to **No**. You typically make this choice in networks where the following conditions are true:

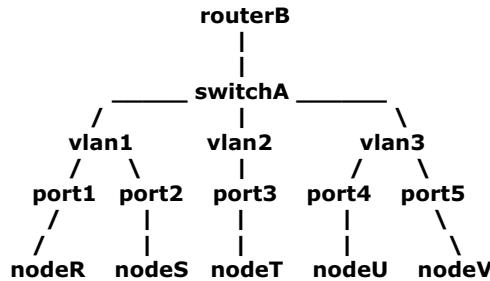
- The networks are highly meshed.
- You want to speed up discovery (the **DISCO** service started by the **ncp_disco** binary) or RCA calculations.

If you select **Yes** here, Tivoli Network Manager models the VLANs and uses that information in its RCA calculations. The RCA calculations understand the following conditions:

- Data can travel between two devices on the same VLAN.
- Data cannot travel between two devices on separate VLANs without first going through a router.

VLAN considerations

- When VLANs are modeled, the Tivoli Network Manager RCA plug-in evaluates VLAN configuration as part of root cause analysis
 - It understands that two nodes on separate VLANs cannot communicate without sending packets through a router
 - If VLAN modeling is disabled, the RCA plug-in evaluates only physical connectivity
- Nodes R and S can communicate with each other but cannot communicate with T, U, or V without going through a router



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-38. VLAN considerations

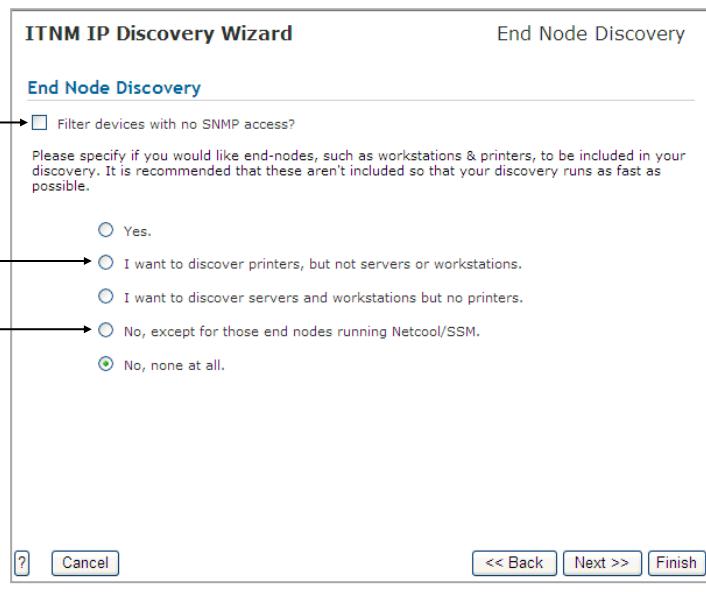
In this example, RCA understands that traffic can go from **nodeR** to **nodeS** because both devices are on the same VLAN. But traffic cannot go from **nodeR** to **nodeT**, **nodeU**, or **nodeV** without going through **routerB**. With multiple VLANs in a switched Ethernet environment, the task of root cause analysis with VLAN modeling becomes complex.

Without VLAN modeling, the RCA engine assumes that packets are able to travel across any available physical connections.

Although modeling VLANs can give you more accurate RCA results, sometimes you might prefer to not model VLANs. In a heavily meshed network with plentiful routing modules, VLANs often employ redundant routes. In this case, you might want to disable VLAN modeling to improve the speed of discovery and network fault isolation.

Prediscovery filters

- You can eliminate devices to which you have no SNMP access
- Tivoli Network Manager comes with predefined object identifier (OID) filters to help it easily eliminate printers and workstations from your discovery
- You can have Tivoli Network Manager discover only those nodes that are running a particular agent that it can interrogate



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-39. Prediscovery filters

A prediscovery filter restricts device discovery by allowing only those devices that pass the filter to be instantiated into the topology. If no prediscovery filter is defined, all devices within the scope are discovered. The prediscovery filter uses the columns of the **Details.returns** database table.

The wizard makes available basic choices for a prediscovery filter. You can make the following typical selections:

- Eliminate printers, but keep other end nodes such as workstations or printers
- Discover printers, but not servers and workstations
- Discover no end nodes unless they are running a Tivoli Netcool SSM agent

To eliminate a specific device from discovery, you can select one of two options:

- Exclude the device from the scope of discovery
- Create a discovery

If the device does not grant SNMP access to the Details agent, the Details agent might not be able to retrieve Management Information Base (MIB) data. However, Tivoli Network Manager determines at least the IP address of each detected device. Therefore, **m_UneAddress** is an ideal field on which to base a prediscovery filter.

You can define multiple prediscovery filters. Filters are combined automatically with a Boolean AND expression. All criteria that are defined in all filters must be matched for a device to get through the filter and pass on to agent interrogation. You can examine the filters in `$NCHOME/etc/precision/DiscoSchema.domainName.cfg` as entries into the **scope.detectionFilter** table.

Discovery speed versus accuracy

- Selected options determine number of agents used
- After discovery is completed, go to the **Full Discovery Agents** tab and select all Layer 2 and Layer 3 agents for a complete discovery

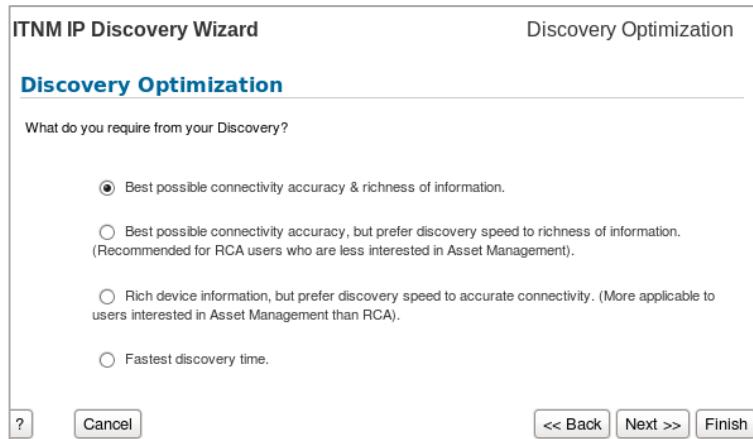


Figure 3-40. Discovery speed versus accuracy

Choices on this discovery wizard screen affect several areas. Choosing **Best possible connectivity accuracy and richness of information** causes Tivoli Network Manager IP Edition to do the following actions:

- Increase the SNMP timeout value from 3000 to 4000
- Increase the number of ping retries from 0 to 1

Choosing **Fastest discovery time** causes the following results:

- Fewer agents to be run, resulting in less discovery detail
- SNMP timeout values and ping retry values to be decreased

Options between these two extremes change the same variables to a corresponding degree. The following configuration files are typically affected:

- `DiscoSnmpHelperSchema.domainName.cfg`
- `DiscoPingFinderSchema.domainName.cfg`
- `DiscoAgents.domainName.cfg`

Network reliability

Selections determine SNMP timeout and retry values



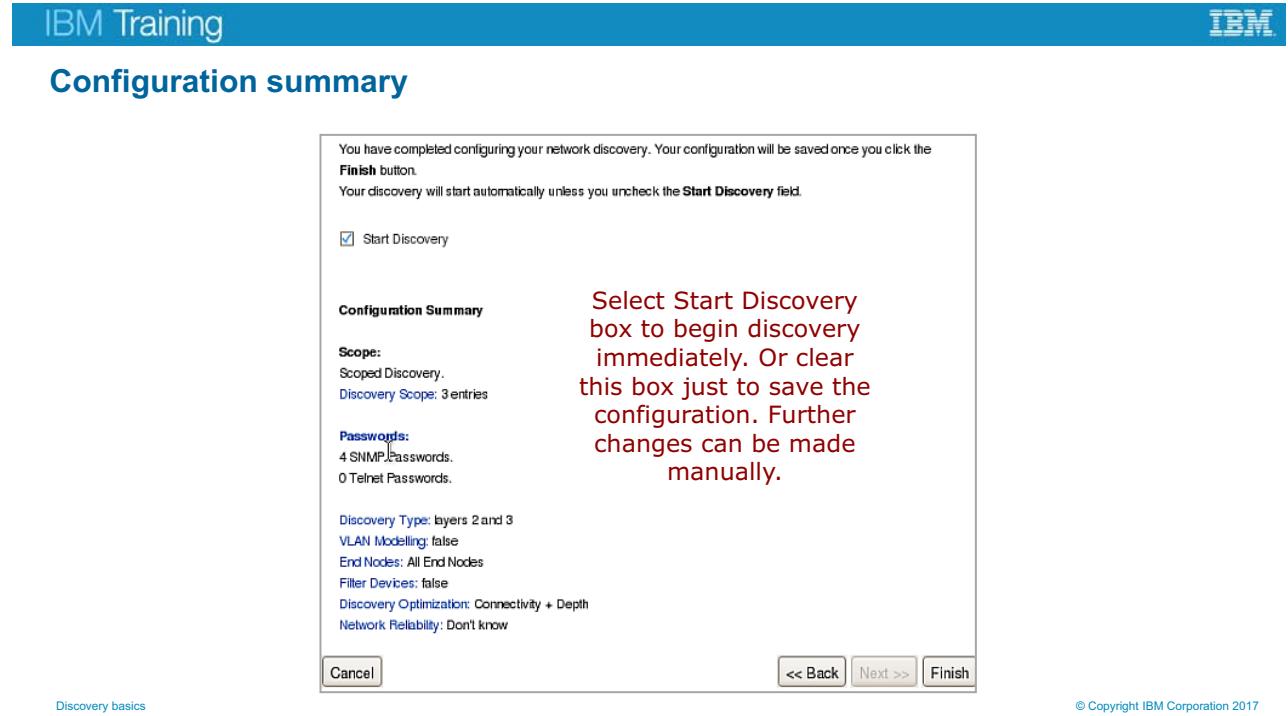
Discovery basics

© Copyright IBM Corporation 2017

Figure 3-41. Network reliability

Use the following descriptions to determine the best SNMP timeout and retry values for your environment:

- **Very reliable:** This option directs the wizard to apply short timeouts, without any retries. Appropriate for a reliable network, this selection results in fast discoveries. If you requested fastest possible discovery on the previous screen, timeouts set by this option are even shorter.
- **Quite reliable:** This option directs the wizard to apply slightly longer timeouts, with a single retry for both SNMP and ping requests.
- **Not very reliable:** This option directs the wizard to apply longer timeouts, two retries for SNMP requests, and one retry for ping requests. These longer times are suitable for less reliable networks.



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-42. Configuration summary

When you click **Finish**, allow the window to close on its own. It is not necessary to close the window manually. If you do not see the **Start Discovery** option on the screen, you have insufficient rights. Return to the security configuration to verify that you have the appropriate roles and groups assigned.

Topic summary

Having completed this topic, you should be able to following tasks:

- Configure a discovery with the discovery wizard
- List the essential steps to prepare for a discovery
- Explain the importance of setting a discovery scope
- Explain VLAN modeling

Figure 3-43. Topic summary

3.3. Starting discovery

Four methods exist for running a discovery: a manual full discovery, a scheduled full discovery, a manual partial discovery, and a triggered discovery.

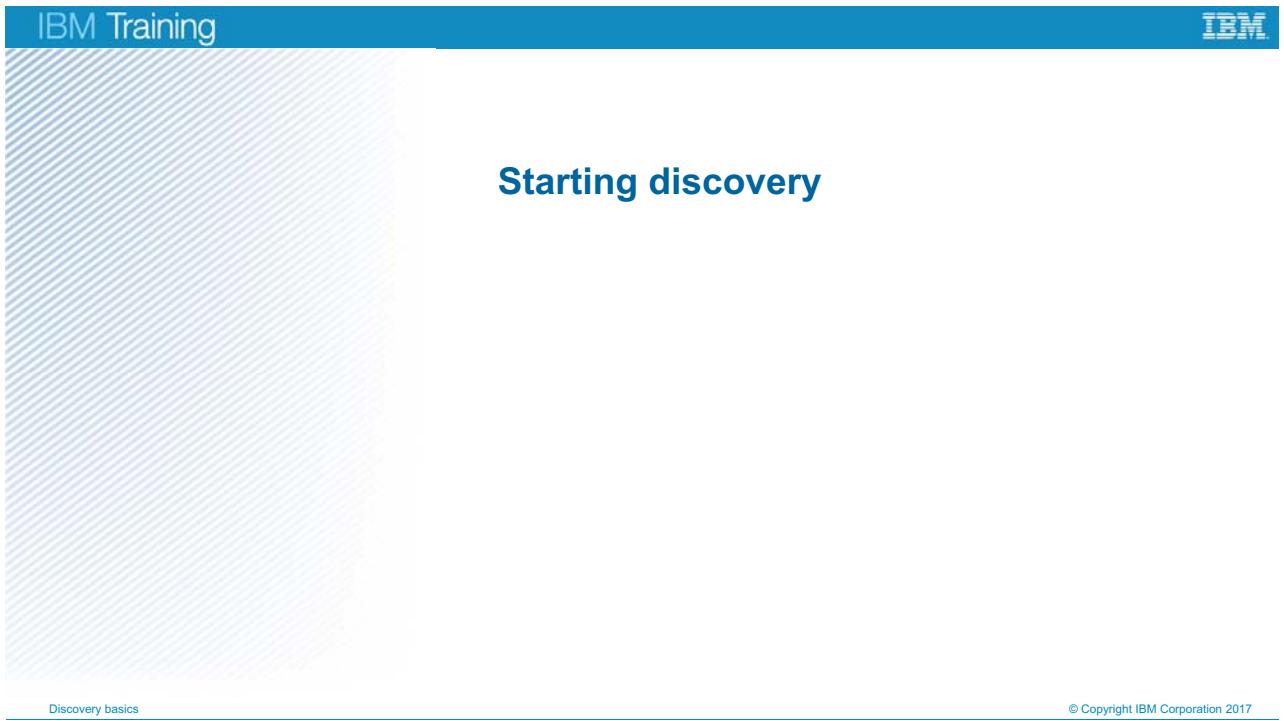


Figure 3-44. Starting discovery

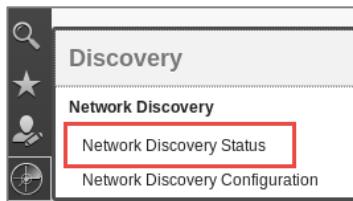
Starting full and partial discoveries

After completing this topic, you should be able to do the following tasks:

- Monitor discovery status
- Conduct a partial discovery
- Schedule full discoveries
- List the four ways to start a discovery

Figure 3-45. Starting full and partial discoveries

Start a full discovery from the GUI



- Select **Network Discovery Status** from the left menu
- Click the drop-down arrow to the right of the green play arrow
- Select **Start Full Discovery**

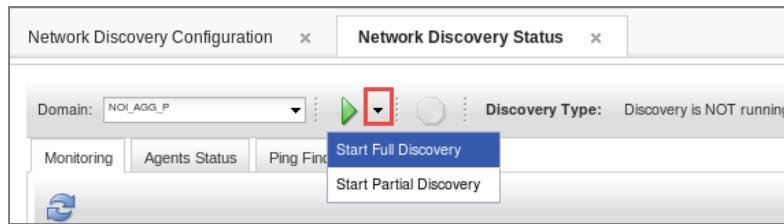


Figure 3-46. Start a full discovery from the GUI

You can also start a discovery from the command line by using the `ncp_disco` command with the `-discoOnStartup` option set to 1. For more information, see the *IBM Tivoli Network Manager IP Edition: Discovery Guide* or view the document at <http://tinyurl.com/h8a2kl5>.

Start a partial discovery from the GUI

- In the **Network Discovery Status** tab, select the drop-down arrow to the right of the green play arrow and select **Start Partial Discovery**

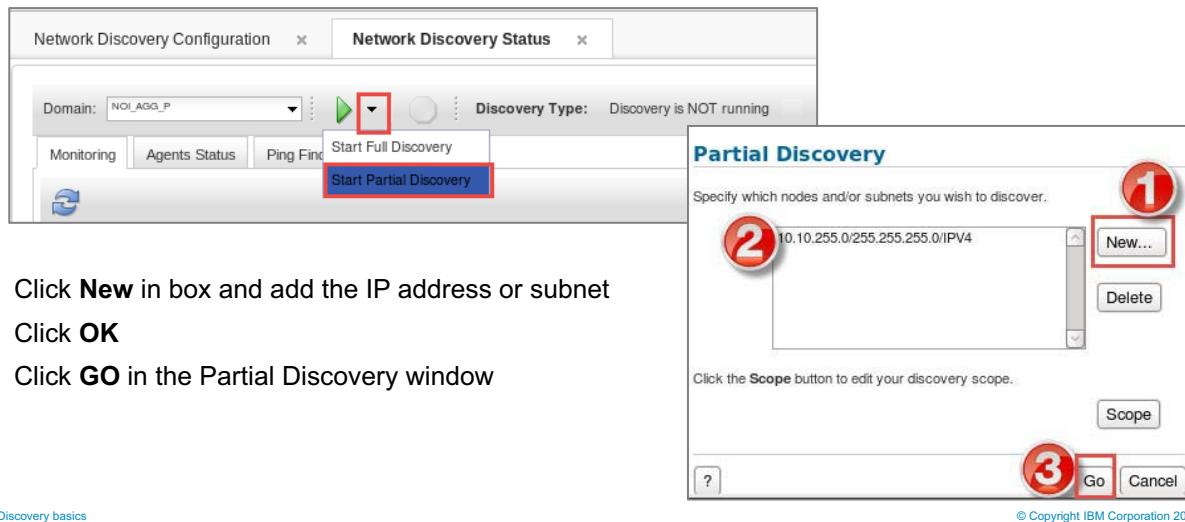


Figure 3-47. Start a partial discovery from the GUI

In dynamic networks, it is common to add multiple devices to the network in a single day. Use this way to discover these devices:

- To instantiate the devices, put in their 32-bit addresses and make sure that they are in scope. Then, start the partial rediscovery.
- If you want to also discover connectivity, make sure that you include the Layer 2 and Layer 3 discovery agents. Then, start the partial rediscovery.
- If you do not have to rediscover the surrounding devices, go to the **Advanced** tab and set Feedback to **Feedback only on full**.



Information

If you stop a running discovery, you must then do a full discovery before you are able to do a partial discovery. If **ncp_disco** remains running after a full discovery, you can run a partial discovery. The partial discovery requires in-memory data from the previous full discovery.

Start scheduled full discoveries

- Uncomment and modify the sample code in **\$ITNMHOME/disco/stitchers/FullDiscovery.stch** to schedule full discoveries

```
#####
// Full Discovery Stitcher
#####
UserDefinedStitcher
{
    StitcherTrigger
    {
        // This is called from the FinalPhase stitcher, during
        // rediscovery.
        // An ActOnTimedTrigger trigger can also be inserted
        // but should not be included until a complete discovery
        // has been made.
        //
        // Activate at 11pm each day.
        // ActOnTimedTrigger(( m_TimeOfDay ) values ( 2300 ) ; );
        //
        // Activate on sixth day of week since Sunday ( Saturday ) at 11pm.
        // Sun - 0, Mon - 1, Tue - 2, Wed - 3, Thur - 4, Fri - 5, Sat - 6
        // ActOnTimedTrigger(( m_DayOfWeek , m_TimeOfDay )
        //                   values ( 6 , 2300 ) ; );
        //
        // Activate on 13th of each month at 2pm.
        // ActOnTimedTrigger(( m_DayOfMonth , m_TimeOfDay )
        //                   values ( 13 , 1400 ) ; );
        //
        // Activate on intervals of 13 hours.
        // ActOnTimedTrigger(( m_Interval ) values ( 13 ) ; );
    }

    StitcherRules
    {
        RecordList discoStatus = RetrieveOQL(
            "select * from disco.status;"
        );
    }
}
"FullDiscovery.stch" [Modified] line 31 of 131 --23%-- col 1
```

Discovery basics

© Copyright IBM Corporation 2017

Figure 3-48. Start scheduled full discoveries

Only the first portion of **\$ITNMHOME/disco/stitchers/FullDiscovery.stch** needs to be modified. Uncomment the lines that you want to modify, making sure that each stitcher command ends with a semicolon (;).

```
UserDefinedStitcher
{
    StitcherTrigger
    {
        // This is called from the FinalPhase stitcher, during
        // rediscovery.
        // An ActOnTimedTrigger trigger can also be inserted
        // but should not be included until a complete discovery
        // has been made.
        //
        // Activate at 11pm each day.
        // ActOnTimedTrigger(( m_TimeOfDay ) values ( 2300 ) ; );
        //
        // Activate on 6th day of week since Sunday (Saturday) at 11pm.
        // Sun - 0, Mon - 1, Tue - 2, Wed - 3, Thur - 4, Fri - 5, Sat - 6
        // ActOnTimedTrigger(( m_DayOfWeek , m_TimeOfDay )
        //                   values ( 6 , 2300 ) ; );
```

```
//  
// Activate on 13th of each month at 2pm.  
// ActOnTimedTrigger( ( m_DayOfMonth , m_TimeOfDay )  
//         values ( 13 , 1400 ) ; );  
//  
// Activate on intervals of 13 hours.  
// ActOnTimedTrigger( ( m_Interval ) values ( 13 ) ; );  
//  
// Activate every 30 minutes  
// ActOnTimedTrigger( ( m_IntervalSeconds ) values ( 1800 ) ; );  
}
```

Start a triggered rediscovery

- When discovery completes, the **ncp_disco** process monitors the **finders.rediscovery** table
 - IP addresses that are found in this table are sent to the discovery engine for rediscovery
- You might want to trigger a partial rediscovery on an IP address when the Netcool/OMNIBus **syslog** or **mttrapd** probes detect alarms that indicate one of the following conditions:
 - When a device experiences a configuration change
 - When a device has a cold start
- Tivoli Network Manager 4.2 can use two methods to automatically trigger a partial rediscovery
 - Method #1 (before 4.1.1): Create a timed trigger on the Netcool/OMNIBus ObjectServer
 - Method #2 (version 4.1.1 and later): Configure the discovery plug-in in the gateway with event information

Figure 3-49. Start a triggered rediscovery

When discovery completes, the **ncp_disco** process monitors the **finders.rediscovery** table. A partial rediscovery begins when the discovery process detects the insert of an IP address into this table. Because of this behavior, you can write custom tools or automations to trigger partial rediscovery on certain network conditions.

When would you want to trigger a partial rediscovery on IP address?

- When a device experiences a configuration change, it can affect your network topology. Examples of such changes would include interface activation or deactivation and routing changes.
- When a device has a cold start, it might indicate that hardware changes were made to the device. An example of such a change would be adding a card or interface to a device.
- How would you detect changes that trigger a partial rediscovery?
 - The Netcool/OMNIBus **syslog** probe can receive configuration change messages from network devices.
 - The Netcool/OMNIBus **mttrapd** probe can receive cold start traps.

You can also automate a triggered rediscovery with one of the following methods:

- **Method #1 (on versions before 4.1.1):** You can create a timed trigger on the Netcool/OMNIBus ObjectServer that recognizes cold start and system configuration change events. The trigger uses a local copy of the **ncp_oql** binary to insert the IP address of the affected device into the **finders.rediscovery** table on the IBM Tivoli Network Manager server.
- **Method #2 (on versions 4.1.1 and later):** Configure the discovery plug-in in the gateway with information about what events trigger a partial rediscovery. When the gateway sees these events, it puts their IP addresses into the **finders.rediscovery** table. A student exercise in this course leads you through the steps of configuring this gateway plug-in.

Considerations for a triggered discovery

- If many of these events occur each day, it might be more efficient to schedule a daily rediscovery of the network
- Each rediscovery involves interrogating the applicable devices and stitching the new information into the existing topology

Figure 3-50. Considerations for a triggered discovery

3.4. Checking discovery status

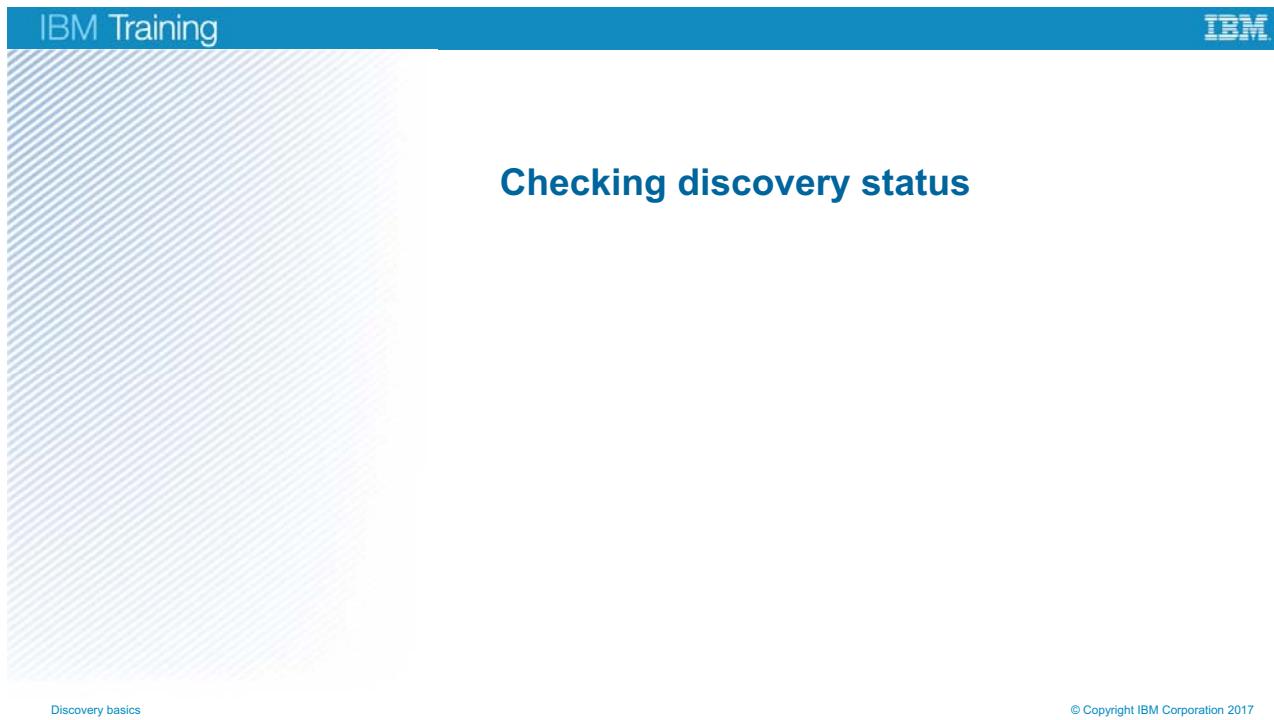


Figure 3-51. Checking discovery status

Checking discovery phase status

- Discovery phase information is under the **Monitoring** section of the **Network Discovery Status**
- You can view the difference in the number of devices that are discovered between the current and previous discovery

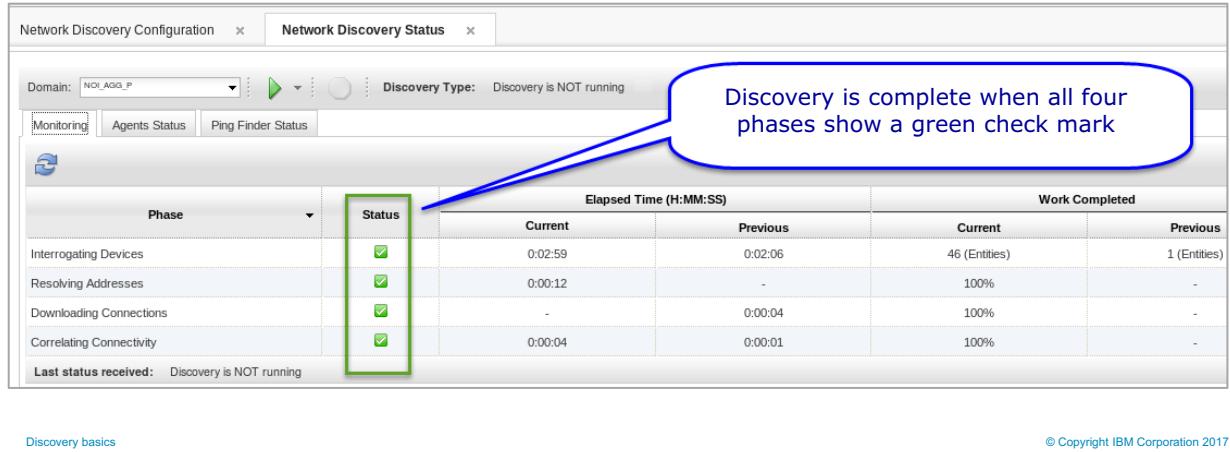


Figure 3-52. Checking discovery phase status

© Copyright IBM Corporation 2017

The **Monitoring** section of the discovery status GUI shows you the completed percentage of each discovery phase:

- Interrogating devices:** This phase begins when the first IP address is found.
- Resolving addresses:** This phase does MAC address to IP address resolution.
- Downloading connectivity:** This phase gets the switch forward database tables.
- Correlating connectivity:** This stage begins final phase stitching to build the layers of connectivity.

This portion of the discovery status GUI also shows the number of devices that are discovered in both the current and previous discovery.

Checking discovery agent status

- Sortable columns indicate state of agent and how many devices in the queue of each agent

Agent	Completes in Phase	State	Total Entities	Outstanding Entities
AssocAddress	Interrogating Devices	✓	4	-
isisExperimental	Interrogating Devices	✓	4	-
LinkStateAdyOSPF	Interrogating Devices	✳	4	1
StandardSwitch	Downloading Connections	✳	1	1
IpForwardingTable	Interrogating Devices	✳	4	3
IpRoutingTable	Interrogating Devices	✳	4	3
MACFromArpCache	Downloading Connections	✳	4	4

Figure 3-53. Checking discovery agent status

The **Agent Status** portion of the discovery status GUI shows how many devices are assigned to each agent. It also shows how many remain in the queue for that agent. Clicking an individual agent's name shows you a list of specific devices that the agents interrogate during discovery.

Checking individual agent status

- Click to highlight a specific agent to see details on what that agent is doing

The screenshot shows a table of agents and their status. The columns are:

Agent	Completes in Phase	State	Total Entities
CiscoOSPFTelnet	Interrogating Devices	<input checked="" type="checkbox"/>	13
CiscoPVC	Interrogating Devices	<input checked="" type="checkbox"/>	13
CiscoRoutedVLANInterface	Interrogating Devices	<input checked="" type="checkbox"/>	13
CiscoSerialInterfaceTelnet	Interrogating Devices	<input checked="" type="checkbox"/>	13
DefaultEthernetHub	Downloading Connections	<input checked="" type="checkbox"/>	14
ExtraDetails	Interrogating Devices	<input checked="" type="checkbox"/>	14
FddiDefault	Interrogating Devices	<input checked="" type="checkbox"/>	14

Below the table, a detailed view for the 'isisexperimental' entity is displayed. It includes a dropdown for 'Entity Status for agent: isisexperimental' with 'Queue' selected, and a table with columns: Identifier, State, Elapsed Time (HH:MM:SS), Despatch Time, and Return Time.

Identifier	State	Elapsed Time (HH:MM:SS)	Despatch Time	Return Time
10.10.255.14		-	Aug 3, 2016 8:34:42 PM	-

Figure 3-54. Checking individual agent status

IBM Training 

Checking ping finder status

Sortable columns show processed subnets



Address	Netmask	Last Pinged	Status
10.10.1.0	255.255.255.0	10.10.1.1	✓
10.10.2.0	255.255.255.0	10.10.2.1	✓
10.10.255.0	255.255.255.0	10.10.255.216	✓
192.168.100.100	255.255.255.255	-	?

Discovery basics

© Copyright IBM Corporation 2017

Figure 3-55. Checking ping finder status

The **Ping Finder Status** shows the list of subnets that supplied to the ping finder and the status of the ping finder.

It is normal to see that ping finders continue their work after discovery agents finish interrogating devices in their respective queues. If you are pinging large subnets inefficiently, it can take a significant amount of time to ping every possible IP address in that subnet. For example, it might take up to 3.5 hours to ping through an entire class B subnet. Many of these pings do not succeed, and it is possible to go several minutes without receiving a ping response in a sparsely populated class A or B network.

So that Tivoli Network Manager can render updated network views in a reasonable time, it follows these guidelines:

- When Tivoli Network Manager finds no new devices for 90 seconds, it puts the finders in a blackout state.
- IP addresses that respond to the finders during a blackout state are placed in a **finders.pending** table. Agents complete their processing and final phase stitchers run. After it builds the new topology, it returns to the finders to see whether any new devices were found after the 90 seconds.

Topic summary

Having completed this topic, you should be able to do the following tasks:

- Monitor discovery status
- Conduct a partial discovery
- Schedule full discoveries
- List the four ways to start a discovery

Figure 3-56. Topic summary

3.5. Discovery configuration tabs

After the first discovery, Tivoli Network Manager administrators use the discovery tabs to modify the discovery configuration. This lesson explains the use of options not available in the discovery wizard.

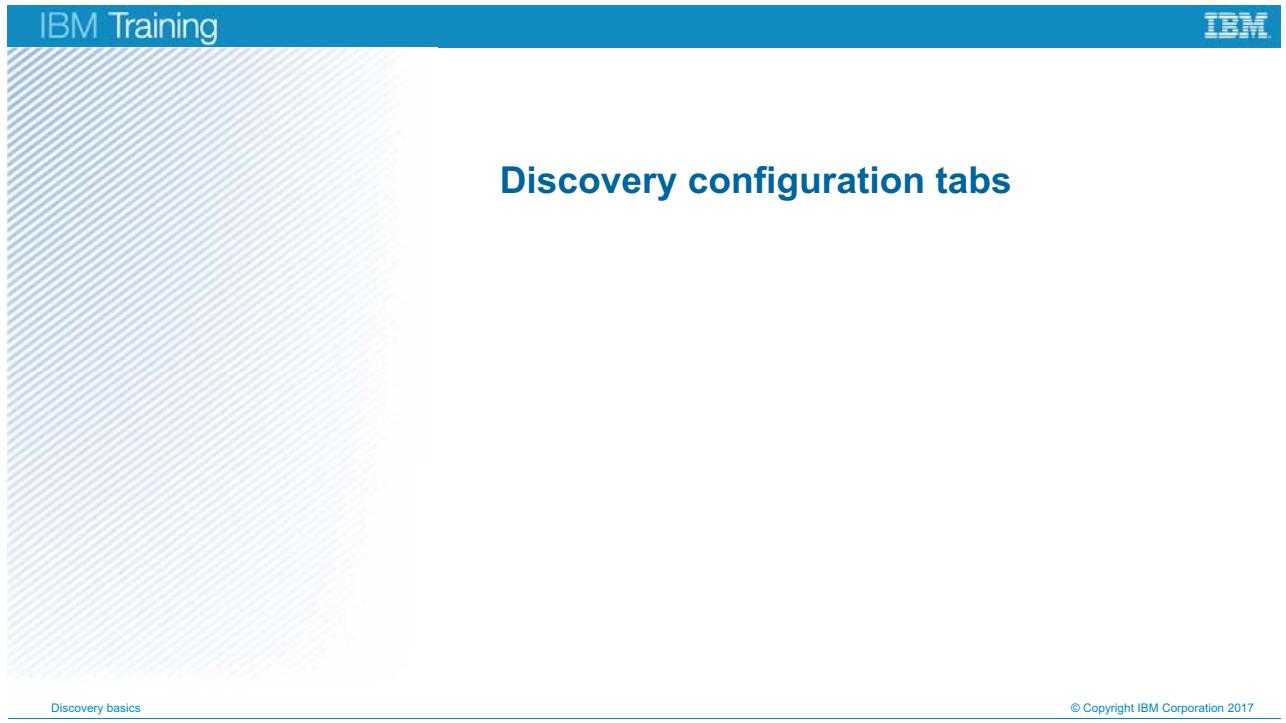


Figure 3-57. Discovery configuration tabs

Using discovery tabs

- After completing this topic, you should be able to configure discovery options in the Discovery Configuration tabs

Figure 3-58. Using discovery tabs

Modifying scope

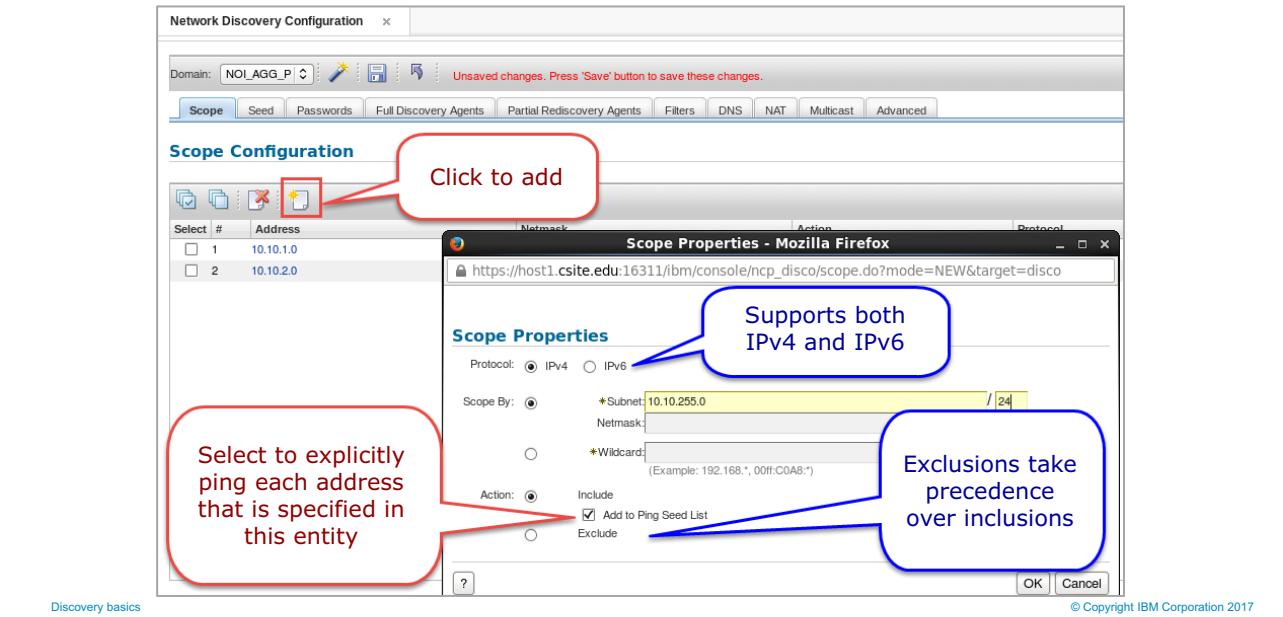


Figure 3-59. Modifying scope

Modifying seeds

- Select ping finder, file finder, or both
- When you use the seed file, include both tabs and spaces as potential delimiters to avoid problems

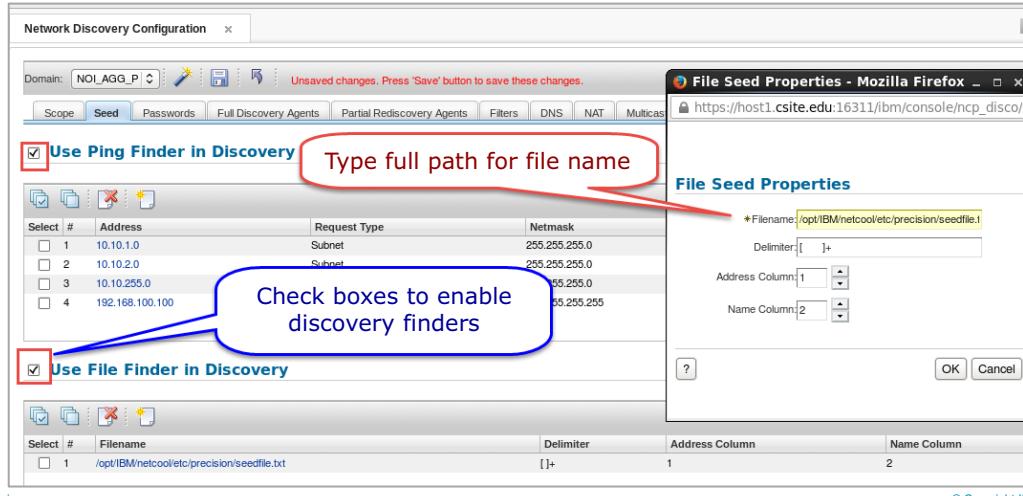


Figure 3-60. Modifying seeds

IBM Training

Modifying community strings

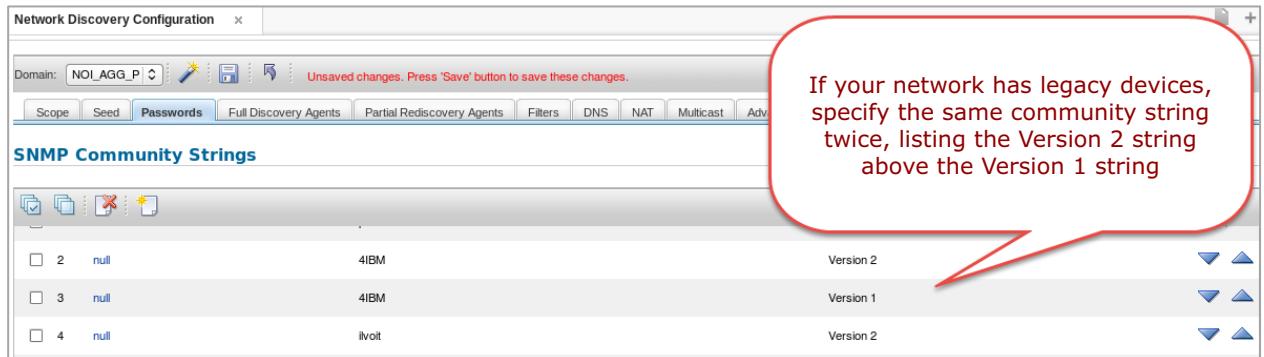


Figure 3-61. Modifying community strings

Selecting agents

- Specify which agents to run in a full discovery on the **Full Discovery Agents** tab
- Specify which agents to run during a partial rediscovery on the **Partial Rediscovery Agents** tab

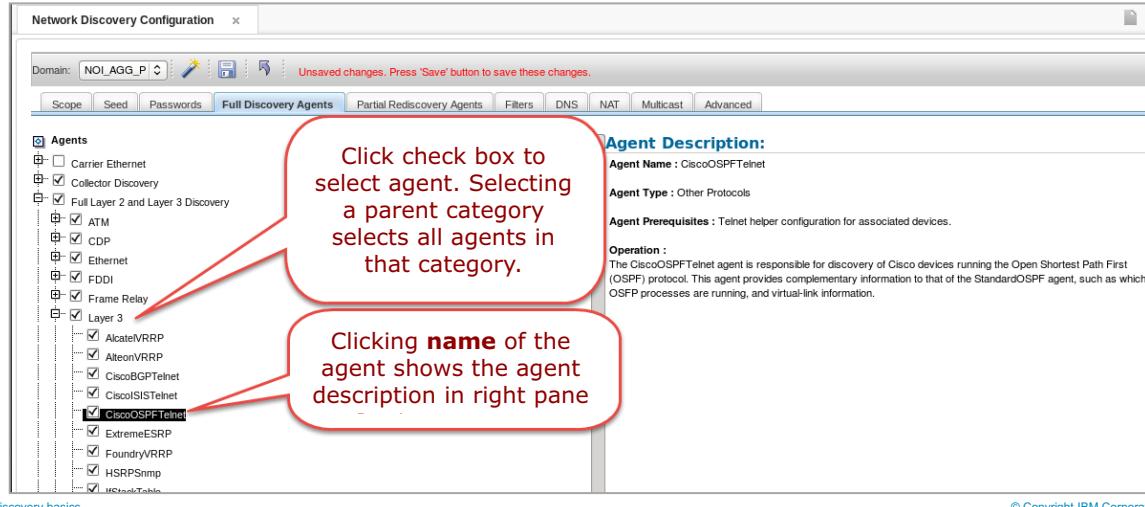


Figure 3-62. Selecting agents

Agents

- Agent configuration files are found in the `$ITNMHOME/disco/agents` directory
- Tivoli Network Manager has three types of discovery agents:
 - A **compiled agent** has all of the queries inside of compiled code so they cannot be modified (example: `TraceRoute.agnt`)
 - A **defined agent** contains all queries inside the agent configuration file (example: `ExtraDetails.agnt`)
 - You can add new queries or comment out queries in the configuration file that you do not need
 - A **combined agent** has its query functions that are compiled to prevent modification
 - Some of its queries are in editable clear text (example: `Interface.agnt`)
- You can create more agents in Perl and run them with the Perl API
 - Look in the following directory for examples of code: `$ITNMHOME/disco/agents/perlAgents`

Figure 3-63. Agents

Domain Name Service (DNS)



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-64. Domain Name Service (DNS)

Tivoli Network Manager can use four sources of name resolution:

- Hosts file
- DNS server
- System DNS
- The **sysName** value that is obtained from an SNMP query

Tivoli Network Manager processes these sources in the order that is shown in the GUI.

If you select the **Enable sysName Naming** option on the Advanced tab, Tivoli Network Manager always uses the **sysName** value of the device, provided discovery agents retrieved the information from the device.



Troubleshooting

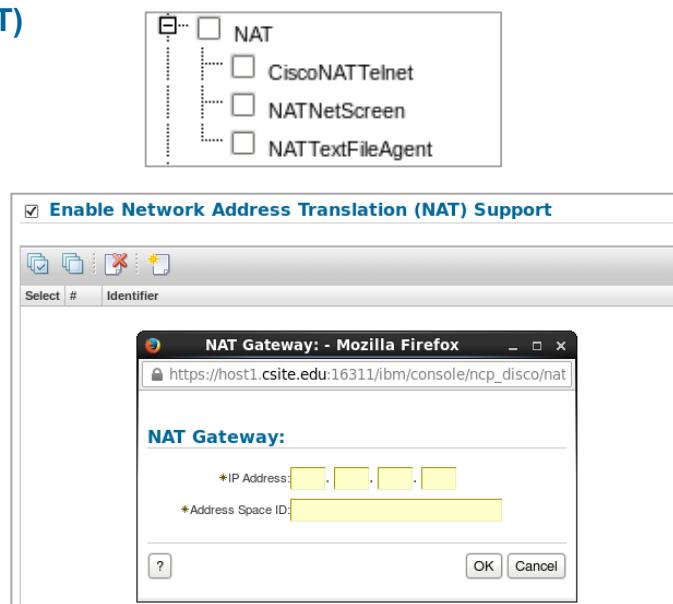
If your discovery shows devices only by their IP address, it means that Tivoli Network Manager was unable to resolve the name with the name resolution sources you specified. You can usually resolve this problem by enabling sysName naming in the Advanced tab or by providing a hosts file.

**Hint**

Large networks often have both a primary and a backup DNS server. If you use the DNS server for resolving names during discovery, specify the backup DNS server as the source of name information. This selection prevents any potential degradation of DNS response time to your users.

Network Address Translation (NAT)

- NAT must be static 1-to-1 addressing
- Agents can query Cisco and Netscreen devices for NAT information
 - Configure NAT agents under the **Agents** tab
- A **NATTextFileAgent** that reads static NAT addressing information from a **NATTranslations.txt** file
- In the GUI, configure the NAT Gateway and an address space name



Discovery basics

© Copyright IBM Corporation 2017

Figure 3-65. Network Address translation (NAT)

These network address ranges are private non-routable IP addresses.

- Class A: 10.0.0.0 – 10.255.255.255
- Class B: 172.16.0.0 – 172.31.255.255
- Class C: 192.168.0.0 – 192.168.255.255

Therefore, an IP address within these ranges is 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 the Internet Assigned Numbers Authority (IANA) or an internet registry. Addresses within this private address space are unique only within a given private network.

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. This identifier is appended to the record for each entity found inside the NAT domain. With this identifier, Tivoli Network Manager can poll and manage these devices.

Restrictions on NAT discovery

Management of NAT environments with Tivoli Network Manager is subject to the following restrictions:

- 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.
- Network Manager cannot discover or manage devices within a NAT domain that have only private IP addresses.
- 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 information 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 **NATTTextFileAgent** discovery agent with a flat file of public-to-private IP address mappings.

Advanced discovery configuration

- Choices on this tab affect the way discovery runs, such as:
 - Alter the root cause analysis by including or excluding VLAN modeling
 - Alter the appearance of MPLS networks by changing the discovery method
 - Use routing table information
 - Verify whether the interface can be pinged

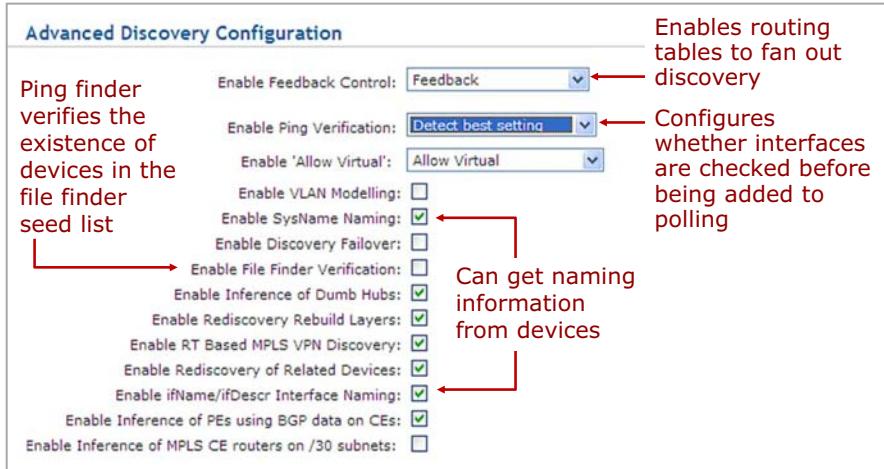


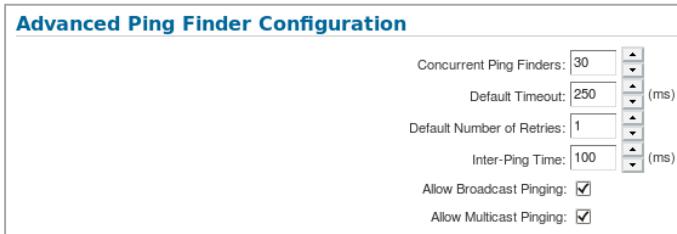
Figure 3-66. Advanced discovery configuration



Information

Ignore the **Enable Inference of Dumb Hubs** option. That selection is a deprecated feature but still exists in the GUI.

Advanced ping finder configuration



- **Concurrent Ping Finder:** Specify the number of threads that are used by the ping finder
- **Default Timeout:** Specify the maximum time, in milliseconds, to wait for a reply from a pinged address
- **Default Number of Retries:** Specify the number of times to retry pinging a device after a failed initial ping
- **Inter-Ping Time:** Specify the interval, in milliseconds, between pinging the addresses in a subnet or list
- **Allow Broadcast Pinging:** To enable broadcast address pinging, select this check box
- **Allow Multicast Pinging:** To enable multicast address pinging, select this check box

Discovery basics

© Copyright IBM Corporation 2017

Figure 3-67. Advanced ping finder configuration

The value for **Concurrent Ping Finders** can usually be increased to a value greater than the default value. Check with your system administrator before you increase the number of ping finder threads.



Hint

You can usually set the number of ping finder threads to 30 without negatively affecting system performance.

Advanced SNMP helper configuration

Advanced SNMP Helper Configuration

Concurrent SNMP Helpers:	300	(ms)
Timeout:	3000	(ms)
Number of Retries:	1	
GetNext Slowdown:	20	(ms)
GetNext Boundary:	5000	

- **Concurrent SNMP Helpers:** Specify the number of simultaneous threads that the SNMP helper can use
- **Timeout:** Specify the maximum time, in 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
- **GetNext Boundary:** Specify the minimum number of **GetNext** requests to be issued when a non-scalar SNMP variable is retrieved from a device

Discovery basics

© Copyright IBM Corporation 2017

Figure 3-68. Advanced SNMP helper configuration

Two advanced parameters can prevent network routers with large routing tables from experiencing increased CPU utilization when they are queried about routing information.

- The `m_GetNextSlowDown` parameter is applied when the number of separate **GetNext** requests that are issued to retrieve a non-scalar SNMP variable exceeds the value of the `m_GetNextBoundary` parameter.
- The `m_GetNextBoundary` parameter is applied before the delay that is specified by the `m_GetNextSlowDown` parameter is introduced.



Hint

The value for **Concurrent SNMP Helpers** can typically be increased to 300. Check with your system administrator before you change this value.

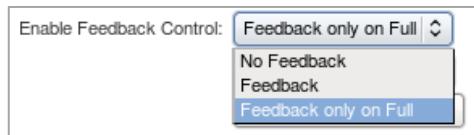
Advanced DNS helper configuration

- **Concurrent DNS Helpers:** Specify the number of threads that the DNS helper can use during discovery
- **Default Timeout:** Specify the maximum time, in milliseconds, to wait for a response from a device

Advanced DNS Helper Configuration	
Concurrent DNS Helpers:	<input type="text" value="10"/>  
Default Timeout:	<input type="text" value="0"/>   (s)

Figure 3-69. Advanced DNS helper configuration

Enable Feedback option on Advanced tab

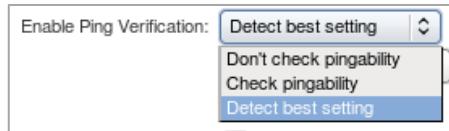


- **No Feedback:** Feedback is disabled for all discoveries
- **Feedback:** Feedback is turned on for full and partial discoveries
- **Feedback Only on Full:** This default setting has feedback that is turned on for full discoveries but turned off for partial discoveries

Figure 3-70. Enable Feedback option on Advanced tab

Feedback is a discovery mechanism that gets a list of attached subnets on a device and then passes those subnets to the ping finder for ping sweeping. Only subnets that are included within the specified discovery scope are passed to the ping finder.

Ping verification options



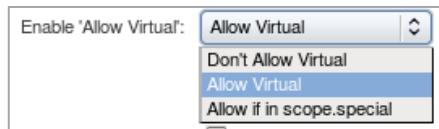
- **Do not Check Pingability:** None of the discovered interfaces are checked to see whether they can be pinged
 - Interfaces are polled regardless of whether they can be pinged at discovery
- **Check Pingability:** After discovery, every discovered interface is checked to see whether it can be pinged
- **Detect Best Setting:** This setting is the default
 - If feedback control is enabled, after discovery, every discovered interface is checked to see whether it can be pinged

Discovery basics
© Copyright IBM Corporation 2017

Figure 3-71. Ping verification options

This pingability 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. These options work only if you select either **Feedback** or **Feedback only on Full** on the **Advanced** options tab.

Allow Virtual option on Advanced tab



- **Do not Allow Virtual:** This option prevents the discovery of virtual IP addresses

- **Allow Virtual:** This default setting discovers virtual IP addresses

- **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

Figure 3-72. Allow Virtual option on Advanced tab

Other advanced discovery options

- Enable VLAN modeling
- Enable sysName Naming
- Enable Caching of Discovery Tables (or Enable Discovery Failover)
- Enable File Finder Verification
- Enable Rediscovery Rebuild Layers
- Enable RT-based MPLS VPN Discovery
- Enable Rediscovery of Related Devices
- Enable ifName / ifDescr Interface Naming
- Enable inference of PEs using BGP data on CEs
- Enable Inference of MPLS CE routers on /30 subnets

Figure 3-73. Other advanced discovery options

A discovery that runs in failover mode stores data on the server hard disk drive. This process results in decreased discovery speed. Discovery failover creates cache files for each agent and stitcher that runs during discovery. If discovery is run in failover mode, an interrupted discovery can resume rather than restart.

The Advanced Discovery Configuration section of the Advanced tab in the Discovery Configuration GUI refers to **Enable Caching of Discovery Tables**. However, the help screen for this feature refers to this as **Enable Discovery Failover**. These terms are interchangeable.



Important

If you need to run a partial discovery at any time, you need to enable caching of discovery tables. A partial discovery requires data from the previous full discovery. If you stop the `ncp_disco` process, you lose the `workingEntities.finalEntity` table and other tables that are used by the partial discovery. However, if you enabled discovery caching and you have at least one full discovery, then you can run a partial discovery at any time. The caching process writes persistent data about the discovery to the hard disk drive on the server.

Topic summary

After you complete this topic, you can do the following task:

- Configure discovery options in the **Discovery Configuration** tabs

Figure 3-74. Topic summary

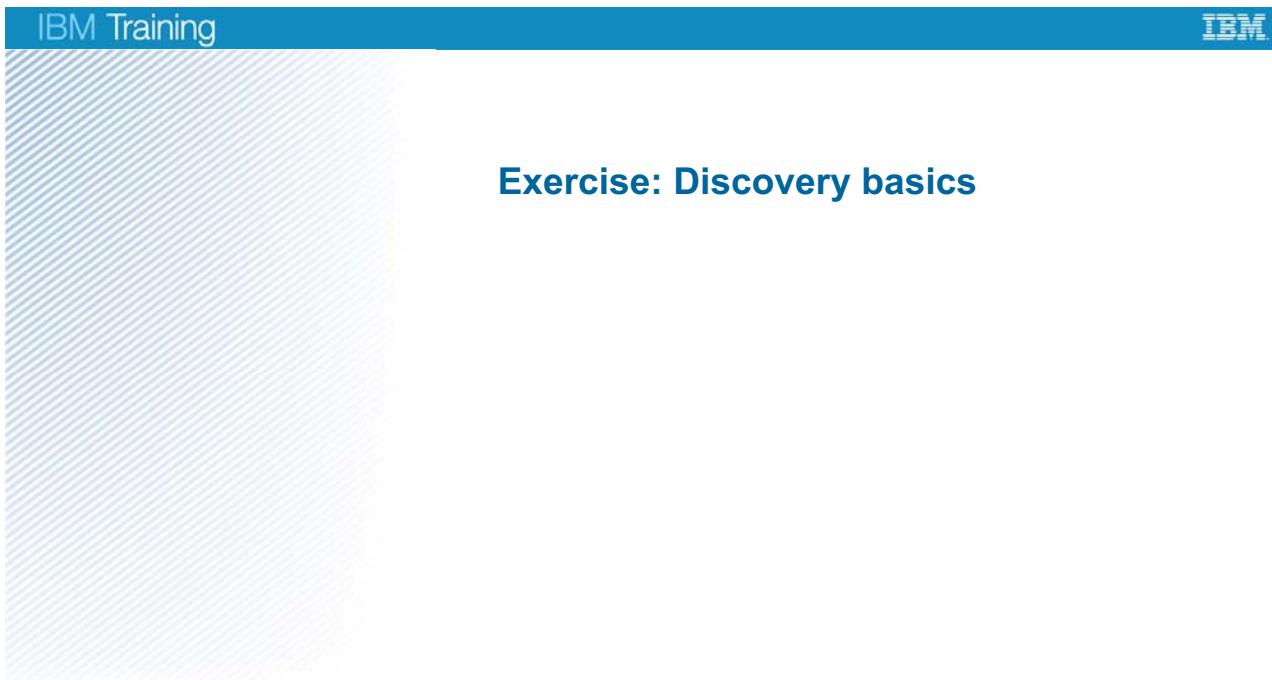


Figure 3-75. Exercise: Discovery basics

Complete the exercise for this unit.

Review questions

1. When do you do an unscoped discovery?
2. What does Tivoli Network Manager do with interfaces that have IP addresses that are not in the discovery scope?
3. In what directory do you find configuration files for Tivoli Network Manager?
4. When unsure about whether network devices on a subnet are using SNMP v1 or v2, what must you do to enhance discovery performance?
5. What is always true about exclusions to a network discovery scope?

Figure 3-76. Review questions

Write your answers here:

Unit summary

- Configure a basic network discovery
- Describe the three fundamental tasks of discovery
- Schedule a regular full discovery of a network
- Use ping and file finders
- Run a network discovery
- Explain the function of the feedback mechanism during discovery

Figure 3-77. Unit summary

Review answers

1. Never do an unscoped discovery when it is avoidable.
2. The interface is in the topology because it is inside an in-scope chassis. The interface record has `m_IsActive=2` set to prevent polling of the out-of-scope interface.
3. Configuration files for Tivoli Network Manager are in the `$NCHOME/etc/precision` directory.
4. Enter each community string twice and always specify the SNMP v2 community string first. In this way, devices that are running SNMP v2 can be queried more efficiently.
5. Exclusions to the discovery scope override inclusions.

Discovery basics

© Copyright IBM Corporation 2017

Figure 3-78. Review answers

Unit 4. Advanced discovery options



Figure 4-1. Advanced discovery options

Estimated time

01:30

Overview

This unit introduces discovery filtering, use of discovery collectors, and configuring discovery for multicast networks. It also equips students to optimize discovery.

How you will check your progress

- Answer checkpoint questions
- Complete student exercises

References

<http://tinyurl.com/he3Injz>

<http://tinyurl.com/hl2kpjb>

This unit covers:

- Less-used discovery configuration features
- Multicast discovery
- Discovery optimization
- Discovery troubleshooting

Unit objectives

- Generate a seed file from a previous discovery and use it to reduce finding time
- Properly configure ping sweeping for a class B subnet
- Configure discovery of multicast networks
- Gather data for technical support
- Optimize discovery to reduce Phase 1 discovery time
- Restart discovery with an empty topology database

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-2. Unit objectives

4.1. Configure discovery of multicast networks

The plethora of online meetings, auctions, training, and webinars drive an increased demand for multicast services. Multicasting allows a variety of communication configurations to make the most effective use of bandwidth for a wide variety of applications. Tivoli Network Manager discovers multicast networks and automatically builds views of multicast distribution trees (MDT) and multicast groups. This capability enables support personnel to quickly identify and troubleshoot multicast resources.

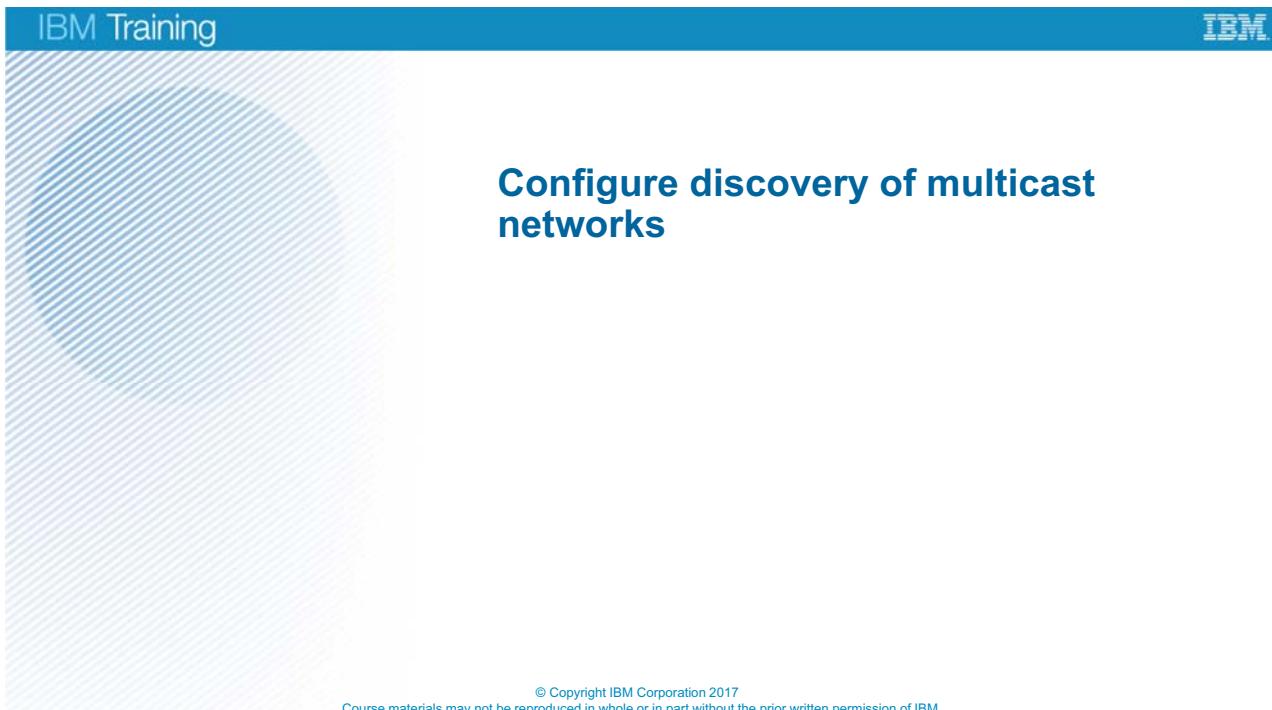
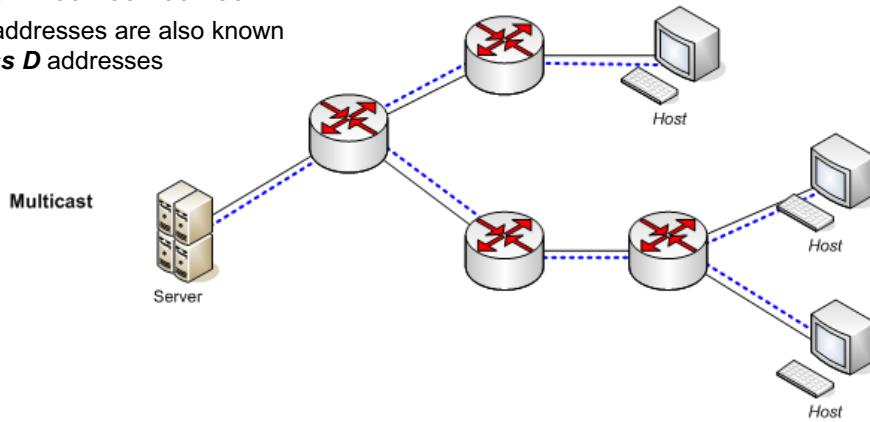


Figure 4-3. Configure discovery of multicast networks

Multicasting

- **Multicasting** is a way to group devices together so that a single data source can send information to multiple recipients in a single data stream
- The range of multicast addresses is 224.0.0.0 – 239.255.255.255
 - These addresses are also known as **Class D** addresses



Advanced discovery options

© Copyright IBM Corporation 2017

Figure 4-4. Multicasting

Some devices use some multicast addresses for specific purposes:

- Cisco routers listen for EIGRP updates on 224.0.0.10
- Cisco routers listen for OSPF updates on 224.0.0.5
- Routing Information Protocol (RIP) Version 2 uses 224.0.0.9 as a destination IP address for routing update packets.
- 224.0.0.10 for IGRP Routers
- 224.0.0.11 for Mobile-Agents

Administratively scoped addresses

IP Multicast uses the range of addresses from 239.0.0.0 through 239.255.255.255 inclusive as “limited” or administratively scoped addresses. These addresses are reserved for private use by an organization and are typically not routed beyond an organization’s autonomous system (AS).

Globally scoped addresses

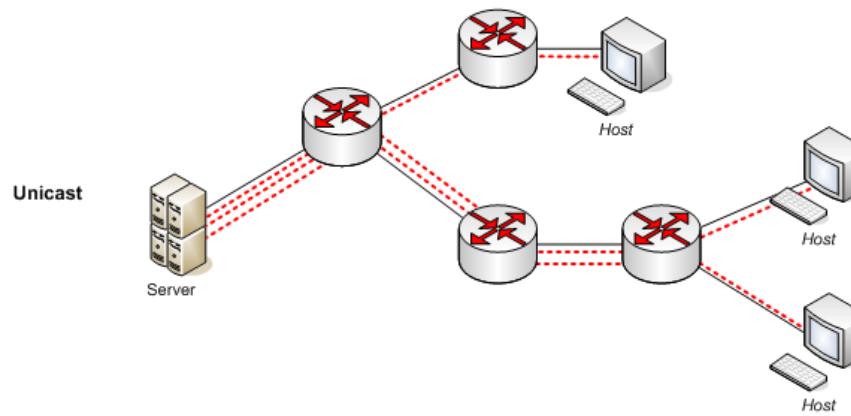
IP Multicast uses the range of addresses from 224.0.1.0 through 238.255.255.255 inclusive as “globally scoped” addresses. These addresses are used to multicast data between organizations across the internet. Examples of globally scoped addresses include 224.0.1.1 for Network Time Protocol (NTP) and 224.0.1.32 for Multicast Traceroute (MTRACE).

GLOP addresses

RFC2770 suggests that organizations with already-assigned autonomous systems (AS) numbers reserve the address range from 233.0.0.0 through 233.255.255.254 (233.0.0.0/8). The organization can embed its AS number into the second and third octets of addresses within the 233.0.0.0/8 range. For instance, the AS number 5378 is represented as 1502 in hexadecimal. Separate this value into two octets of 15 and 02, and then convert them into decimal. This result gives the multicast subnet of 233.21.2.0/24 that is globally reserved for AS5378 to use.

Unicasting

Unicasting requires a data source to send an individual data stream to each recipient



Advanced discovery options

© Copyright IBM Corporation 2017

Figure 4-5. Unicasting

Unicasting is less efficient than multicasting because it requires an individual data stream to each recipient.

Reasons for using multicasting

One to Many	Many to One	Many to Many
Database updates	Resource discovery	Multimedia conferencing
Live concerts	Data collection	Synchronized resources
Broadcasts	Auctions	Concurrent processing
Newsfeeds	Polling	Collaboration
Push media	Moderated applications	Distance learning
Monitoring		Multi-player games

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-6. Reasons for using multicasting

At some point, you participated in an activity that used multicasting.

Multicasting discovery features

IBM Tivoli Network Manager 4.2 contains features to discover and model Protocol Independent Multicast (PIM) sources (currently, only IPv4 addresses)

- These features help you answer questions like these examples:
 - What multicast groups do the routers support?
 - Where are the sources and receivers for a specific multicast group?
 - How many IP Multicast Routes (IPMROUTE) does a specific router have?
 - What does the distribution tree look like for a particular multicast group?
 - What does the Internet Group Membership Protocol (IGMP) cache look like?

Figure 4-7. Multicasting discovery features

Multicast discovery agents

- The **StandardPIM.agnt** is a compiled agent that can discover PIM enabled interfaces, PIM adjacencies, candidate rendezvous points (RPs), and bootstrap routers
- StandardIGMP
- StandardIPMRoute

Figure 4-8. Multicast discovery agents

Multicast discovery stitchers

- CreatePIMNetworksCollection.stch
- CreatePIMProtocolEndPoints.stch
- CreatePIMServices.stch
- CreatePIMTopology.stch
- PIMLayer.stch
- PreProcessPIMEndPointData.stch
- PreProcessPIMServiceData.stch
- PreProcessPIMTopology.stch

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-9. Multicast discovery stitchers

IP multicast configuration tasks

- Select agents
 - StandardIGMP
 - StandardIPMRoute
 - StandardPIM
- Configure **Multicast Groups** and **Multicast Sources** scoping

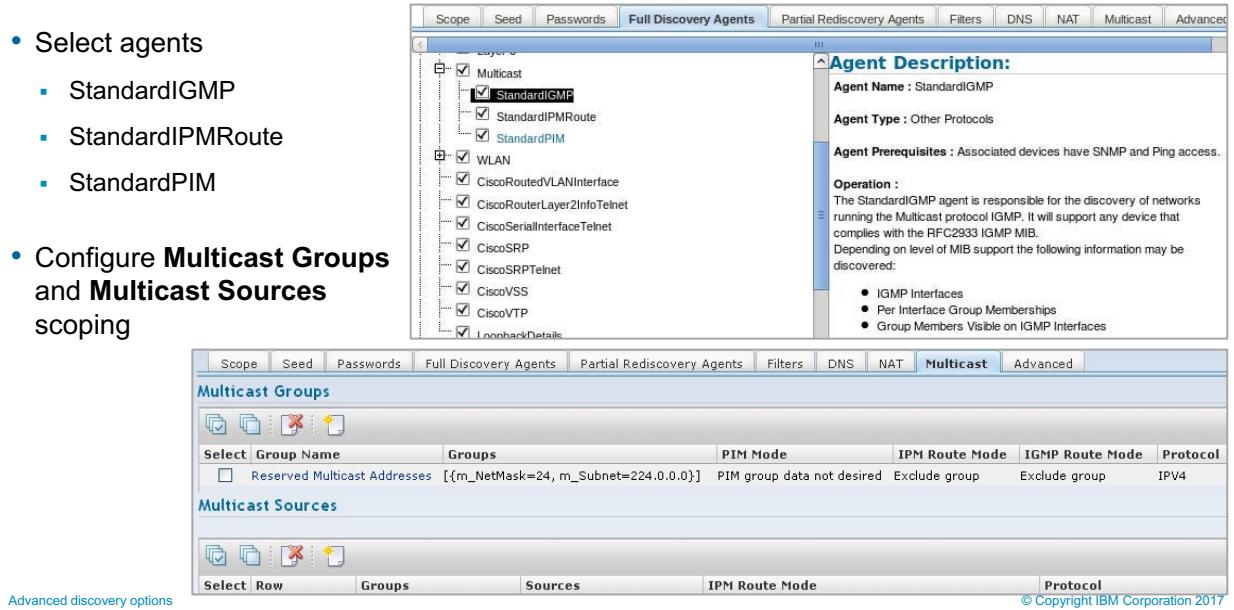
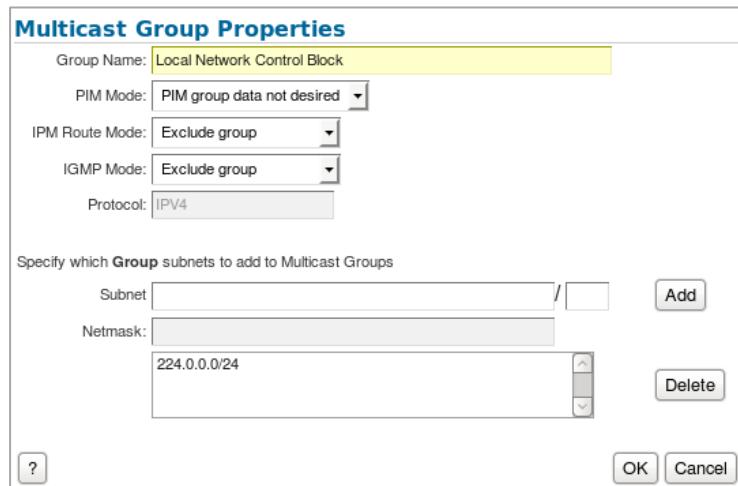


Figure 4-10. IP multicast configuration tasks

Configure multicast groups

- Assign a group name
- Determine whether to include or exclude information from:
 - Protocol Independent Multicast (PIM)
 - Internet Protocol Multicast (IPM) group data
 - Internet Group Management Protocol (IGMP)
- Add group subnets for the group



[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-11. Configure multicast groups

A multicast source sends data to a group of recipients. You must specify the Class D subnets that apply to a named group of devices. Click the question mark (?) in this properties window for context-sensitive help that explains the options on this screen.

- **Group Name:** Specify a descriptive 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. Hosts use this protocol to insert and remove themselves from multicast groups.
- **Protocol:** Only IPv4 is supported currently. You cannot select another protocol in this version.

You can name the multicast group anything that you want. Name it something descriptive of the function so that the data is more meaningful to others.

A multicast group represents an arbitrary group of receivers that are interested in receiving data from a unicast source.

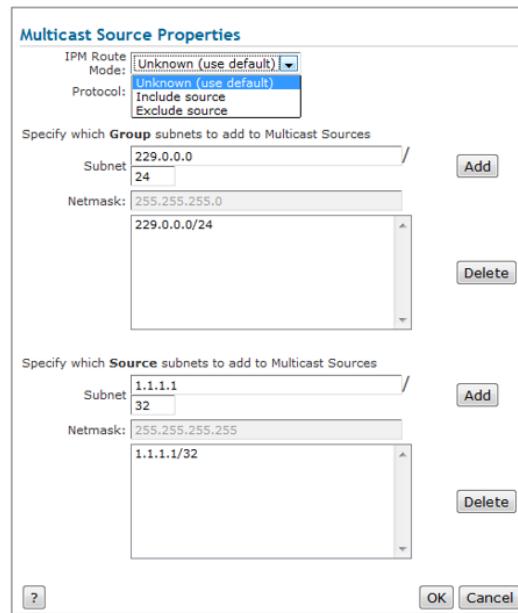
- For a receiver to receive data from a multicast source, it must be a member of the correct multicast group.
- To receive this traffic, receivers are said to join the multicast group. When a source sends data to a multicast group, all members of the group receive the data.

Configure a group name descriptive of the function of the devices in the multicast group. Consider the following examples.

- IP Multicast Group for IP TV
- RFC2565 Administrative Scope IP Multicast

Configure multicast sources

- Define whether to include or exclude the IPM group
- Only IPv4 protocol is supported
- Enter a subnet and netmask for a group subnet to add to multicast sources
- Enter source subnets to add to multicast sources



[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-12. Configure multicast sources

If you have a single host multicasting to a group, you can specify a single host IP address with a 32-bit (255.255.255.255) subnet mask. This configuration is appropriate for a one-to-many multicast network.

If you have several hosts that are communicating to a group of recipients, you can specify source subnets. Any hosts on these source subnets that are properly configured can multicast to the group of recipients. This configuration is appropriate for a many-to-many multicast network.

Usage: IP multicast views

- IGMP Group View
 - All routers with IGMP sessions, partitioned by multicast group
 - Uses Layer 3 topology relevant to IGMP interfaces
- Structure browser shows IGMP and IPMRoute services
- Multicast Routing MDT View
 - All routers with knowledge of IP Multicast Routing
 - Per multicast distribution tree MDT (source or group pairs)
 - Topology reflects MDT routing data
- Right-click tools

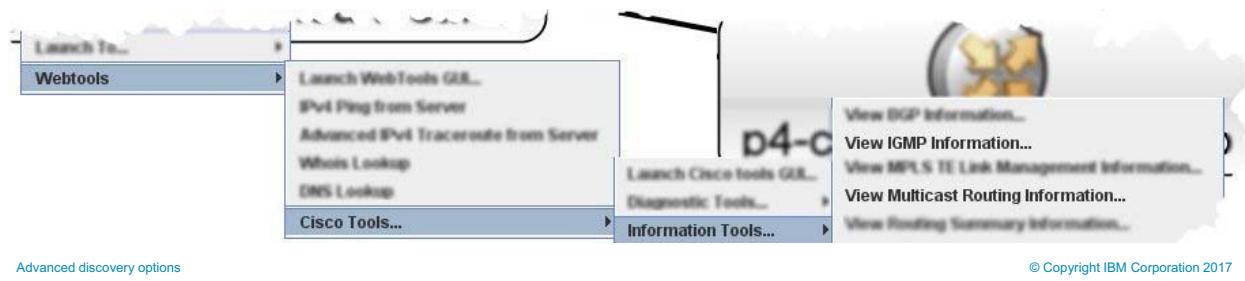


Figure 4-13. Usage: IP multicast views

IP multicast integration

- Multicast data is stored in NCIM tables (IPMRoute and IGMP)
- Data that is offered in NCIM (IPMRoute) includes:
 - Multicast routing data (upstream, downstream)
 - Interfaces that are involved in multicast routing
 - Multicast sources and groups referred to as multicast distribution trees (MDTs)
 - Various metrics
- Data that is offered in NCIM (IGMP) include:
 - IGMP interfaces
 - Per-interface group memberships
 - Group members visible on IGMP interfaces
 - Various metrics

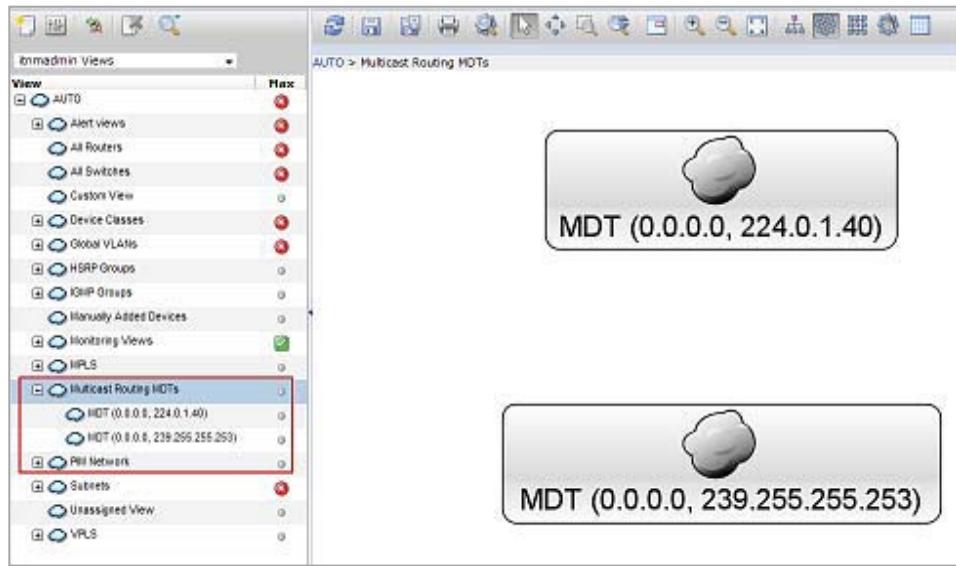
[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-14. IP multicast integration

IGMP group membership view

- Shows all routers that are known to participate in IGMP for a specified group
- Drill down from top levels to see group members



Advanced discovery options

© Copyright IBM Corporation 2017

Figure 4-15. IGMP group membership view

Troubleshooting multicast discovery

- Device verification: Check device configuration and accessibility
- Agent verification: Check agent configuration and output (verify agent output by writing discovery to cache)
- NCIM verification
 - Understand the common causes of failure
 - Check the `$NCHOME/var/precision/ncp_model.DOMAIN_NAME.trace` file for errors
 - Use OQL to query NCIM for key multicast data
- GUI verification
 - NCIM data quality
 - Quantity of data
 - Network views are collecting the multicast routers correctly
 - Check IGMP and IPMRoute services

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-16. Troubleshooting multicast discovery

Topic summary

Having completed this topic, you should be able to do the following tasks:

- Identify examples of multicast networking
- Identify the IPv4 address range for multicast networking
- Complete the steps necessary for configuring discovery of multicast networks
 - Specify the source
 - Specify the recipients
 - Define the group

Figure 4-17. Topic summary

4.2. Discovery filtering

Most organizations want to exclude certain types of devices or interfaces from their discovery. For example, many organizations want to see only routers and switches in their topology. They want to exclude printers and workstations from their discovery. Others might want servers to be included in their topology maps, but not workstations. Tivoli Network Manager has a prediscovery filter to eliminate unwanted devices from your network discovery. It can also eliminate certain types of interfaces from your network discovery.

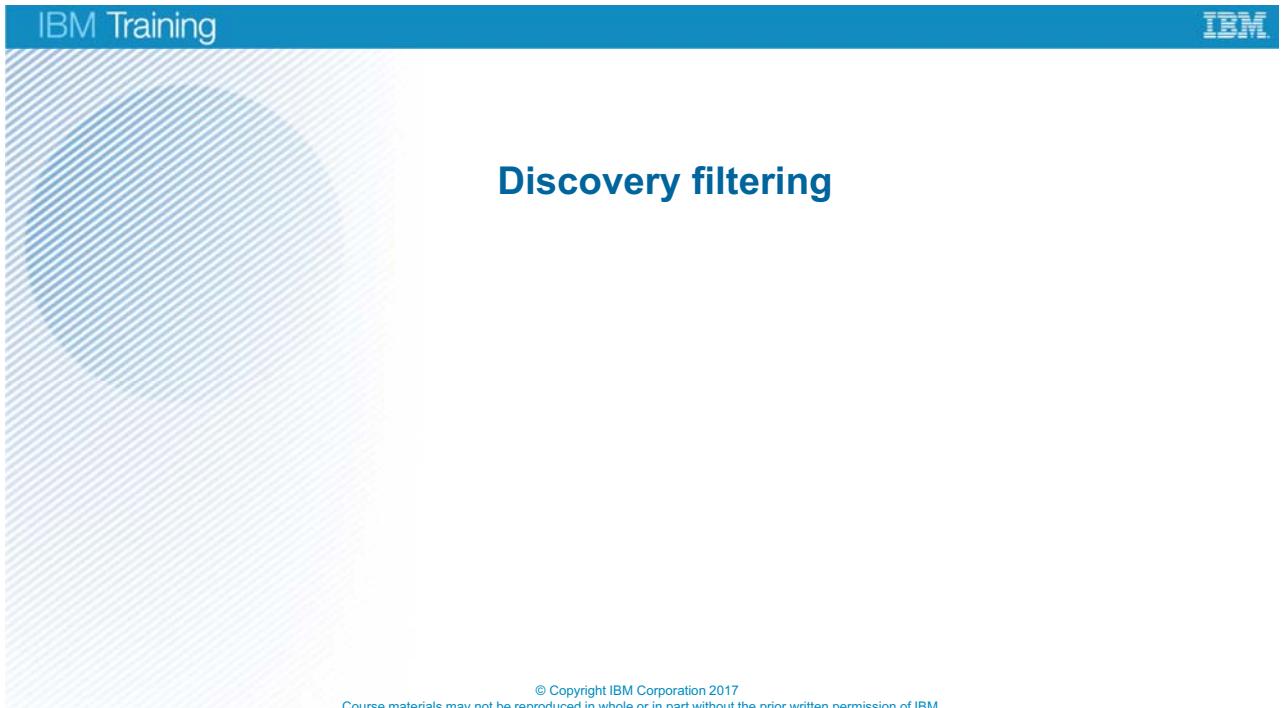


Figure 4-18. *Discovery filtering*

Discovery filtering eliminates items that you do not want in your discovery.

Scenario 1: College campus classroom

Scenario

- A college campus has a 172.16.0.0/16 network
- One classroom has Linux hosts from 172.16.0.65 to 172.16.0.78 with firewalls
- Avoid sending ICMP or SNMP traffic to these classroom devices

Exclusions always
take precedence
over inclusions

Solution

- Modify the scope of the discovery
- Include the 172.16.0.0/16 network
- Exclude the 172.16.0.64/28 (255.255.255.240) network
 - This setup prevents SNMP agents from querying classroom devices
 - The exclusion overrides the inclusion, and no devices in the exclusion are instantiated
 - This exclusion prevents the ping finder from pinging any of the excluded devices

Figure 4-19. Scenario 1: College campus classroom

Exclusions always take precedence over inclusions. In this example scenario, a few devices on a Class B network are excluded from discovery. The entire Class B network is included in the scope of discovery, and the ping finder is configured to ping sweep the subnet. Because the exclusion has precedence over the inclusions, none of the excluded devices are pinged or receive SNMP queries from Tivoli Network Manager.

You can download a subnet calculator application to your smartphone. This tool makes it easier to specify subnets with the appropriate mask to exclude a few devices.

Scenario 2: Dynamic dial backup interfaces

- Customer is a managed service provider (MSP) who needs to discover and monitor the core network
- Customers have many routers with dynamic dial backup interfaces that are used when their primary WAN connection is disabled
 - These interfaces are normally down, but activate when the T1 serial interfaces lose their connection to the remote site
 - When the dynamic dial backup lines are used, the customer incurs telecommunications charges
- The customer wants you to show these interfaces, but not ping them
- Another vendor attempted to discover the customer's network and generated more than \$8,000 of line charges with one discovery effort
- Discover the network without activating the dynamic backup links

Figure 4-20. Scenario 2: Dynamic dial backup interfaces

In this case, the customer wants to discover the chassis and the interfaces but avoid activating dynamic dial backup interfaces. If discovery activates the dynamic dial backup interfaces, it results in incurring telecommunications charges.

The easiest way to avoid this problem is to exclude the dynamic dial backup interfaces from the discovery. Since exclusions take precedence over inclusions, the dynamic dial backup interfaces are not pinged or queried. Tivoli Network Manager can complete discovery without activating these interfaces and incurring charges. The interfaces are still visible in the Device Structure browser, but they are flagged to prevent the poller from polling the interfaces.

Scenario 2: Use scoping solution

Solution

- Explicitly exclude each of the dynamic dial backup interface addresses from the scope of the discovery

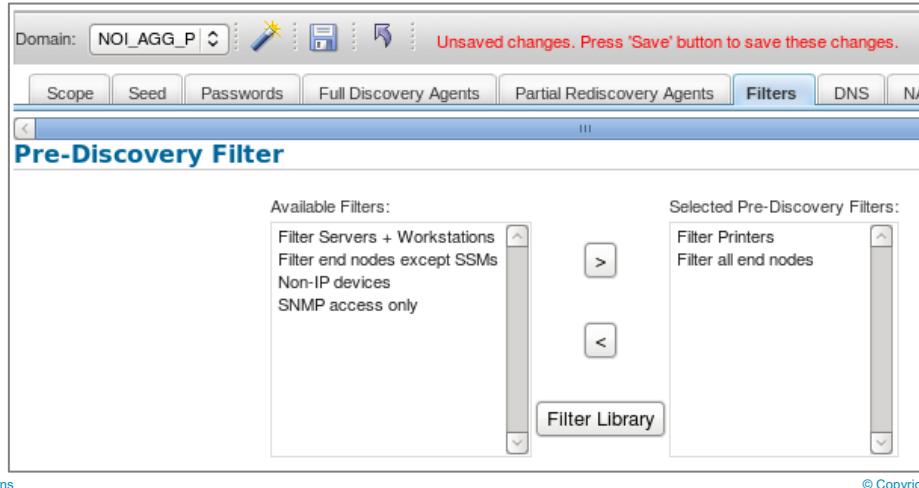
Effect

- The dynamic dial backup interfaces are instantiated
 - Although the dynamic dial backup interfaces are out of scope, the chassis in which they are located are in scope
 - The interfaces are instantiated but discovery does not continue through that interface
- The **CheckInterfaceStatus** stitcher sets the **IsActive** flag to a value of **2** on the out-of-scope interfaces
 - This flag prevents the interfaces from being pinged during monitoring activities

Figure 4-21. Scenario 2: Use scoping solution

Prediscovery filters

- This filter uses field names from the **Details.returns** table
- This table contains the results of queries by the **Details** agent



© Copyright IBM Corporation 2017

Figure 4-22. Prediscovery filters

You can select a predefined discovery filter from the list of Available Filters or create your own custom filter by clicking **Filter Library**. Several preconfigured filters are available after you install Tivoli Network Manager 4.2.

- **Filter Printers:** This option eliminates all devices with Object ID (OID) numbers that correspond to printers.
- **Filter Servers + Workstations:** This option eliminates all devices with OID numbers that correspond to Windows, Linux, Solaris, or UNIX hosts.
- **Filter all end nodes:** This option eliminates all devices included in the preceding two filters.
- **Filter end nodes except SSMs:** This option eliminates all devices with OID numbers that correspond to Windows, Linux, Solaris, or UNIX hosts unless the host is running a Netcool/System Service Monitor (SSM) agent. If SSM agents are loaded only on your servers, this option would eliminate all workstations but include servers in your discovery.
- **Non-IP devices:** This option eliminates any device that does not have an IP address. In previous versions of Tivoli Network Manager, these devices were undiscoverable. However, in version 4.2 discovery collectors can discover devices without IP addresses by gathering information from element management systems (EMS). This filter would eliminate any devices without an IP address that was found with the discovery collectors.
- **SNMP access only:** This option permits only devices to which Tivoli Network Manager had successful SNMP access to be included in the discovery topology. Any devices that did not permit SNMP access is excluded from the topology.

Prediscovery filters use columns in the **Details.returns** table. Filters are typically based on the **sysObjectId** of a device (called **m_ObjectId** in the **Details.returns** table). However, in some cases 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 the **sysObjectId**. However, the IP address of the device can still be used in a prediscovery filter.

A device must pass all filters to be discovered. You can define multiple prediscovery filters. Filters are combined automatically with a Boolean **AND** expression. All criteria that are defined in all filters must be matched.

When you create a prediscovery filter, you can filter based on any of the fields in the **Details.returns** table:

- **m_Name**
- **m_UniqueAddress**
- **m_Protocol**
- **m_ObjectId**
- **m_Description**
- **m_HaveAccess**
- **m_UsdAgent**
- **m_AddressSpace**

In addition, by using the Advanced tab, you can construct filter rows with any of the fields from within the **m_ExraInfo** field.

Tivoli Network Manager also has a post-discovery filter. You can use it to limit discovery to only Cisco devices by using a filter like **snmpSystem > SYSOBJECTID like '1\3\6\1\4\1\9'**. The primary use of the post-discovery filter in previous versions was to eliminate unwanted interfaces from discovery. This use is deprecated by using interface filtering.



Information

The post-discovery filter can be seen on this same tab. However, it is deprecated by using the DNCIM database and by interface filtering. Do not attempt to use the post-discovery filter.

Creating prediscovery filters (GUI)

- Filters in the **Basic** tab must use the **All** (Boolean AND) or **ANY** (Boolean OR) option
 - If you need to combine both Boolean AND and OR statements with parentheses, use the **Advanced** filter view

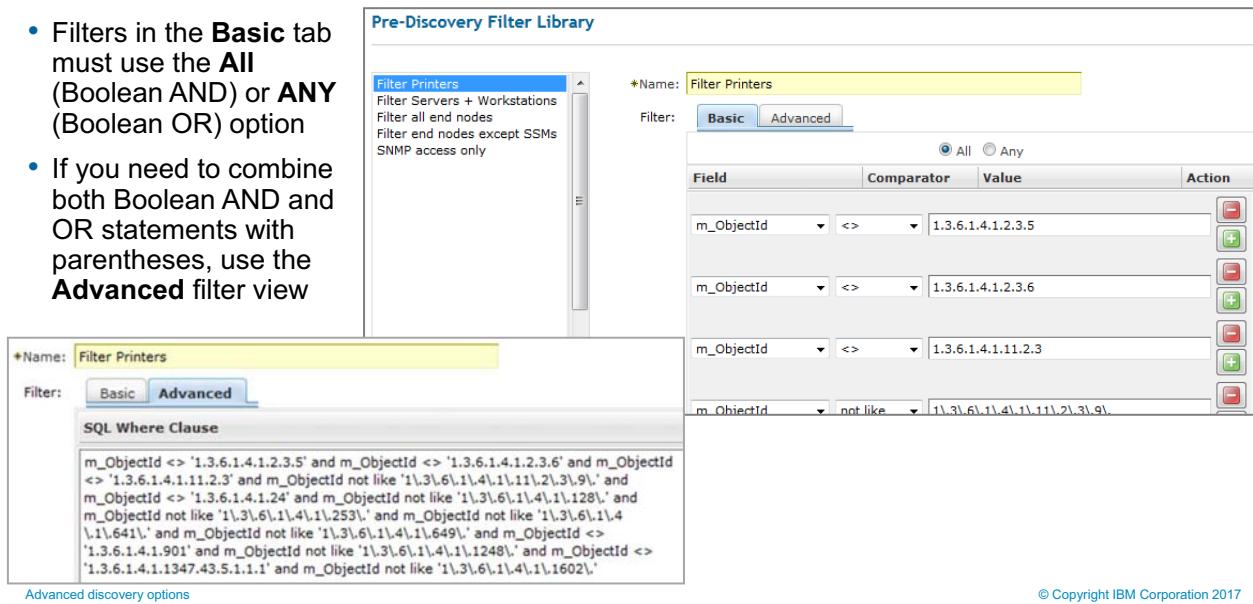


Figure 4-23. Creating prediscovery filters (GUI)

If you click the **Filter Library** button on the Filters discovery configuration tab, you can view the predefined filters or write your own custom filter.

- The **Basic** filter tab combines all criteria with **All** (Boolean **AND** condition) or **Any** (Boolean **OR** condition). If the filter is an **All** filter, a device must meet every condition to pass the filter and be included in the topology. If the filter is an **Any** filter, the device is included in the topology if it matches any of the specified conditions.
 - You can create a more complex filter that uses a combination of Boolean **AND** and **OR** conditions that are grouped in parentheses. To create this type of filter, enter the filter manually in the **Advanced** tab.

Consider the following scenario:

- An enterprise organization has a Windows server farm.
 - This company did not purchase Netcool SSM agents.
 - The network personnel want to exclude all Windows workstations from their discovery but include all the devices in their server farm. Their 200 servers are all on the 192.168.1.0/24 subnet. The problem is that both servers and workstations have a **sysObjectId** of 1.3.6.1.4.2.311. What filter can you create to include servers but exclude workstations in this scenario?

In this case, you can use a filter similar to the following one:

`m_ObjectId` not like `1\..3\..6\..1\..4\..1\..311` OR `m_UniqueAddress` like `192\..168\..1\..`

The preceding filter eliminates all Windows devices from discovery unless they are on the 192.168.1.0/24 subnet.

The filter is like a water pipe. All devices go into one end of the pipe. Only devices that match the specified filter condition come out of the other end of the pipe. Devices that do not match the filter condition are excluded from the discovery.

Interface filtering

- Why use interface filtering?
 - Remove all interface information from the discovery for unwanted types of interfaces such as ISDN interfaces (BRI), local loopbacks, or cable modem interfaces on a Cable Modem Termination System (CMTS)
 - Remove interfaces that can show false connectivity to other devices (**dsc0**, **fpx0**)
 - If virtual devices have many interfaces, discovery can take a long time
 - Speed up discovery by using an interface filter to reduce the number of interfaces that the SNMP helper retrieves
- Interface filtering is more efficient than using a stitcher or a post-discovery filter to eliminate interfaces
 - Interface filtering happens after the agents retrieve data but before the data gets inserted into tables for final-phase stitchers to process

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-24. Interface filtering

The **dsc0** (on Juniper routers) and **fpx0** (on Juniper and Cisco routers) interfaces can report false connectivity information. Eliminating these interfaces can resolve problems where one or two devices in your network appear to be connected to everything else in your network.

- A Juniper **dsc0** interface is called a discard interface. This dsc0 interface is a physical interface that is used to identify the ingress point of a denial-of-service (DoS) attack. When a network is under attack, the target host IP address is identified, and a local policy on the Juniper router forwards attacking packets to the discard interface. This traffic is discarded and not routed to the rest of the network.
- The **fpx0** interface is an internal Ethernet interface that connects a routing engine to a packet-forwarding engine. It is regarded as a management interface.
- When devices with these interfaces are queried for connectivity information, the devices can falsely report that these interfaces are connected to everything in your network.



Information

The post-discovery filter that still appears in the GUI is disabled by default in Tivoli Network Manager 4.2. The post-discovery filter can be used only if you disable the use of the DNCIM database. For this reason, the post-discovery filter is deprecated and this course does not cover its use.

Configuring SNMP interface filters

- You can configure one or more interface filters per device type
- To configure one or more interface filters, complete the following steps:
 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
 4. Give this filter a name
 - Locate the value of **m_FilterName** and give the filter a descriptive name, which is enclosed in double quotation marks
 5. Configure to which devices the filter applies by defining a device filter
 - Locate the value of **m_DeviceFilter** and define a filter, which is enclosed in double quotation marks
 - You can use any Object Identifiers (OIDs) to construct the filter
 - Use Object Query Language (OQL) syntax

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-25. Configuring SNMP interface filters

Interface filter syntax

- The filter must be in the following form:

- mibVariableName expression values [optional_Boolean_operator expression optional_Boolean_operator]

- Examples

```
// Apply the interface filter to only a specific type of device  
sysObjectID = '1.3.6.1.4.1.4874.2.1.1.3'
```

```
// More complex example of the preceding filter  
sysObjectID = '1.3.6.1.4.1.4874.2.1.1.3' 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 devices.  
sysObjectID != ''
```

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-26. Interface filter syntax

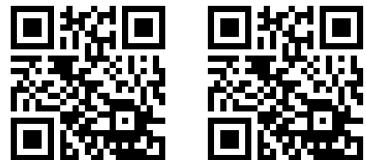
When to use SNMP interface filters

- You can use SNMP interface filters on any SNMP MIB table that is keyed on **ifIndex**
 - For example, filtering on **ifTable** or **ifXTable** allows filtering on values such as **ifType** and **ifDescr**
- Filtering on any SNMP MIB variable other than interfaces is not supported
 - However, you can block access to any table by using the **m_InstanceFilterTable** option

Figure 4-27. When to use SNMP interface filters

How 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 an unfiltered set of queries returns
 - It sends this data 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
- You can also define dependent filters
- For more information, see the chapter entitled *Configuring Network Discovery* in the *IBM Tivoli Network Manager IP Edition: Discovery Guide* information at the following URLs:
 - <http://tinyurl.com/he3Injz>
 - <http://tinyurl.com/hl2kpjb>



[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-28. How interface filtering works

Topic summary

After you complete this topic, you can do these tasks:

- Create a prediscovery filter to eliminate printers from network discovery
- List the two places you can implement an interface filter
- Modify the scope of discovery to exclude individual devices so they are not pinged or interrogated during discovery

Figure 4-29. Topic summary

4.3. Discovering with collectors

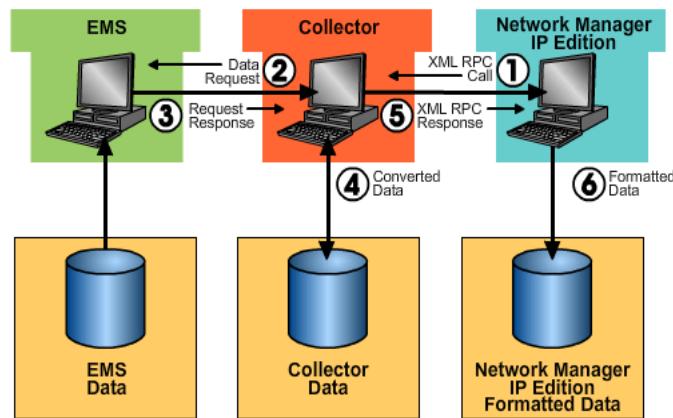
Devices operating at Layers 2 – 3 of the OSI model can be discovered with the normal agents and stitchers. Tivoli Network Manager can also use discovery collectors to get information from element management systems (EMS) or read in information from comma-separated value (CSV) files. Discovery collectors enable Tivoli Network Manager to discover some devices that are used for 2G and 3G networking, and radio access devices.



Figure 4-30. Discovering with collectors

Overview of EMS integration

- The element management system (EMS) integration mechanism makes it possible for Tivoli Network Manager to incorporate data from an external data source into the discovery
 - Collectors** gather entity information from external sources



Advanced discovery options

© Copyright IBM Corporation 2017

Figure 4-31. Overview of EMS integration

Definition of a collector

- A collector is a software module that retrieves topology data from a data source
 - A data source can be an element management system (EMS), comma-separated value (CSV) file, or a database
 - It 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 converts the topology data from the EMS format into a standard XML structure that Network Manager can process
 - Since each EMS has a unique data format, a different collector must be developed for each different EMS vendor and model
- The predefined collectors that are provided with Network Manager are written in either Java or Perl
 - However, collectors can be written in any language
 - The language must provide an XML-RPC server that the **ncp_disco** process can query
 - Network Manager includes Java and Perl modules to support the development of collectors in those languages
- Unlike conventional agents, collectors are largely independent of Tivoli Network Manager and do not use ICMP, SNMP, or Telnet
 - One collector is needed for each EMS
 - It returns data for all the devices that the EMS manages

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-32. Definition of a collector

Tivoli Network Manager 4.2 now includes both Perl collectors and Java collectors.

- Network Manager has a set of default Perl collectors. Perl scripts and a plain-text configuration file for each collector are held in a separate directory within the `$NCHOME/precision/collectors/perlCollectors` directory. Experienced users can develop new collectors to enable Network Manager to interact with other EMS. Configuration and executable files for each new collector must be placed in a subdirectory of the `$NCHOME/precision/collectors/perlCollectors` directory.
- Java collectors are included in the `$NCHOME/precision/collectors/javaCollectors` directory.

Default Java collectors

- **Alcatel5529Idm**
- **Alcatel5620Sam**
- **Cisco APIC REST**: Cisco Application Policy Infrastructure Controller EMS
- **CiscoLMS**: Cisco Works LMS EMS
- **CSV**: Processes comma-separated value (CSV) files
- **Huawei CORBA TMF 814**
- **Huawei M2000**
- **MTOSISoap**: Collector for that discovers Element Management Systems that support the MTOSI Soap NBI, for example, the Huawei U2000 iManager EMS
- **NetActCMDump**: Processes 2G, 3G, and LTE RAN data by using the configuration management XML file for the NetAct EMS collector
- **NetViewer**: Collector for the Nokia Solutions and Networks (NSN) NetViewer EMS
- **Tellabs INM8000**

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-33. Default Java collectors

Default Perl collectors

- **Alcatel5529IdmSoap:** Collector for the Alcatel5529IdmSoap EMS
- **Alcatel5620SamSoap:** Collector for the Alcatel 5620 SAM EMS
- **Alcatel5620SamSoap FindToFile:** Collector for the Alcatel 5620 SAM EMS
- **Alcatel5620SamCsv:** Collector for the Alcatel 5620 SAM EMS
 - This collector retrieves EMS topology data from a CSV file generated from the Alcatel 5620 SAM EMS
- **AlcateINR8PLloolsn:** Collector for the Alcatel Lucent 1353 NM and Alcatel Lucent 1354 RM components within the AlcateINR8PL EMS
- **GenericCsv:** This collector reads `.csv` files for input data
 - Tivoli Network Manager has two versions of this collector
 - One is written in Java and the other written in Perl
- **HuaweiU2000ImanagerTL1:** Collector for the HuaweiU2000Imanager EMS
- **HuaweiU2000iManager TL1DumpExport:** Collector for the HuaweiU2000Imanager EMS
 - This collector uses XML files from the EMS
- **OpticalBlackboxXml:** You can add passive Layer 1 entities to the discovered network topology with this collector

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-34. Default Perl collectors

EMS integration components

- Most EMS Integration components are installed as part of the standard core components
- Finder: **ncp_df_collector** and **ncp_df_dbentry**
- Helper: **ncp_dh_xmlrpc**

Key agents	Key stitchers
CollectorDetails.agnt CollectorInventory.agnt CollectorLayer2.agnt CollectorLayer3.agnt CollectorVpn.agnt	SendToCollectors.stch CollectorIPLayer.stch CollectorSwitchLayer.stch CollectorAddressTranslation.stch CollectorDetailsRetProcessing.stch

Figure 4-35. EMS integration components

Configuring an EMS discovery

To configure an EMS discovery, do the following activities in addition to the standard discovery configuration activities:

1. Configure and start the EMS collectors
2. Seed the EMS discovery by seeding the collector finder
3. Enable collector discovery agents

Figure 4-36. Configuring an EMS discovery

How to configure a collector

1. Configure the collector
2. Edit the collector configuration file
3. Specify the port that the collector must listen on for XML-RPC requests from Network Manager
4. Specify the data source for the collector
5. When you use the CSV collector, specify the file name of the CSV file

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-37. How to configure a collector

Part of the collector configuration file specifies the port number on which the collector listens for requests from the collector agents and finders.

```
General =>
{
  Debug => 0,
  Listen => 8081
},
```

EMS integration collector configuration

The configuration file contains basic information:

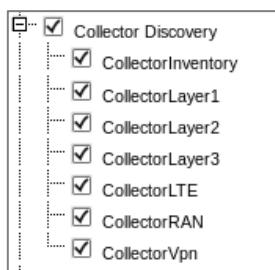
- Name of configuration file
- XML-RPC server port with which Tivoli Network Manager communicates
- **DataSource** is either a comma-separated value (CSV) file or EMS login with host and port information
- The **DataAcquisition** section specifies which Collector agents to use

```
# itnmClassCsvCollector.cfg
{
    General =>
    {
        Debug => 0,
        Listen => 8081,
    },
    DataSource =>
    {
        CsvCfg => 'itnmClassCsv.cfg',
    },
    DataAcquisition =>
    {
        GetEntities => 1,
        GetLayer2Vpns => 1,
        GetLayer3Vpns => 1,
        GetLayer2Connections => 1,
        GetLayer3Connections => 1
    },
}
```

Figure 4-38. EMS integration collector configuration

EMS integration (1 of 2)

1. Manually edit the `DiscoConfig.cfg` file to make sure the `ncp_df_collector` is inserted into the `disco.managedProcesses` table
2. Edit `DiscoCollectorFinderSeeds.cfg` to specify the port value
3. Select Collector Agents with the Discovery Configuration GUI



```
insert into disco.managedProcesses
(
    m_Name
)
values
(
    "ncp_df_collector"
);
```

DiscoConfig.cfg

A screenshot of a Notepad window titled 'DiscoCollectorFinderSeeds.cfg - Notepad'. The window contains a SQL insert statement for the 'collectorFinder.collectorRules' table. The code specifies a port of 8081. The code includes comments explaining the purpose of the file and its creation for different domains.

```
DiscoCollectorFinderSeeds.cfg - Notepad
File Edit Format View Help
// Collector Finder Seed File
// Populates the collector finders rules table to indicate
// which collectors to attempt to contact.
// It is recommended that a different version of this file be
// created for each configured domain. For example if the domain
// is named NCOMS then a file named:
// discoCollectorFinderseeds.NCOMS.cfg
// should be created with the discovery seeds for this domain.
insert into collectorFinder.collectorRules
(
    m_Port
)
values
(
    8081
);
```

© Copyright IBM Corporation 2017

Figure 4-39. EMS integration (1 of 2)

EMS integration (2 of 2)

4. Save the collector configuration file

Host name and port number must be correctly specified in the configuration file

5. Start the collector:

- For Perl collector

```
ncp_perl collector_script -cfg Collector_config_file -listen listeningPort  
-logdir DirectoryName -csvcfg CSV_Collector_Config_File
```

- For Java collector

```
collector.sh -jar [-Xmsminimum_memory-sizem] [-Xmxmaximum_memory-sizem] JARfile  
-propsFile filename -port port_number [-bg]
```

6. Save discovery configuration

7. Stop discovery and restart it

Figure 4-40. EMS integration (2 of 2)

Running EMS integration collectors

Start Collectors before you start the Tivoli Network Manager IP Edition discovery

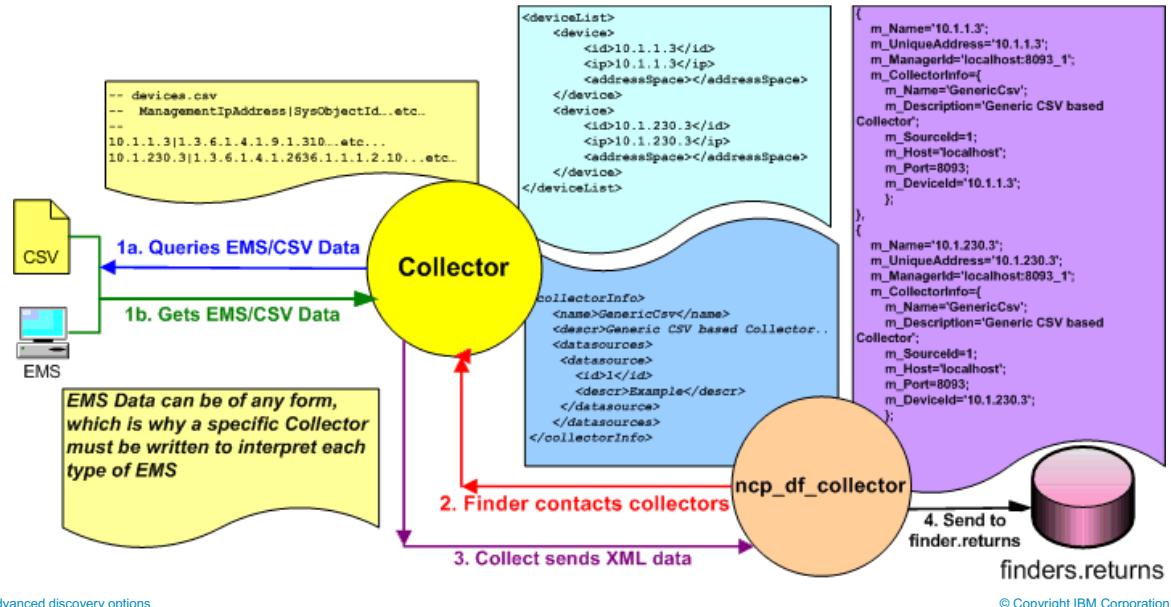
What does a collector do?

- Loads data from the data source (EMS or CSV file)
- Listens for Tivoli Network Manager Remote Procedure Calls (RPC)
 - Sends data on itself and a list of devices that the EMS manages
 - Sends data from CSV file in response to query

```
ncp_perl ./main.pl -cfg itnmClassCsvCollector.cfg &
[1] 27969
[student@localhost GenericCsv]$ Starting collector...
GenericCsv LoadDeviceList() @ 1395192569
GenericCsv LoadInterfaceList() @ 1395192569
GenericCsv LoadEntityList() @ 1395192569
GenericCsv LoadGenericEntityList() @ 1395192569
GenericCsv LoadLayer3Vpns() @ 1395192570
GenericCsv LoadLayer2Vpns() @ 1395192570
GenericCsv LoadMplsInterfaces() @ 1395192570
GenericCsv LoadConnections() - Layer1 @ 1395192570
GenericCsv LoadConnections() - Microwave @ 1395192570
GenericCsv LoadConnections() - Layer2 @ 1395192570
GenericCsv LoadConnections() - Layer3 @ 1395192570
Listening for Precision calls...
```

Figure 4-41. Running EMS integration collectors

Collector finder



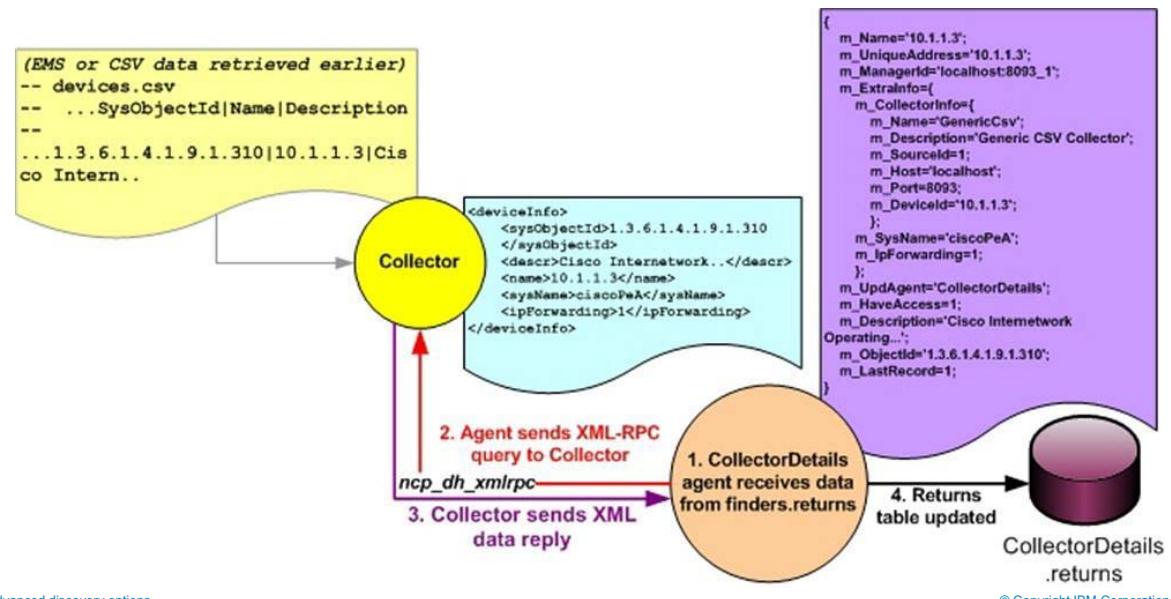
Advanced discovery options

© Copyright IBM Corporation 2017

Figure 4-42. Collector finder

1. The collector runs and gets data from the EMS or CSV file.
2. Collector finder (**ncp_df_collector**) contacts all seeded collectors.
3. The collector responds with information about itself and a list of devices that the EMS manages or that are in the CSV file.
4. Collector Finder sends each device to **finders.returns** for processing.

Collector agents



Advanced discovery options

© Copyright IBM Corporation 2017

Figure 4-43. Collector agents

1. The collector details agent receives information from the **finders.returns** table.
2. The agent sends an XML-RPC query to the collector.
3. The collector sends an XML reply.
4. The collector details agent formats data for storage in the **CollectorDetails.returns** table.

The database finder

Four components can find devices during a discovery

- Ping finder (**ncp_df_ping**)
- File finder (**ncp_df_file**)
- Collector finders (**ncp_df_collector**)
- Database entry finder (**ncp_df_dbentry**): Reads a database to retrieve a list of devices to find on the network

A **dbEntryFinder database** defines the operation of the database finder

- The dbEntryFinder database is defined in the **DiscoDBEntryFinderSchema.cfg** file
- The database has these tables:
 - **dbEntryFinder.configuration**
 - **dbEntryFinder.dbQueries**

Figure 4-44. The database finder

The **DiscoDBEntryFinderQueries.cfg** configuration file specifies a database query to be run against a specified database to retrieve a list of IP addresses of devices to discover on the network. This configuration file can be used to configure inserts into the following database tables:

- **dbEntryFinder.configuration**
- **dbEntryFinder.dbQueries**

The following example insert configures the database finder to use five threads:

```
insert into dbEntryFinder. configuration
( m_NumThreads )
values
(   5   );
```

The following example configures the database finder to use 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'
}
]
);

```

Topic summary

After you complete this topic, you can do the following task

- Configure a discovery to use a CSV file as a source of discovery information

Figure 4-45. Topic summary

4.4. Troubleshooting discovery

This lesson provides troubleshooting tips to resolve common network discovery issues.

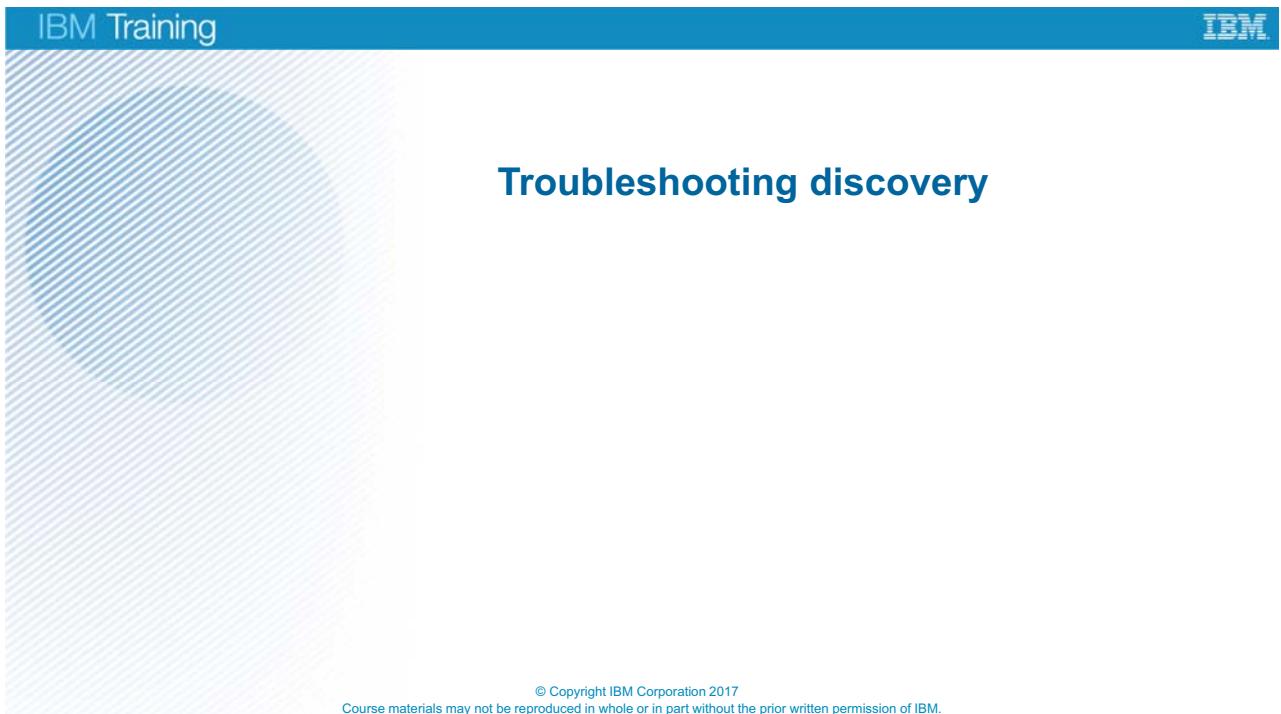


Figure 4-46. Troubleshooting discovery

Discovery takes a long time

Causes

- Large number of entities
- Using agents to gather extra data beyond what is necessary for mapping:
 - **ExtraDetails**
 - **Entity**

Common solutions

- Partition network discovery into smaller separate domains
- Eliminate unnecessary agents
- Reduce finding time by using hosts file for file finder, and eliminate ping sweeping and ping verification
- Increase server network bandwidth and processing power

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-47. Discovery takes a long time

Enable only those agents that you need to discover your network with an appropriate level of detail. Choosing the full Layer 2 and 3 discovery agents is adequate for most discoveries. After you run several discoveries, use the script or view the report that shows what agents were used during discovery. Disable any agents that were not used during the previous discoveries.

The **Entity** agent gathers detailed information such as component serial numbers. This information is useful for inventory purposes once or twice a year, but it is unlikely that you need it in every discovery. Since this agent adds significantly to discovery time, disable it for normal discoveries.

The **ExtraDetails** agent gathers some useful information such as **sysContact** and **sysLocation**. Some of the other information that this agent gathers is less useful. Edit the `$ITNMHOME/disco/agents/ExtraDetails.agnt` file and comment out those queries that you do not want to run. Editing agent files to remove unnecessary queries can reduce the time that is taken to interrogate network devices.

After you run the first discovery, run a script to generate a seed file that contains all previously discovered hosts. Enable the file finder to read this file. Remove seeds from the ping finder but leave it enabled. These steps drastically reduce Phase 1 discovery time.

If your network is large, it can help to split the network into separate domains. If you have more than 350,000 entities in a domain, consider splitting the discovery into smaller domains to reduce overall discovery time.

Reducing finding time is the key area of making discovery more efficient. Run the `$ITNMHOME/scripts/perl/scripts/BuildSeedList.pl` scripts periodically and have the file finder read the resulting `$INCHOME/etc/precision/seedlist` file to reduce stage 1 discovery time.

Discovery does not start

- Check `$NCHOME/log/precision` directory for errors in discovery log files
- If `ncp_disco` process is not starting, review recent changes to configuration files, stitchers, and agent files
- If `ncp_disco` is running, but it remains in Phase 0, a seeded device is not in scope

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-48. Discovery does not start

If a Tivoli Network Manager process does not start, follow these suggested steps:

1. Check the existing log files in the `$NCHOME/log/precision` directory for the process that is not starting. Focus on the `*.trace` file if one is available. Look through the file for the words **fail** or **error**.
2. If `itnm_status` shows that a particular process failed, it means that the CTRL service already tried five times to start it.
3. Open two terminal windows on the server.
4. In one window, start the process manually. For example, to start the MODEL service, type:
`ncp_model -domain DOMAIN_NAME &`
5. In the second window, watch the resulting trace file from the process. For example, type:
`tail -f $NCHOME/log/precision/ncp_model.DOMAIN_NAME.trace`
6. Observe both windows for messages that indicate the source of the failure. For example, `ncp_model` cannot start if it cannot establish a connection to the NCIM database.
7. Resolve the problem.
8. As soon as you can manually start the process (see step 4), stop all Tivoli Network Manager processes and then restart with the following commands:

```
itnm_stop ncp
pkill ncp
itnm_start ncp
```

Other discovery issues

- If the discovery status GUI shows a status of **Server busy**, wait for the screen to refresh
- Tivoli Network Manager identifies management interfaces as the root cause of outages
 - Add the management interface IP addresses to the **scope.special** table
 - Interfaces that are listed in the **scope.special** table are not considered in root cause analysis (since these interfaces do not route traffic)
 - Run a new discovery
- Discovery runs out of memory
 - Previous 32-bit versions of Tivoli Network Manager in large MPLS networks often gathered data that exceeded the limits of 32-bit memory
 - Version 4.2 is 64-bit software, so any memory limitations are related to hardware rather than software
 - Increase system memory or split discovery into smaller domains

Figure 4-49. Other discovery issues

If the discovery status GUI hangs for a long time, try the following steps:

1. Close the discovery status tab and reopen. Check to see whether the problem is resolved.
2. Log out of Tivoli Integrated Portal. Close the browser. Clear the cache. Reopen the browser. Log on to Tivoli Integrated Portal. Check to see whether the problem is resolved.
3. Restart Tivoli Integrated Portal and Tivoli Network Manager. Check to see whether the problem is resolved.
4. If the problem is still not resolved, check the log files for **ncp_config** and **ncp_disco** for errors.
5. In large MPLS networks, it is possible to have enough entities to cause Tivoli Network Manager discovery to reach the limit of 32-bit memory in versions before 4.2. In this case, take one of the following actions.
 - Split the discovery into separate domains
 - Upgrade to Tivoli Network Manager 4.2, which is 64-bit software. This upgrade eliminates the 32-bit memory limitation for discovery.

Uses for discovery table caching

- Submit discovery cache files when you open a support ticket
 - These files enable support personnel to replicate your discovery information and results
 - Submit a `.tar` file of the `$NCHOME/var/precision` directory with your trouble ticket
- When you develop custom stitchers or agents, run discovery in failover mode
 - Move cache files from production environment data to a development environment
 - Write and test custom stitcher modifications in the development environment
 - When the custom stitchers or visualizations are successful, implement the modifications in the production environment

Figure 4-50. Uses for discovery table caching

Enable caching of discovery tables

- To enable this advanced option, do the following steps:
 - Select the option under **Advanced Discovery Configuration**
 - Save discovery configuration
 - Restart discovery
- A rediscovery requires information from the previous discovery
 - If the discovery process stops, you cannot run a rediscovery until another full discovery occurs
- Enabling the discovery cache keeps a persistent copy of discovery information
 - Even if you stop the discovery process, you can run a rediscovery at any time if at least one full discovery process took place with caching enabled
 - However, enabling the discovery cache slows discovery

Figure 4-51. Enable caching of discovery tables

Selecting this option adds to discovery time by introducing latency that is necessary to write to the hard disk drive. For optimal performance, do not leave this option enabled in a production environment. After a full discovery runs, leave the discovery engine running so that you can run a partial discovery when necessary.

The discovery cache files

- Discovery cache files are stored in the following location:
 - `$NCHOME/var/precision` directory
 - Contains all information that is needed for network support to reconstruct network topology
- Load these cache files into another server to view the topology

Figure 4-52. The discovery cache files

Discovery agent and stitcher data

- Cache files for agents and stitchers can help support personnel reconstruct what happened in the discovery process
- To gather discovery data, complete the following steps:
 1. Change the debug level of the `*.trace` file for **ncp_disco** to 3 or 4 (with `kill -USR2 Process_Id`)
 2. Run a full discovery
 3. Gather discovery data with one of these two methods:
 - Option 1: Configure **Advanced** tab to run discovery in caching mode. Run discovery. Compress files in `$NCHOME/var/precision` and send to support. Enabling caching of discovery data adds to discovery time.
 - Option 2: A better way is to run a full discovery and then run the `$ITNMHOME/scripts/perl/scripts/GetDiscoCache.pl` script. Send the resulting `.tar` file to support.

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-53. Discovery agent and stitcher data

In most cases, setting the debug level of the trace file to 3 is adequate to diagnose problems with Tivoli Network Manager processes.

Data to send to technical support

- Create *.tar files for each of the following directories:
 - **\$NCHOME/var/precision:** A .tar file of this information is created with the \$ITNMHOME/scripts/perl/scripts/GetDiscoCache.pl script
`ncp_perl $ITNMHOME/scripts/perl/scripts/GetDiscoCache.pl -domain DOMAIN_NAME`
 - **\$NCHOME/log/precision:** Contains log files
 - **\$NCHOME/etc/precision:** Contains configuration files
- Compress files and attach to support ticket
 - Providing all this information when you open a support ticket reduces the amount of time necessary to receive a problem resolution

Figure 4-54. Data to send to technical support

Useful commands (1 of 2)

- Check Tivoli Network Manager status
 - `itnm_status`
- Start or stop all processes:
 - `itnm_start`
 - `itnm_stop`
- Use a command-line argument to start or stop individual products:
 - `tip`: Tivoli Integrated Portal (not used with IBM Tivoli Network Manager 4.2 under Netcool Operations Insight)
 - `ncp`: Tivoli Network Manager
- Examples
 - `itnm_start tip`: Starts Tivoli Integrated Portal
 - `itnm_stop ncp`: Stops Tivoli Network Manager but leaves OMNIbus and Tivoli Integrated Portal running

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-55. Useful commands (1 of 2)

In previous versions of Tivoli Network Manager, OMNIbus could be stopped with the `itnm_stop nco` command. With version 4.2, OMNIbus usually runs under its own process control.

Useful commands (2 of 2)

- Display routing information on the server with the `netstat -rn` command

```
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
192.168.1.0    0.0.0.0        255.255.255.0 U        0 0          0 eth1
172.30.0.0     10.191.101.1  255.255.0.0   UG      0 0          0 eth0
172.31.0.0     10.191.101.1  255.255.0.0   UG      0 0          0 eth0
10.191.0.0     0.0.0.0        255.255.0.0   U        0 0          0 eth0
169.254.0.0    0.0.0.0        255.255.0.0   U        0 0          0 eth0
127.0.0.0      0.0.0.0        255.0.0.0    U        0 0          0 lo
224.0.0.0      0.0.0.0        240.0.0.0    U        0 0          0 eth0
0.0.0.0         192.168.1.1   0.0.0.0      UG      0 0          0 eth1
```

- Enable multicasting on Linux or UNIX server as the **root** user:

```
/sbin/route add -net 224.0.0.0 netmask 240.0.0.0 dev eth0
```

- Without multicast enabled, some Tivoli Network Manager components cannot communicate, resulting in socket errors similar to the following message

```
Tue Jan 24 17:57:37 2012 Info: Attempt to set a socket option failed.
found in file CRivObjMulticast.cdd at line 350
```

Figure 4-56. Useful commands (2 of 2)

Multicasting must be enabled on the Tivoli Network Manager server in order for its components to communicate properly. Multicast networks use the 224.0.0.0/4 subnet (all the addresses from 224.0.0.0 – 239.255.255.255). The available addresses within this range are referred to as Class D addresses. In the preceding example, the 224.0.0.0 subnet with the 240.0.0.0 netmask shows that this server is running multicast networking.

When multicast is not enabled on the Tivoli Network Manager server, some processes fail to start, and the log file for those processes contains a **socket option failed** message. The error message includes a reference to **CRivObjMulticast**.

- Symptom:** Components such as the **ncp_d_helpserv** fail with the following error in the trace files:

```
Info : Attempt to set a socket option failed. found in file
CRivObjMulticast.cc at line 545
I nfo : Attempt to set a socket option failed. found in file
CRivObjMulticast.cc at line 350
I nfo : Attempt to set a socket option failed. found in file CRivSockEngine.cc
at line 139
```

- Cause:** Either multicasting is not enabled or the first available interface on the device is disabled.
- Diagnosing the problem:** Use `ifconfig -a` to list the status of the interfaces on the server. If the first interface is not **UP**, then Tivoli Network Manager stores this interface IP address in the `ServiceData.cfg` file, which causes components that attempt to connect to the IP address to fail. In the following example `ifconfig`, you can see that `bge0` is not enabled but `bge1` is enabled.

```
# ifconfig -a
lo0 : flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
      inet 127.0.0.1 netmask ff000000
bge0 : flags=1000843<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
      inet 192.168.62.52 netmask ffffff00 broadcast 192.168.62.255
      ether 0:14:4f:61: da: 5e
bge1 : flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500
index 3
      inet 192.168.62.99 netmask ffffff00 broadcast 192.168.62.255
      ether 0:14:4f:61: da: 5f
```



Note

The **bge0** interface has no **UP** indicator within the arrow brackets (< >).

In this example, the bge0 interface was not enabled, or someone administratively disabled the interface with the following command:

```
ifconfig bge0 down
```

The `ServiceData.cfg` on this example server shows that the Tivoli Network Manager services are using the 192.168.62.52 address on the bge0 interface that is not operational:

```
bash-2.05# cat ServiceData.cfg
-
- 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: ncp_config DOMAIN: NONROOT_TNM ADDRESS: 192.168.62.52 PORT: 7968
SERVERNAME: bge0 DYNAMIC: NO
```

Solution:

1. Verify that multicasting is enabled by viewing the output of the `netstat -rn` command as seen in the preceding slide graphic.
2. Modify the `ServiceData.cfg` file to connect Tivoli Network Manager processes to the first enabled interface listed by the `ifconfig` command.

Restarting discovery with an empty database

1. Stop all **ncp** processes

```
itnm_stop ncp
```

2. Remove all the cache files

```
rm -f $NCHOME/var/precision/*DOMAIN_NAME
```

3. Remove agent data from the embedded **sqlite** database

```
rm -f $ITNMHOME/embeddedDb/sqlite/ncp_disco.DOMAIN_NAME
```

4. Restart **ncp** processes

```
itnm_start ncp
```

- The embedded sqlite database stores discovery agent data for processing by discovery stitchers
- The field names in the sqlite database have the same names as the NCIM database

*For versions before 4.1, use only steps 1, 2, and 4

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-57. Restarting discovery with an empty database

The embedded sqlite database exists for several reasons.

- The database uses the same table and column names as the Network Connectivity and Inventory Model (NCIM) database. The NCIM database is the database from which topology maps are drawn. By using identical table and column names, the development of custom discovery stitchers is made easier.
- The sqlite database is fast and results in decreased discovery times.

Tivoli Network Manager 4.2 uses the same discovery mechanisms as version 3.9 in phases 1 – 3 of the discovery process. However, stage 4 is different.

- In version 3.9, discovery created a flat table called **scratchTopology.entityByName**. A **PostScratchProcessing** stitcher called modular custom stitchers to make custom modifications to this table. Then, the **SendTopologyToModel** stitcher wrote the topology to the MODEL service. Stopping the **ncp_disco** process deleted the memory-resident copy of the **scratchTopology.entityByName** table and all agent data. A partial discovery cannot be run after the data from the agents, and the **scratchTopology.entityByName** table is deleted. The only way to run a partial discovery was to first run a full discovery or enable the option on the Advanced tab of the Discovery Configuration GUI to cache discovery data. However, enabling this option resulted in slower discovery times.
- In Tivoli Network Manager 4.x, the agent data builds a **workingEntities.finalEntity** table. Then, a **PopulateDNCIM** stitcher runs and populates an embedded sqlite database. This database is persistent even when you stop and restart Tivoli Network Manager. Using the sqlite database is fast and results in faster stitching in the final phase of discovery.

Topic summary

Having completed this topic, you can do the following tasks:

- Start and stop discovery
- Remove all previous discovery data and start discovery with an empty database
- Troubleshoot a slow discovery
- Locate the proper files to diagnose why discovery is not starting
- Collect the appropriate data to send to technical support when you open a trouble ticket for Tivoli Network Manager

Figure 4-58. Topic summary

4.5. Optimizing discovery

This lesson gives tips on how to reduce discovery time.

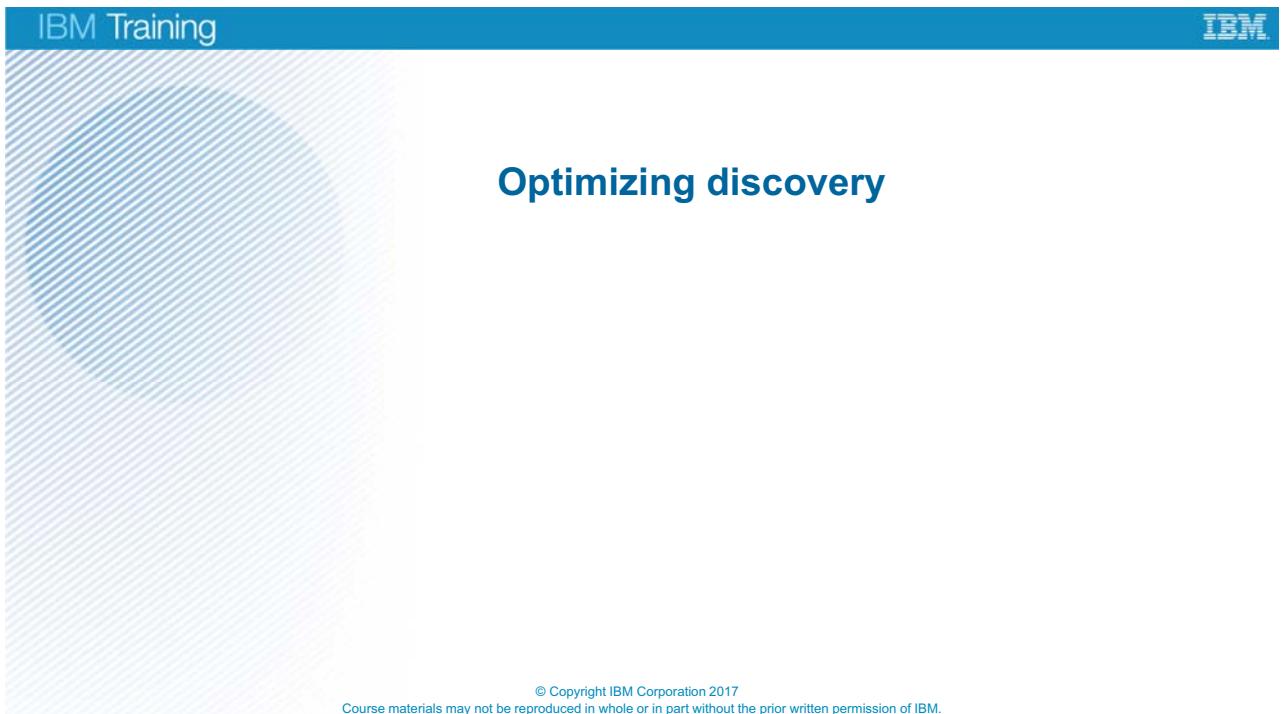


Figure 4-59. Optimizing discovery

IBM Training IBM

Tweaking discovery

The screenshot shows the 'Tweaking discovery' section of the configuration interface. It includes five tabs: Advanced Finder Configuration, Advanced Telnet Helper Configuration, Advanced Ping Finder Configuration, Advanced SNMP Helper Configuration, and Advanced Discovery Configuration.

- Advanced Finder Configuration:** Sets Concurrent File Finders to 5.
- Advanced Telnet Helper Configuration:** Sets Concurrent Telnet Helpers to 10, Default Timeout to 5000 ms, and Number of Retries to 1.
- Advanced Ping Finder Configuration:** Sets Concurrent Ping Finders to 30, Default Timeout to 250 ms, Default Number of Retries to 1, Inter-Ping Time to 100 ms, Allow Broadcast Pinging to checked, and Allow Multicast Pinging to checked.
- Advanced SNMP Helper Configuration:** Sets Concurrent SNMP Helpers to 300, Timeout to 3000 ms, Number of Retries to 1, GetNext Slowdown to 20 ms, and GetNext Boundary to 5000.
- Advanced Discovery Configuration:** Includes settings for Feedback Control (Feedback), Ping Verification (Detect best setting), and Virtual Naming (Allow Virtual). It also lists several naming and inference options with checkboxes:
 - Enable VLAN Modeling:
 - Enable SysName Naming:
 - Enable Caching of Discovery Tables:
 - Enable File Finder Verification:
 - Enable Inference of Dumb Hubs:
 - Enable Rediscovery Rebuild Layers:
 - Enable Rediscovery of Related Devices:
 - Enable iName/iDesc Interface Naming:
 - Enable Inference of PEs using BGP data on CEs:
 - Enable Inference of MPLS CE routers on /30 subnets:

A callout box highlights the following points:

- You can adjust discovery to better fit the capabilities of your server by changing values on the **Advanced** tab of the discovery configuration GUI
- Change values with care and be aware of hardware capabilities
 - Changes can affect the performance of Tivoli Network Manager

Advanced discovery options © Copyright IBM Corporation 2017

Figure 4-60. Tweaking discovery

Reducing Phase 1 discovery time with seed file

- Run a script to generate a list of discovered devices in `$NCHOME/etc/precision/seedfile.txt`:

```
cd $ITNMHOME/scripts/perl/scripts/  
ncp_perl ./BuildSeedList.pl -domain DOMAIN_NAME
```

- Enable the file finder to use `seedfile.txt`
 - If your network is dynamic, consider configuring file finder verification to verify entries in `seedfile.txt`
- Remove host and subnet entries from the ping finder configuration, but leave the ping finder enabled
 - Disable ping finder when you want to limit discovery to only those devices in the seed file
 - Enable ping finder and feedback to find devices that are in scope but not in the seed file
- Run discovery

Figure 4-61. Reducing Phase 1 discovery time with seed file

Discovery tips (1 of 5)

General discovery

- Develop a written plan that specifies the parts of your network and the types of devices that you want to discover
- Use a phased approach for the initial discoveries
 - Discover the core network elements first
 - Then, add key edge networks

Scope

- If Class A or Class B subnet is sparsely populated, include its individual routers in the ping or file finder seedlist
 - Include the key routers on these subnets in the ping finder seedlist, but do not ping sweep entire subnets
 - Enable feedback on full discovery
 - These steps reduce Phase 1 discovery time
- Do not run discovery in debug mode or enable discovery caching unless advised to do so by IBM technical support
- Eliminate Integrated Services Digital Network (ISDN), Basic Rate Interfaces (BRI), and other dynamic dial backup interfaces from the scope of discovery to prevent circuit activation

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-62. Discovery tips (1 of 5)

Discovery tips (2 of 5)

Seedlist

- Populate the seedlist with the IP addresses of high-traffic routers
 - Use feedback instead of subnet ping sweeping to discover attached subnets
- If you need to restrict discovery to the devices in the seedlist, disable feedback in the Advanced tab
- Break large subnets into smaller ones for ping sweeping to reduce Phase 1 discovery time
 - IPv6 networks must have a mask of 112 bits or greater

Figure 4-63. Discovery tips (2 of 5)

Discovery tips (3 of 5)

- **Discovery agents**

- Use only those agents that are necessary to meet discovery requirements
- Use agent filters to eliminate unwanted interfaces
- Increase the number of threads for heavily used processes such as **ncp_disco**, the SNMP helper, and the ping finder
- Determine which agents were unused in a discovery and then disable those agents for subsequent discoveries
- When you use the **TraceRoute** agent to discover networks through a firewall, include the subnet on the other side of the firewall as a discovery seed entry
- On versions before 4.2, enable the **IPv4/6 InetRouting** agent if you have IPv6 addressing in your network

Figure 4-64. Discovery tips (3 of 5)

Discovery tips (4 of 5)

Filters

- Use prediscovery filters to eliminate all end nodes, printers, and other devices that you do not want in the final discovery
- Use filters to prevent querying of sensitive devices where polling a device constitutes a security risk or causes the device to overload

Discovery tuning

- In reliable networks, reduce the discovery helper timeouts and number of retries to increase discovery speed
- Enable broadcast and multicast pinging to reduce Phase 1 discovery time
 - Disable broadcast and multicast pinging to reduce the amount of network traffic that the discovery process generates
- Generate a hosts file after the initial discovery and use in subsequent discoveries to reduce Phase 1 discovery time
 - Eliminate entries in the ping finder seedlist when you use a file finder to read a generated hosts list
 - Leave the ping finder enabled so that devices recently added to the network are discovered

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-65. Discovery tips (4 of 5)

Discovery tips (5 of 5)

Partial rediscoveries

- You can run a partial discovery only if the **ncp_disco** process continues to run after the full discovery
- If you stop a running discovery, you must then do a full discovery before you are able to do a partial discovery
- Enable the ping finder on the **Seeds** tab (even without specified seeds) to facilitate finding changes on surrounding devices

Domain Name Service (DNS)

- Configure discovery to use hosts files first if available
- Configure discovery to use a backup domain name server to resolve host names when they are not available in the file finder
 - Using a backup DNS server minimizes DNS response-time impact to normal users who are on the primary server
- Configure discovery to use the **sysName** of a device as its host name
 - This option is on the Advanced tab

Disable VLAN modeling to reduce discovery time when it is not needed for network fault isolation

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-66. Discovery tips (5 of 5)

Running a partial discovery requires information from the last full discovery.

- **Partial discovery in Tivoli Network Manager 3.x versions**

- In previous versions of Tivoli Network Manager, all information from the last discovery is discarded when you stop the **ncp_disco** process. This process can be stopped from the command line or by clicking the red stop sign icon in the Discovery Status window. After **ncp_disco** stops, you must run a new full discovery and leave the **ncp_disco** running. After you complete the full discovery, you can run a partial discovery if the **ncp_disco** process is not stopped.

- **Partial discovery in Tivoli Network Manager 4.x**

- In version 4.x of Tivoli Network Manager, the embedded sqlite DNCIM database maintains a persistent store of discovery processing information in an optimized format.
- Any time after you run at least one full discovery, you can run a partial discovery even if you stop the **ncp_disco** process. When you run a partial discovery, Tivoli Network Manager reads information about the last discovery from the DNCIM database.

The DNCIM database partially negates the need to select the **Enable Caching of Discovery Tables** option on the Advanced tab of the Discovery Configuration GUI. Selecting this option adds to discovery time by introducing latency because of writing frequently to the hard disk. If you need to send discovery data to support, use the `$ITNMHOME/scripts/perl/scripts/GetDiscoCache.pl` script to create a `.tar` file of discovery data instead of selecting the caching option in the GUI.

**DANGER**

If you enable the caching of discovery tables for a large network discovery, it can significantly increase the amount of time it takes to complete discovery.

If the **ncp_disco** process is running, the `$ITNMHOME/scripts/perl/scripts/GetDiscoCache.pl` script can create on-demand cache files of agent data. The DNCIM database allows partial discoveries to be run at any time. The only reason to enable the caching of discovery tables is if the **ncp_disco** process terminated unexpectedly in previous discoveries. In this case, the cache files might provide diagnostic information to determine the cause of the abnormal termination. The **GetDiscoCache.pl** script and other Perl scripts are highlighted in another unit. The DNCIM database in 4.2 represents a major improvement in the usability of partial discoveries over previous versions of Tivoli Network Manager.

Poor ping sweep seeding

- Customer has 5000 devices on a 192.168.0.0 / 16 subnet to be found with ping sweeping
- Ping sweep the 192.168.0.0 / 16 subnet
 - Threads = 1
 - Addresses to ping = $256 * 256 = 65536$
 - Pings per second = 10
 - Number of retries = 1
- Time taken (seconds)
 - $(\text{Addresses to ping} / \text{Number of threads} / \text{Pings per second}) + ((\text{Addresses to ping} - \text{Number of Devices}) * \text{Number of retries} / \text{Number of threads} / \text{Pings per second})$
 - $(65536 / 1 / 10) + (65536 - 5000) * 1 / 1 / 10 = 12607 (\sim 3.5 \text{ hours})$

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-67. Poor ping sweep seeding

Efficient ping sweep seeding

- Customer has 5000 devices on a 192.168.0.0 / 16 subnet to be found with ping sweeping
- Ping sweep eight /19 subnets
 - Threads = 8
 - Addresses to ping = $256 * 256 = 65536$
 - Pings per second = 10
 - Number of retries = 1
- Time taken (seconds) =

$$(65536 / 8 / 10) + (65536 - 5000) * 1 / 8 / 10 = 1576 (\sim 26 \text{ minutes})$$

Address		Address Block Mask
192.168.0.0		255.255.0.0 (/16)
Address Block Range		
192.168.0.0 - 192.168.255.255		
CIDR Bits	Max Routes	CIDR Mask
3	8	255.255.224.0 (/19)
Routes/Address Allocations		
Route	Address Range	
0 192.168.0.0	192.168.0.0 - 192.168.31.255	
1 192.168.32.0	192.168.32.0 - 192.168.63.255	
2 192.168.64.0	192.168.64.0 - 192.168.95.255	
3 192.168.96.0	192.168.96.0 - 192.168.127.255	
4 192.168.128.0	192.168.128.0 - 192.168.159.255	
5 192.168.160.0	192.168.160.0 - 192.168.191.255	
6 192.168.192.0	192.168.192.0 - 192.168.223.255	
7 192.168.224.0	192.168.224.0 - 192.168.255.255	

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-68. Efficient ping sweep seeding

Exercise: Advanced discovery options

Advanced discovery options

© Copyright IBM Corporation
2017

Figure 4-69. Exercise: Advanced discovery options

Complete the exercise for this unit.

Review questions

1. What two things must you remove to restart Tivoli Network Manager with an empty topology database?
2. What filter do you use to eliminate printers from your network discovery?
 - A. An interface filter
 - B. A prediscovery filter
 - C. A post-discovery instantiation filter
3. What must you start before you run a network discovery that uses collectors?
4. What is the best way to ensure that you do not activate ISDN interfaces during a network discovery?

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-70. Review questions

Write your answers here:

Unit summary

- Generate a seed file from a previous discovery and use it to reduce finding time
- Properly configure ping sweeping for a class B subnet
- Configure discovery of multicast networks
- Gather data for technical support
- Optimize discovery to reduce Phase 1 discovery time
- Restart discovery with an empty topology database

[Advanced discovery options](#)

© Copyright IBM Corporation 2017

Figure 4-71. Unit summary

Review answers

1. To restart Tivoli Network Manager with an empty topology database, remove the `$NCHOME/var/precision` and `$ITNMHOME/embeddedDb/sqlite/ncp_disco.DOMAIN_NAME` directories.
2. To eliminate printers from your network discovery, use a prediscovery filter.
 - a. An interface filter
 - b. A prediscovery filter
 - c. A post-discovery instantiation filter
3. Start the **collector** before you start the network discovery.
4. To avoid activating ISDN interfaces during a network discovery, exclude their IP addresses from the discovery scope.

Figure 4-72. Review answers

Unit 5. Topology visualization basics

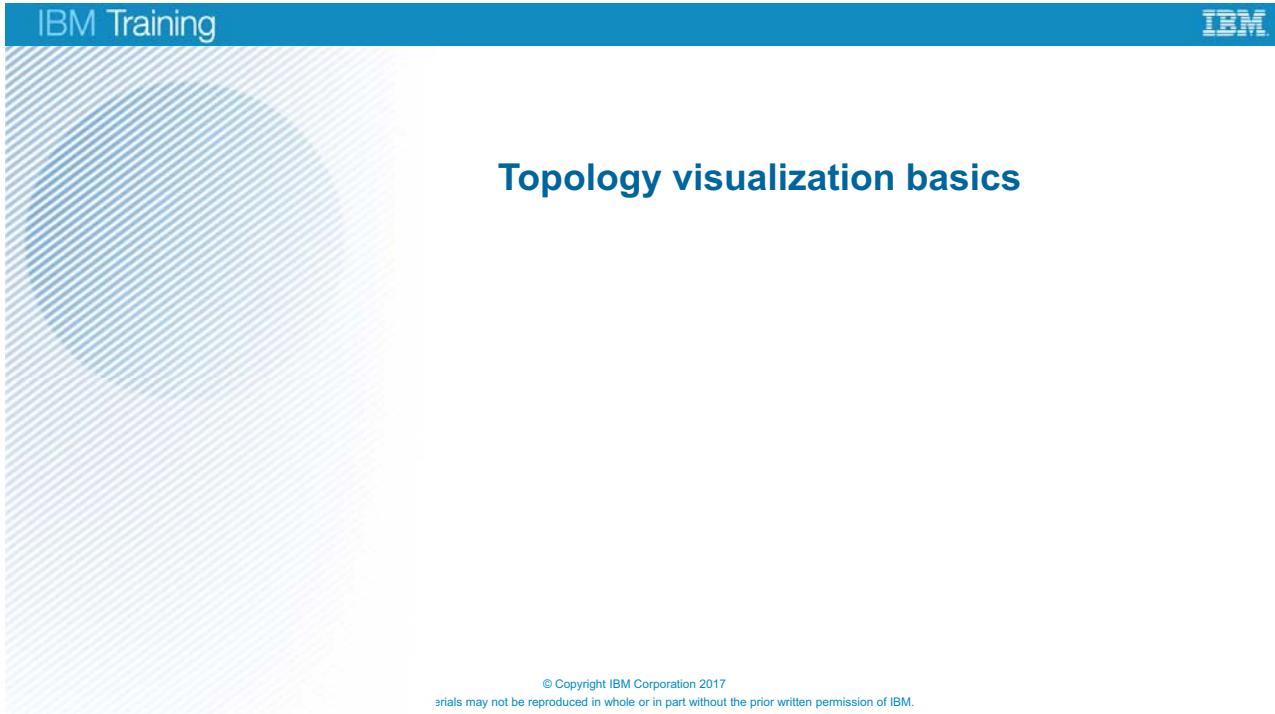


Figure 5-1. Topology visualization basics

Estimated time

01:00

Overview

In this unit, students learn to use the Hop View, the Network View, bookmark views, and the in-context tools associated with each view.

How you will check your progress

- Answer checkpoint questions
- Complete student exercises

Unit objectives

- Use the Hop View
- Use Network Views
- Search for network devices
- Display link status and indicate bandwidth
- Create Dynamic Network Partition views
- Create an Internet Protocol (IP) filtered view

Figure 5-2. Unit objectives

5.1. Visualization architecture

This topic explains the data sources for visualization in Tivoli Network Manager.

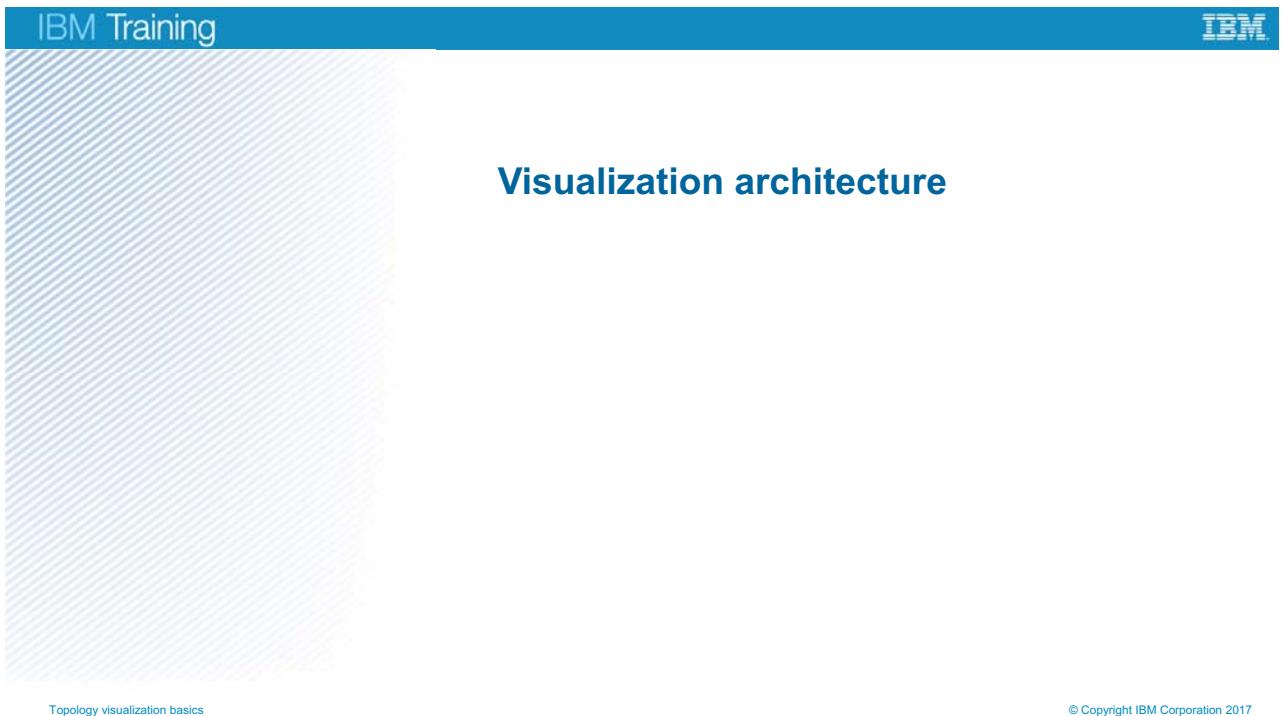


Figure 5-3. Visualization architecture

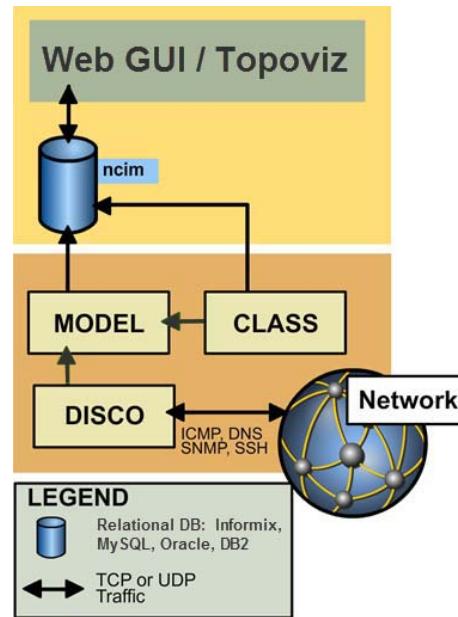
Dashboard Application Services Hub

- The Dashboard Application Services Hub (DASH) provides a GUI interface that eliminates the need for Java runtime engines and integrates several products into a common GUI:
 - OMNIBUS
 - Tivoli Netcool Configuration Manager
 - Tivoli Netcool/Impact
 - Tivoli Network Manager
 - Other Netcool products
 - Custom dashboards
- Facilitates single sign-on

Figure 5-4. Dashboard Application Services Hub

Topoviz architecture

- The WebGUI shows events from the ObjectServer
 - Topoviz is a component that shows network topology
 - Both of these components integrate into DASH
- Tivoli Network Manager discovers the network and creates a topology in the MODEL service
- MODEL gets classification information from the CLASS service and categorizes devices
- MODEL pushes discovery into the Network Connectivity and Information Model (NCIM) database
- Topoviz reads the NCIM database to derive topology views



[Topology visualization basics](#)

© Copyright IBM Corporation 2017

Figure 5-5. Topoviz architecture

- Web GUI** is the component in DASH that shows event views.
- Topoviz** is the component of Tivoli Integrated Netcool Operations Insight dashboard that accesses topology data in the Network Connectivity and Inventory Model (NCIM) database and presents it in graphical format.

5.2. Event views

The Web GUI portion of the Netcool Operations Insight dashboard shows event data that is stored in the OMNIbus ObjectServer.

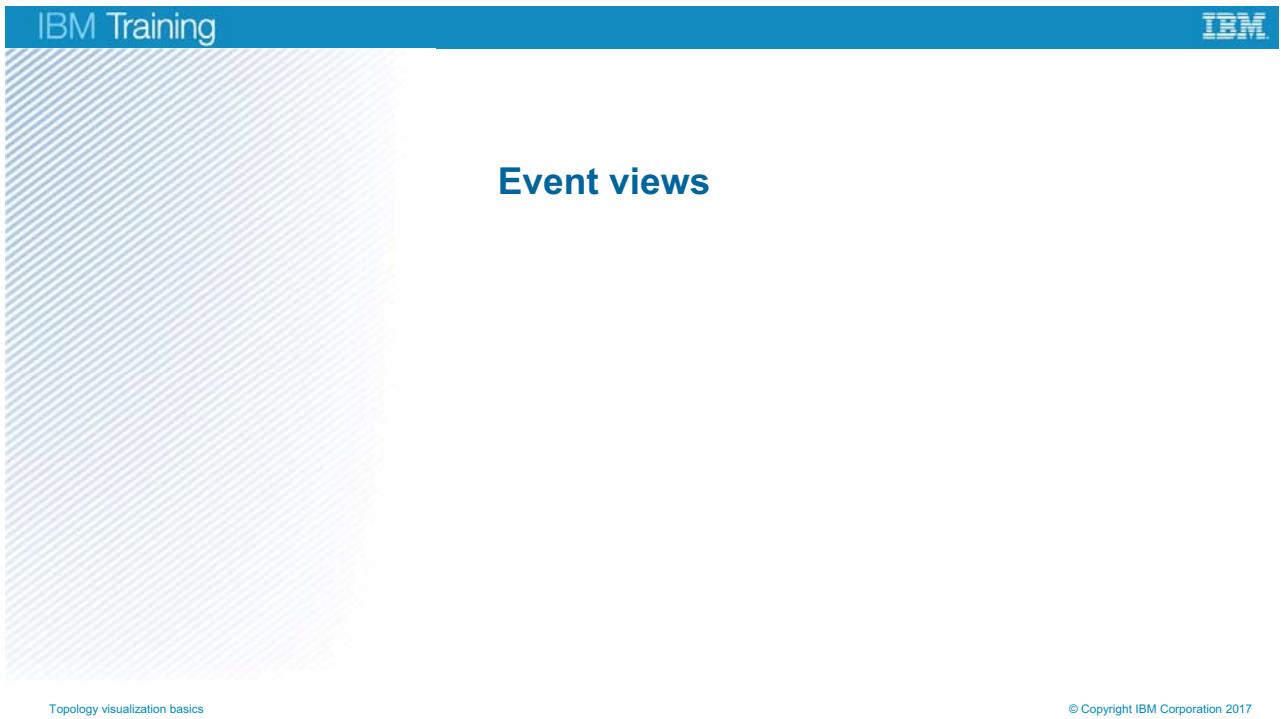


Figure 5-6. Event views

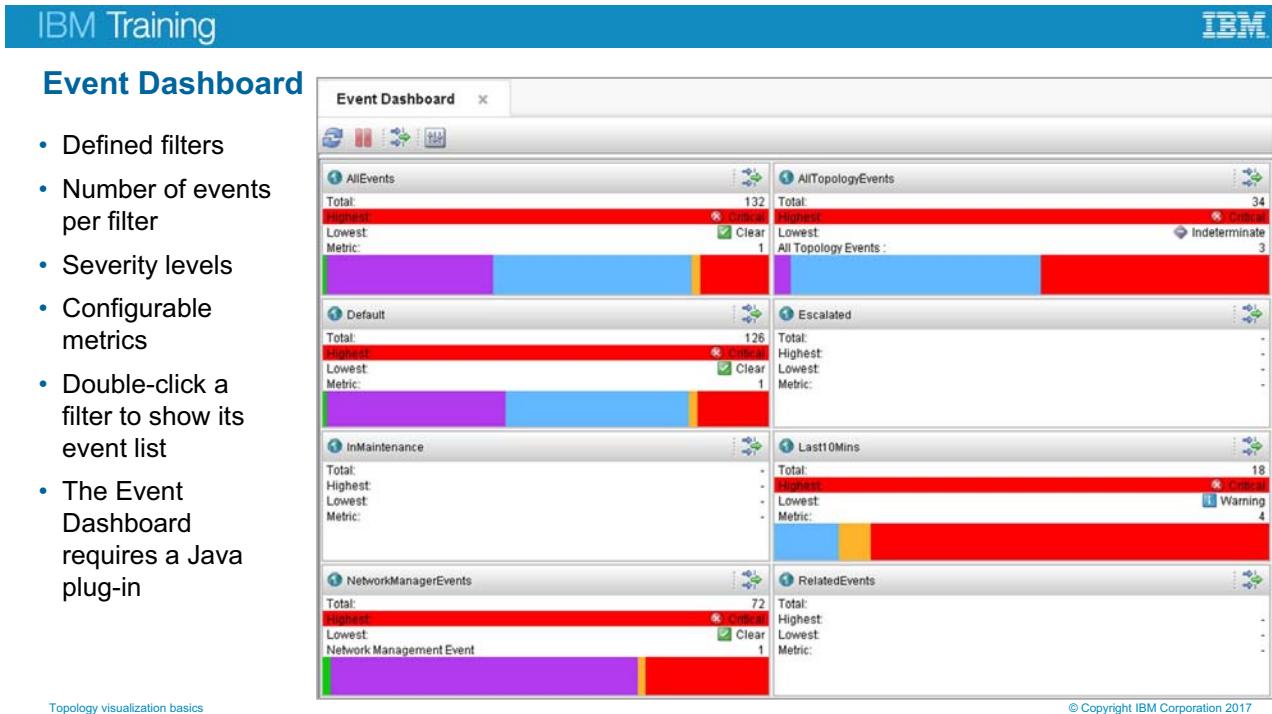


Figure 5-7. Event Dashboard

Another term for the **Event Dashboard** is **Monitor boxes**.



Information

The Event dashboard and the Active Event List (AEL) both require a Java plug-in on the system where the browser is running. The AEL's functionality is deprecated in the **Event Viewer**, which does not require a Java plug-in. If you cannot view the Event Dashboard or AEL, then your browser needs the Java libraries. Usually, installing the Java runtime environment (JRE) on your local workstation solves this problem. Your lab machine does not have the Java plug-in for Firefox loaded.

The active event list requires a Java plug-in or runtime engine (JRE) to be installed on your workstation so that it works with your browser

You can configure properties to change the way it looks

Sev	Ack	Node	Alert Group	Summary	First Occurrence	Last Occurrence	Count	Type	ExpireTime	Agent	Manager
!	Yes	10.10.255.17	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.17...	8/10/16, 1:42 PM	8/11/16, 12:25 PM	682	Problem	Not Set	ncp_poller	ITNM
!	No	10.10.255.15	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.15...	8/10/16, 1:42 PM	8/11/16, 1:23 PM	682	Problem	Not Set	ncp_poller	ITNM
!	No	10.10.255.14	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.14...	8/10/16, 1:42 PM	8/11/16, 1:23 PM	682	Problem	Not Set	ncp_poller	ITNM
!	No	10.10.255.13	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.13...	8/10/16, 1:42 PM	8/11/16, 1:23 PM	682	Problem	Not Set	ncp_poller	ITNM
!	No	10.10.255.16	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.16...	8/10/16, 1:42 PM	8/11/16, 1:24 PM	682	Problem	Not Set	ncp_poller	ITNM
!	No	10.10.255.1	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.1:1...	8/10/16, 1:42 PM	8/11/16, 1:24 PM	682	Problem	Not Set	ncp_poller	ITNM
!	No	10.10.255.7	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.7:1...	8/10/16, 1:42 PM	8/11/16, 1:23 PM	682	Problem	Not Set	ncp_poller	ITNM
!	No	host1.csite.edu	WebGUI Status	ALERT: Web GUI OMNIBUS data source METRIC cache size is 3...	8/10/16, 1:42 PM	8/11/16, 1:23 PM	1	Problem	Not Set	OMNIBus Self...	OMNIBus Self...
!	No	NOI_AGG_P	ITNM Status	NM Storm failed heartbeat check. Time of last heartbeat: 2016-08...	8/10/16, 1:42 PM	8/11/16, 1:23 PM	45	Problem	Not Set	ncp_poller	ITNM
!	No	host1.csite.edu	WebGUI Status	Web GUI OMNIBUS data source is available	8/10/16, 1:42 PM	8/11/16, 1:23 PM	272	Information	Not Set	OMNIBus Self...	OMNIBus Self...
!	No	host1.csite.edu	WebGUI Status	Web GUI NOI_AGG_P data source is available	8/10/16, 1:42 PM	8/11/16, 1:23 PM	272	Information	Not Set	OMNIBus Self...	OMNIBus Self...

Figure 5-8. Active Event List (AEL)

The **Event Viewer** duplicates the functions of the **Active Event List** but eliminates the use of Java. If you use on the Event Viewer, you do not need to have a Java Runtime Engine (JRE) installed. Double-clicking an event in the AEL or the Event Viewer opens a window with three tabs:

- The **Fields** tab shows all of the fields from the OMNIbus ObjectServer **alerts.status** table. You can see the value in each column of the event row.
- The **Detail** tab is often empty. However, if a probe rules file was unable to fully parse all of the data in the raw event stream from a device, that raw data appears in this tab.
- The **Journal** tab shows a record of people or processes that modified the event.

Right-clicking an event opens a contextual menu that enables a user with sufficient rights to manage the event or see the associated topology information for the device in Tivoli Network Manager.

The screenshot shows the IBM Event Viewer window. The title bar says "IBM Training" and "Event Viewer". The main area is a table with columns: Sev, Ack, Node, Alert Group, Summary, First Occurrence, and Last Occurrence. The table contains several rows of event data. A yellow callout box on the right side of the table lists two bullet points:

- Does not require Java plug-in
- Has the same functions as the Active Event List (AEL)

Sev	Ack	Node	Alert Group	Summary	First Occurrence	Last Occurrence
☒	No	10.1.1.20	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.1.1.20: ICMP timeout	5/2/17, 4:48 PM	5/2/17, 5:59 PM
☒	No	10.1.1.19	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.1.1.19: ICMP timeout	5/2/17, 5:45 PM	5/2/17, 5:59 PM
☒	No	10.1.1.18	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.1.1.18: ICMP timeout	5/2/17, 5:45 PM	5/2/17, 5:59 PM
☒	No	10.1.1.3	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.1.1.3: ICMP timeout	5/2/17, 5:45 PM	5/2/17, 5:59 PM
☒	No	10.1.254.1	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.1.254.1: ICMP timeout	5/2/17, 5:45 PM	5/2/17, 5:59 PM
⚠	No	NOI_AGG_P	ITNM Status	NM Storm failed heartbeat check. Time of last heartbeat: 2017-02-09 17:33:46	10/18/16, 3:59 PM	5/1/17, 7:23 PM
ℹ	No	192.168.100.100	UOW trap	Import UOW '12' submitted by administrator is executing (2587947668) 299 days, 12:44:36.6	10/26/16, 12:43 PM	10/26/16, 12:44 PM
ℹ	No	192.168.100.100	UOW trap	Autodiscovery UOW '30' submitted by administrator is executing (2952290876) 341 days,		
ℹ	No	192.168.100.100	UOW trap	Autodiscovery UOW '5' submitted by administrator is executing (2587933143) 299 days,		
ℹ	No	192.168.100.100	UOW trap	Autodiscovery UOW '6' submitted by administrator is executing (2587934142) 299 days,		
ℹ	No	192.168.100.100	Resource Event	Manage Cisco Device 'PAR-Core-01'		
ℹ	No	192.168.100.100	UOW trap	Autodiscovery UOW '1' submitted by administrator is executing (2587931140) 299 days, 12:	10/26/16, 12:41 PM	10/26/16, 12:41 PM
ℹ	No	192.168.100.100	Resource Event	Create Unknown Device 'CPU Core 01'	10/26/16, 12:41 PM	10/26/16, 12:41 PM

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-9. Event Viewer

The **Event Viewer** duplicates the functions of the **Active Event List** but eliminates the need for a Java Runtime Engine (JRE) on a user workstation. Double-clicking an event in the Event Viewer opens a window with three tabs:

- The **Fields** tab shows all of the fields from the OMNIbus ObjectServer **alerts.status** table. You can see the value in each column of the event row.
- The **Detail** tab is often empty. However, if a probe rules file was unable to fully parse all of the data in the raw event stream from a device, that raw data appears in this tab.
- The **Journal** tab shows a record of people or processes that modified the event.

Right-clicking an event opens a contextual menu that enables a user with sufficient rights to manage the event or see the associated topology information for the device in Tivoli Network Manager.

IBM Training

Key Performance Indicators

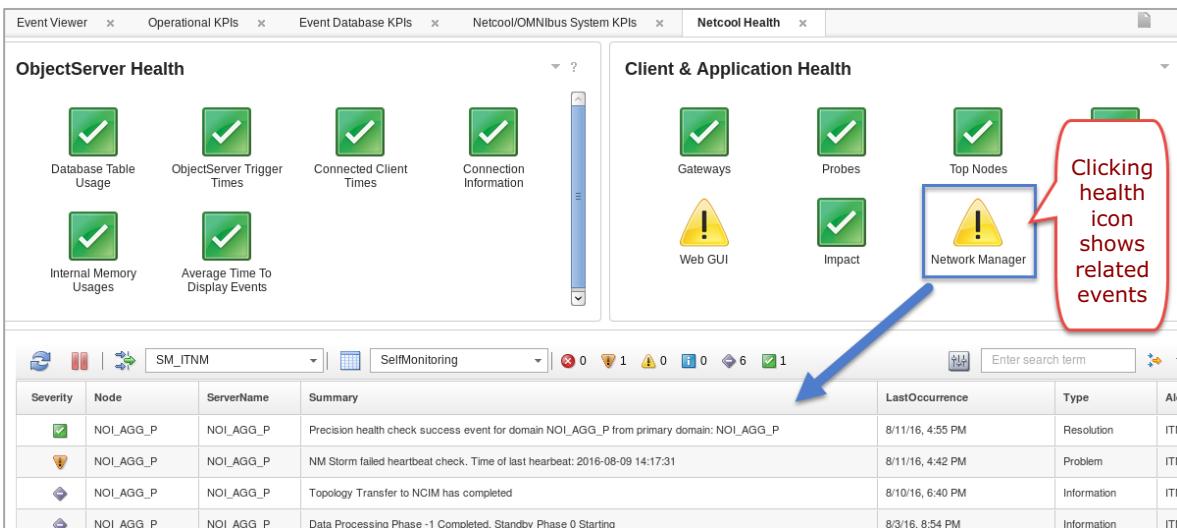
The screenshot shows the IBM KPI interface. On the left is a sidebar with icons for Incident, Events, and KPIs. Under KPIs, there are links for Operational KPIs, Event Database KPIs, Netcool/OMNIbus System KPIs, and Netcool Health. The main area is titled "Operational KPIs" and displays three status icons: "Open Incidents" (green checkmark), "Escalated" (green checkmark), and "Old Unacknowledged" (red X). Below these is a timestamp: "Last Updated: 04:43:58 PM". To the right of the sidebar are eight circular gauges arranged in two rows of four. The top row is labeled "Incidents", "Escalated", "Unacknowledged", and "Acknowledged". The bottom row is labeled "Status", "Last Minute", "Journals", and "Details". Each gauge has a green arc at the top and a red arc at the bottom. Numerical values are displayed in the center of each gauge: 24.15k, 0, 128, 1, 130, 2, 100k, and 10k respectively.

- Refreshes every 10 seconds by default (configurable)
- Key Performance Indicators (KPIs)
 - Operational status
 - Event Database
 - Netcool/OMNIbus System
 - Netcool Health
- Gauges can be published to mobile devices

Figure 5-10. Key Performance Indicators

IBM Training 

Netcool Health KPI



ObjectServer Health

- Database Table Usage
- ObjectServer Trigger Times
- Connected Client Times
- Connection Information
- Internal Memory Usages
- Average Time To Display Events

Client & Application Health

- Gateways
- Probes
- Top Nodes
- Web GUI
- Impact
- Network Manager

Clicking health icon shows related events

Severity	Node	ServerName	Summary	LastOccurrence	Type
	NOI_AGG_P	NOI_AGG_P	Precision health check success event for domain NOI_AGG_P from primary domain: NOI_AGG_P	8/11/16, 4:55 PM	Resolution
	NOI_AGG_P	NOI_AGG_P	NM Storm failed heartbeat check. Time of last heartbeat: 2016-08-09 14:17:31	8/11/16, 4:42 PM	Problem
	NOI_AGG_P	NOI_AGG_P	Topology Transfer to NCIM has completed	8/10/16, 6:40 PM	Information
	NOI_AGG_P	NOI_AGG_P	Data Processing Phase -1 Completed. Standby Phase 0 Starting	8/3/16, 8:54 PM	Information

Figure 5-11. Netcool Health KPI

IBM Training IBM

Creating an event filter (1 of 2)

1. Click the filter builder icon in AEL 
2. Click the New Filter icon 
3. Select users who can access the filter
 - A **filter** determines what rows of data are visible

New Filter
Select users who will have access to the filter

Public:

 global
 system

Users:

 itnmadmin
 itnmuser
 ncoadmin
 ncouser
 smadmin

Groups:

 Netcool_Admin
 Netcool_User

Ok **Clear**

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-12. Creating an event filter (1 of 2)

An **event filter** determines what rows of data the user can see. An **event view** determines what columns appear in the event list. Click the event filter icon that is shown in step 1 of this example to start creating a new event filter. The Netcool Operations Insight dashboard administrator can restrict users so that they can view only data that matches a list of defined filters.

IBM Training IBM

Creating an event filter (2 of 2)

The screenshot shows two windows related to creating an event filter:

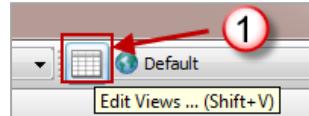
- Edit Filter: ITNM_Events** (Left Window):
 - Filter Attributes** section: Shows a preview icon and the name "ITNM_Events". A red box highlights the name field with the instruction "Name the filter (no spaces in name)".
 - Description:** A red box highlights the description input field with the instruction "Enter a description".
 - Filter Conditions** section: Shows a table with one row. A red box highlights the "Value" column with the instruction "Select filter criteria". A red arrow points from this box to the "Value" column.
 - Buttons:** "Save and Close" (highlighted with a red box), "Save", and "Close".
- Event Viewer** (Right Window):
 - Shows a list of network events. The search bar at the top contains "ITNM_Events".
 - The list includes several entries, such as:
 - ITNM Monitor: Default Chassis Ping/Default Chassis Ping fail for 10.10.255.12; ICMP timeo...
 - ITNM Monitor: Default Chassis Ping/Default Chassis Ping fail for 10.10.255.15; ICMP timeo...
 - ITNM Monitor: Default Chassis Ping/Default Chassis Ping fail for 10.10.255.16; ICMP timeo...
 - ITNM Monitor: Default Chassis Ping/Default Chassis Ping fail for 10.10.255.17; ICMP timeo...
 - ITNM Status: NM Storm failed heartbeat check. Time of last heartbeat: 2016-09-09 14:17:...
 - ITNM Status: Data Collection Phase 1 Completed. Data Collection Phase 2 Starting
 - ITNM Status: Topology Transfer to NCM has completed
 - ITNM Status: Data Processing Phase -1 Completed. Standby Phase 0 Starting

Figure 5-13. Creating an event filter (2 of 2)

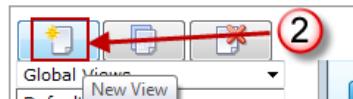
Create a view (1 of 2)

A **view** determines what columns are shown

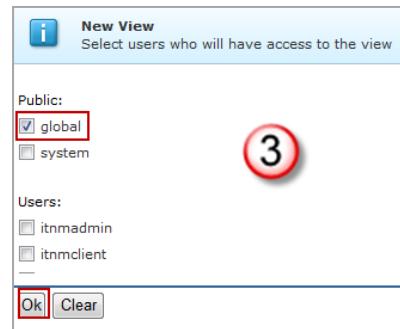
1. Click the **Edit Views** icon in an AEL



2. Click the **New View** icon



3. Select users who can access view and click **OK**



© Copyright IBM Corporation 2017

Figure 5-14. Create a view (1 of 2)

A **view** determines what columns of an event appear in the user display.

IBM Training IBM

Create a view (2 of 2)

4. Name view (no spaces)
5. Confirm ObjectServer Data Source
6. Select columns for view; use arrows to order columns
7. Specify column width and justification
8. Specify how to sort events; use arrows to prioritize fields for sorting
9. Click Save and Close

Figure 5-15. Create a view (2 of 2)

Filter and view names cannot contain a space as part of the name. The space is evaluated as a delimiter that causes Tivoli Network Manager to parse the name incorrectly. Use an underscore or hyphen to separate words when it is necessary.

New view is available

The screenshot shows the Event Viewer interface with a new view definition named "Guru". The "Event Viewer" tab is selected. A red box highlights the "Guru" view name in the top navigation bar. Another red box highlights the "Node" column header in the main table. A callout bubble points to the "Node" column header with the text "Only columns in specified view definition are shown". A red arrow points from the "Guru" view name to the "Node" column header.

Group	Sev	Node	Summary	NmosCauseType
All	119	host1.csite.edu	ALERT: Web GUI NOI_AGO_P data source VMM sync maximum response time: 32.468 secs	Unknown
10.10.255.10	2	host1.csite.edu	ALERT: Web GUI Security Repository maximum response time: 29.535 secs	Unknown
10.10.255.12	1	10.10.255.12	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.12: ICMP timeout	Root Cause
10.10.255.15	1	host1.csite.edu	ALERT: Web GUI Security Repository average response time: 25.633 secs	Unknown
10.10.255.16	1	host1.csite.edu	ALERT: Web GUI NOI_AGO_P data source EVENTLIST cache size is 1, hit rate: 20%	Unknown
10.10.255.17	1	10.10.255.17	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.17: ICMP timeout	Symptom
192.168.100.100	47	10.10.255.16	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.16: ICMP timeout	Symptom
host1	1	10.10.255.15	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.15: ICMP timeout	Symptom
host1.csite.edu	6	host1.csite.edu	Web GUI OMNIBUS data source is available	Unknown
host1.csite.edu.localdomain	1	host1.csite.edu	Web GUI NOI_AGO_P data source is available	Unknown
NOI_AGO_P	58	192.168.100.100	Import UOW '17 submitted by administrator is executing (1864971541) 215 days, 20:28:35.41	Unknown

Figure 5-16. New view is available

© Copyright IBM Corporation 2017

Show root cause event from symptom event

- Select symptom event
- Right-click the event and select **Show Root Cause**
- View the map that shows the root cause for this symptom

The screenshot shows a list of events in Tivoli Network Manager. A red callout box points to the context menu for the top event, which is highlighted in yellow. The menu includes options like Acknowledge, De-acknowledge, Prioritize, Suppress/Escalate, Take ownership, User Assign, Group Assign, Delete, Network Manager Reports, Configuration Management, Ping, Event Search, Find In Hop View, Find In Network View, Find in Path View, Trace IP Path, Show Device Structure, Open SNMP MIB Browser, Open SNMP MIB Grapher, Show Root Cause, and Show Suppressed Events. The 'Show Root Cause' option is highlighted with a red border.

Node	Alert Group	Summary	Last Occurrence	Type	Cause Type
10.10.255.12	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.12: ICMP timeout	8/17/16, 5:24 PM	Problem	Root Cause

© Copyright IBM Corporation 2017

Figure 5-17. Show root cause event from symptom event

Tivoli Network Manager correlates root cause events to symptom events. It maintains this correlation record so that you can right-click a symptom event and see the root cause event. You can also right-click a root cause event and see all associated symptom events.

Show suppressed events from root cause event

Tivoli Network Manager maintains the correlation between root cause events and symptom events

Summary	NmosCauseType	Count
ALERT: Web GUI NOI_AGG_P data source VMM sync maximum response time: 32.468 secs	Unknown	562
Default Chassis Ping/Default Chassis Ping fail for 10.10.255.12; ICMP timeout	Root Cause	
ALERT: Web GUI Security Repository average response time: 25.633 secs	Unknown	
ALERT: Web GUI NOI_AGG_P data source VMM sync maximum response time: 32.468 secs	Unknown	
Default Chassis Ping/Default Chassis Ping fail for 10.10.255.12; ICMP timeout	Symptom	
Default Chassis Ping/Default Chassis Ping fail for 10.10.255.12; ICMP timeout	Symptom	
Default Chassis Ping/Default Chassis Ping fail for 10.10.255.12; ICMP timeout	Symptom	
Web GUI OMNIBUS data source is available	Unknown	
Web GUI NOI_AGG_P data source is available	Unknown	
Import UOW '17' submitted by administrator is executing (1864971541) 215 days, 20:28:35.41	Unknown	

Node	Alert Group	Summary	Last Occurrence	Type	Cause Type
10.10.255.17	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.17; ICMP timeout	8/17/16, 5:30 PM	Problem	Symptom
10.10.255.16	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.16; ICMP timeout	8/17/16, 5:30 PM	Problem	Symptom
10.10.255.15	ITNM Monitor	Default Chassis Ping/Default Chassis Ping fail for 10.10.255.15; ICMP timeout	8/17/16, 5:30 PM	Problem	Symptom

Import UOW '27' submitted by administrator is executing (1864983604) 215 days, 20:30:36.04 Unknown

Right-click root cause event and select
Show Suppressed Events

Acknowledge Ctrl+A
 De-acknowledge Ctrl+D
 Prioritize >
 Suppress/Escalate >
 Take ownership >
 User Assign >
 Group Assign >
 Delete
 Network Manager Reports >
 Configuration Management >
 Ping >
 Event Search >
 Find In Hop View
 Open SNMP MIB Browser
 Open SNMP MIB Grapher
 Show Root Cause
Show Suppressed Events
 Show SAE Related Events
 Show SAE Related Services

Figure 5-18. Show suppressed events from root cause event

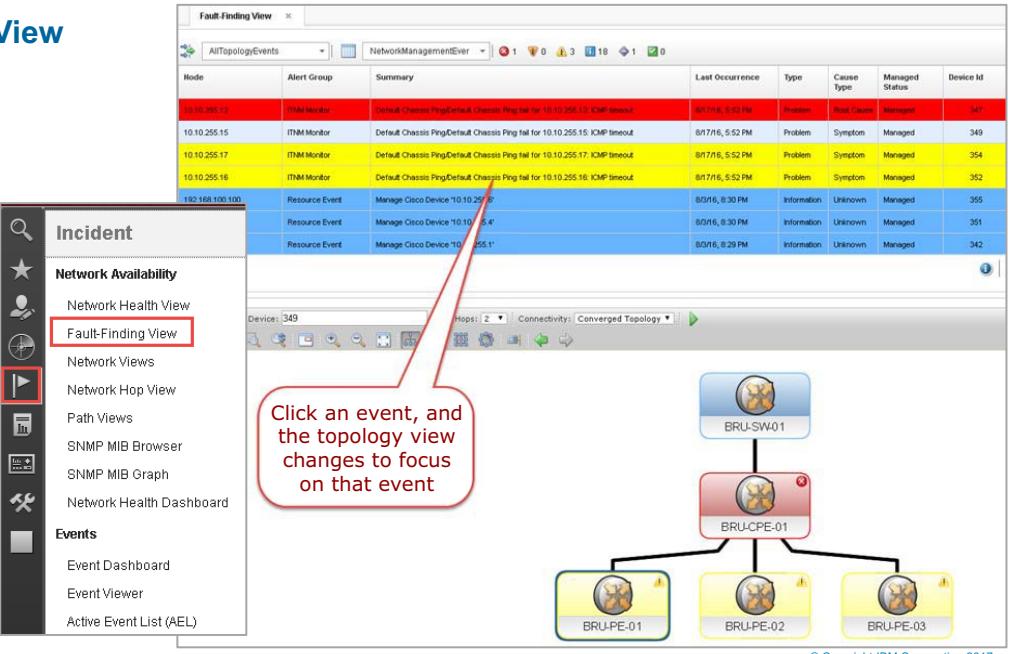
Topology visualization basics

© Copyright IBM Corporation 2017

IBM Training 

Fault Finding View

- AEL is shown in the top window
- When you click an event, that device is shown in the **Network Hop View** in the bottom window with a specified number of hops in each direction



Node	Alert Group	Summary	Last Occurrence	Type	Cause Type	Managed Status	Device Id
10.10.255.13	ITM Monitor	Default Chassis Ping fail for 10.10.255.13 ICMP timeout	8/17/16, 5:33 PM	Problem	Symptom	Managed	347
10.10.255.15	ITM Monitor	Default Chassis Ping fail for 10.10.255.15 ICMP timeout	8/17/16, 5:32 PM	Problem	Symptom	Managed	349
10.10.255.17	ITM Monitor	Default Chassis Ping fail for 10.10.255.17 ICMP timeout	8/17/16, 5:32 PM	Problem	Symptom	Managed	354
10.10.255.16	ITM Monitor	Default Chassis Ping fail for 10.10.255.16 ICMP timeout	8/17/16, 5:32 PM	Problem	Symptom	Managed	352
192.168.100.100	Resource Event	Manage Cisco Device "10.10.255.100"	8/17/16, 8:30 PM	Information	Unknown	Managed	365
	Resource Event	Manage Cisco Device "10.10.255.4"	8/17/16, 8:30 PM	Information	Unknown	Managed	361
	Resource Event	Manage Cisco Device "10.10.255.1"	8/17/16, 8:29 PM	Information	Unknown	Managed	342

Figure 5-19. Fault Finding View

The **Fault Finding View** shows events in the top portion of the window. Clicking any event changes the Hop View in the bottom portion of the window so that the view focuses on the device that is associated with the event. This view requires sufficient screen space to be useful.

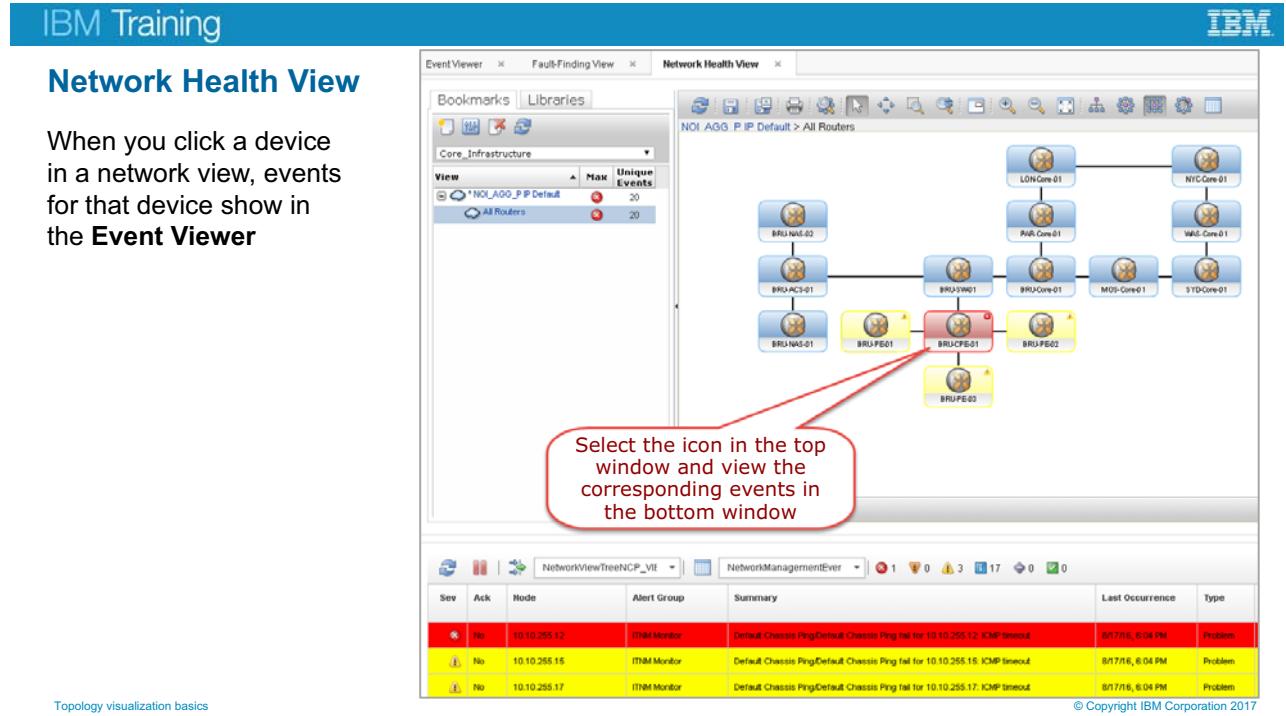


Figure 5-20. Network Health View

The **Network Health View** shows topology in the top window. The bottom window shows events that are associated with the in-focus device. A device is in-focus when it has a thicker border around the device icon.

Topology visualization icons

Icon	Name
	Router icon
	Switch icon
	Radio area network (RAN) router icon
	Radio area network (RAN) switch icon
	Element management system (EMS) icon
	Cell icon
	Geographical location icon

Icon	Name
	Geographical region icon
	Generic logical collection icon
	End node icon
	Unknown device icon

Icon	Name
	Subnet icon
	Manually added device or connection between devices
[4]	Number of connections indicator
	Completely unmanaged device
	Partially unmanaged device

Icon	Name
	Antenna icon
	Equipment Identity Register (EIR) icon
	Evolved NodeB (eNodeB) icon
	eUtranCell icon
	Home Subscriber Server (HSS) icon
	LTE pool icon

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-21. Topology visualization icons

Icons with a question mark designate an unknown device. This designation occurs under one of the following conditions:

- Tivoli Network Manager did not have SNMP access to a device that was located by a finder.
- The sysObjectID (or OID) number of the device did not match any of the filters in the Active Object Class (AOC) files. These files are located in the \$ITNMHOME/aoc directory. If you know what the device is, you can add the OID number to the appropriate AOC file. For example, you might have a Cisco 72033 router in your network. You applied an iOS software update to the router. When you run a discovery, the device shows as an unknown device because the OID number is not recognized. Software updates to devices can cause the OID number to change. In this situation, you can locate the appropriate AOC file and add the new OID number. The best place to add an OID number is on the line that precedes the last OID number as seen in the following example.

```
//*****
//  
// File: CiscoS72xx.aoc  
//  
// This file was automatically generated for  
// Cisco 72033 Router  
//  
//*****  
active object 'CiscoS72xx'  
{  
    super_class = 'Cisco';  
    instantiate_rule = "EntityOID = '1.3.6.1.4.2.9.1.832' OR  
                        EntityOID = '1.3.6.1.4.2.9.1.534' OR  
                        EntityOID = 'Add new OID number here' OR  
                        EntityOID = '1.3.6.1.4.2.9.1.835'";  
    visual_icon = 'Router';  
};
```

The Tivoli Network Manager GUI uses two different wrench icons to indicate the extent to which a device is unmanaged. If an entire chassis is unmanaged, the device icon is overlaid with a two-ended wrench. If only a portion of a device is unmanaged, the device icon is overlaid with a one-ended wrench.

Find in Hop View from event list

- Right-click an event in the Event Viewer or AEL
- Select **Find in Hop View**
- A topology map starts in a new window, and the map is centered on the selected device
- You can change the number of hops or the view structure
- NmosObjInst** must exist in the database for the selected device, or no map is shown

Default Chassis Ping/Default Chassis Ping fail for 10.10.255.12: ICMP timeout	Root Cause
Web GUI OMNIBUS data source is available	Unknown
Web GUI NOI_0GG_P data source is available	Unknown
Import UOW '17' submitted by administrator is executing (1864971541) 215 days, 20:28:35.41	Unknown
Autodiscovery UOW '5' submitted by administrator is executing (1864953493) 215 days, 20:2	Unknown
Autodiscovery UOW '15' submitted by administrator is executing (1864970531) 215 days, 20:	Unknown
Autodiscovery UOW '6' submitted by administrator is executing (1864959506) 215 days, 20:2	Unknown
Autodiscovery UOW '11' submitted by administrator is executing (1864864317) 215 days, 20:1	Unknown
Autodiscovery UOW '11' submitted by administrator is executing (1864965522) 215 days, 20:	Unknown
Import UOW '23' submitted by administrator is executing (1864979588) 215 days, 20:29:55.88	Unknown
Import UOW '25' submitted by administrator is executing (1864980595) 215 days, 20:30:05.95	Unknown

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-22. Find in Hop View from event list

When you right-click an event and select **Find in Hop View**, a window opens focused on that device in the topology map. You can select from Layer 2, Layer 3, or subnet views.

IBM Training

Find in Network View from event list

The screenshot shows a network topology with several nodes: BRU-PE-03, BRU-CPE-01, BRU-PE-02, and BRU-PE-01. A context menu is open over an event entry in the event list. The menu includes options like Root Cause, Acknowledge, De-acknowledge, Prioritize, Suppress/Escalate, Take ownership, User Assign, Group Assign, Delete, Network Manager Reports, Configuration Management, Ping, Event Search, Find In Hop View, Find In Network View, and Find in Path View. The 'Find In Network View' option is highlighted with a red box.

Find in Network View

Device found in multiple views:

- Cisco36xx
- Minor
- PingFailRootCause
- All Routers
- OSPF_Routing_Domain[1]

Select the view that you want, and click **OK**

Selected device icon appears in selected network view

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-23. Find in Network View from event list

When you right-click an event and select **Find in Network View**, Tivoli Network Manager shows you a list of all network views that include that device. Select the network view that you want and click **OK** to see the device.

5.3. The Hop View

You use the Hop and Network views more than other views in Tivoli Network Manager.

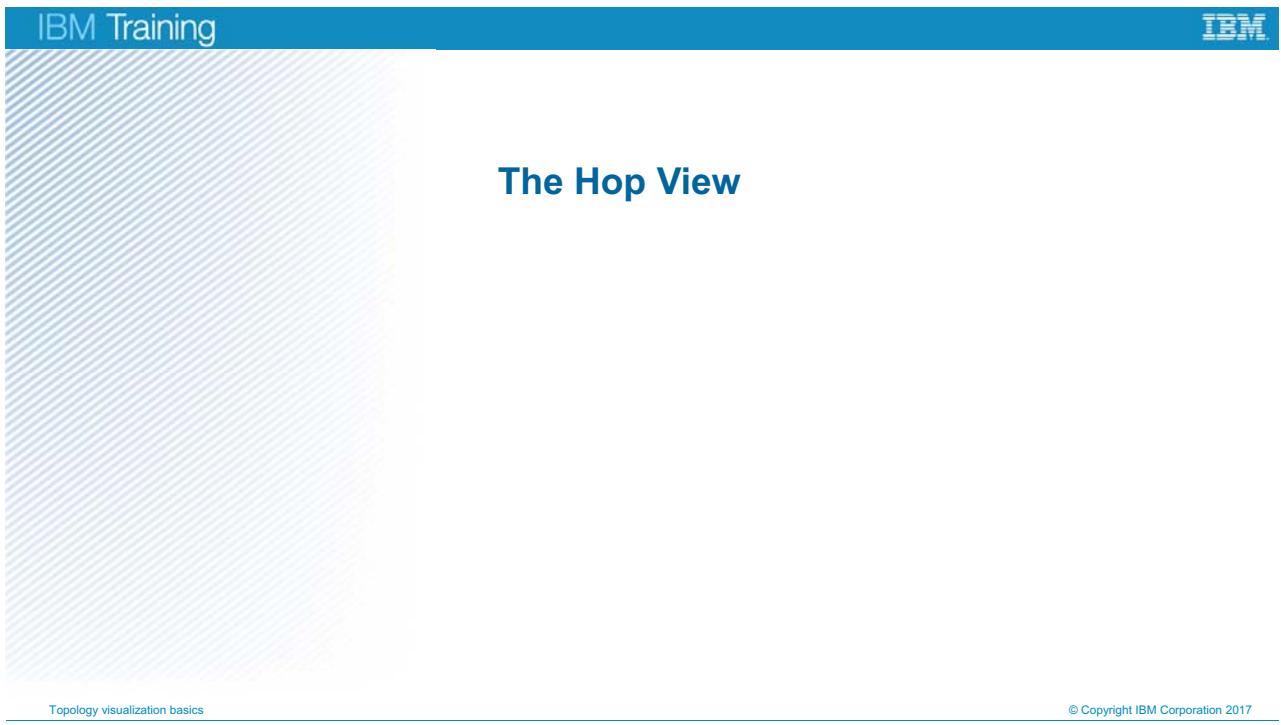


Figure 5-24. The Hop View

A **Hop View** shows a specified device and all devices within a specified number of hops.

Hop View

1. Select Network Hop View
2. Enter a **Seed** value and the number of hops
3. Click the **Play** icon to view the topology

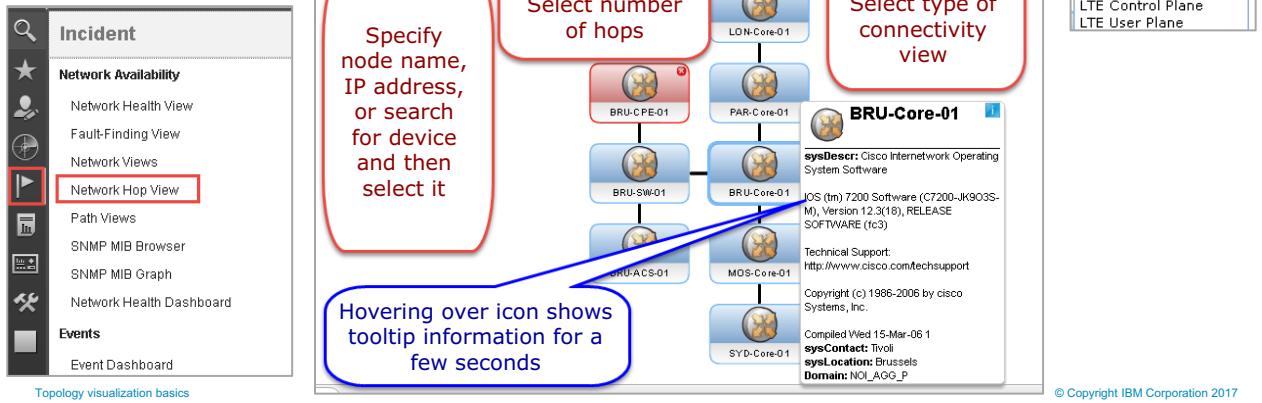


Figure 5-25. Hop View

In the **Seed Device** box, you can enter one of three values:

- IP address of the device
- **EntityName** of the device
- **NmosObjectId** of the device

You can select from a number of connectivity views:

- **Layer 1**
- **Layer 2**
- **Layer 3**
- **IP Subnets**
- **Converged Topology**
- **PIM**
- **IPMRoute**
- **Microwave**
- **Logical RAN**

IBM Training



Search for a device

1. Specify search criteria
 - Use percent sign (%) as wildcard character
2. Click **Find**
3. Select seed device from **Results** window
4. Click **Select and Close**
5. The selected device name now appears in the **Seed Device** box
 - Clicking the green play arrow shows the device in the **Network Hop View**

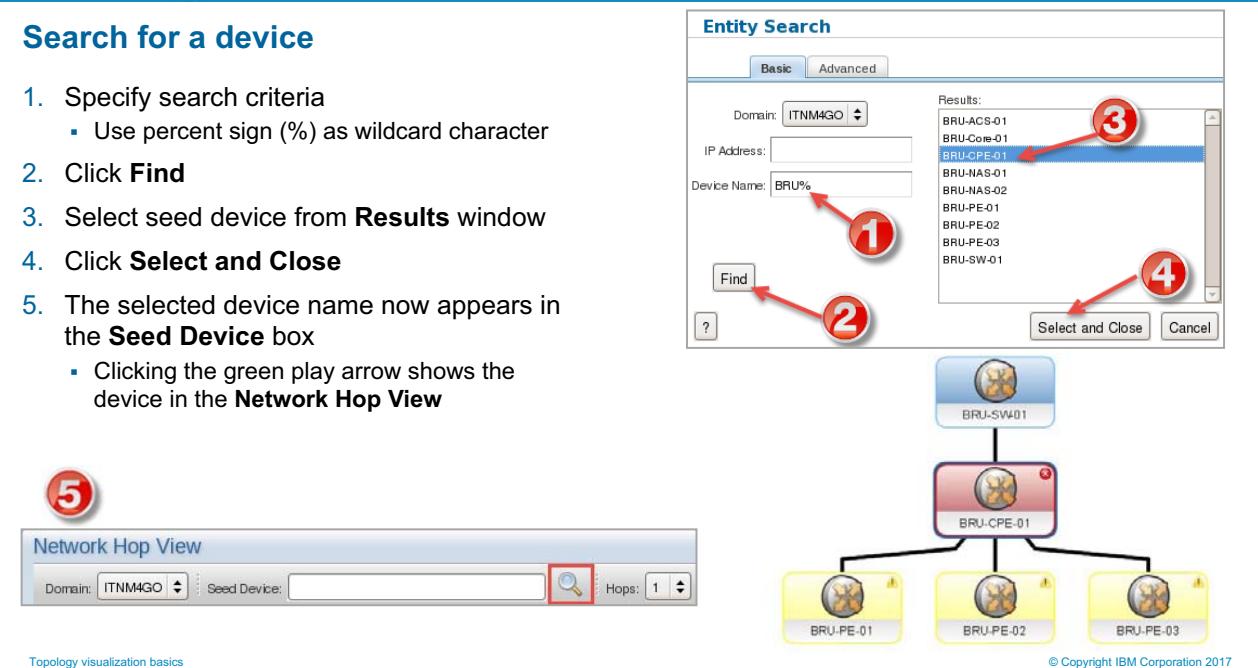


Figure 5-26. Search for a device

1. Click the **Search** icon and then enter all or part of a device name or IP address.
2. Use the percent sign (%) as a wildcard character and click **Find**.
3. Click a device in the **Results** window and then click the **Select and Close** icon.
4. The selected device name appears in the **Seed** box.
5. Select a number of hops and a type of view and then click the green play arrow to view the selected device.

Graph layout icons

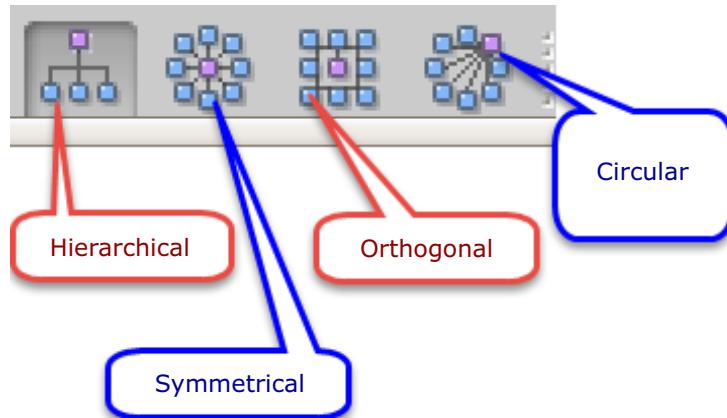


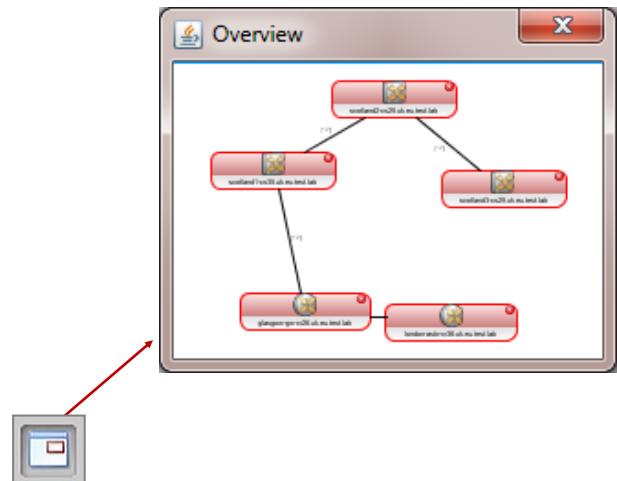
Figure 5-27. Graph layout icons

Layouts have four types:

- **Hierarchical** view shows the selected device at the top of the view and connected devices underneath it.
- **Symmetrical** view shows the selected device near the center of the view with the other devices that surround it.
- **Orthogonal** view shows all connectivity lines that are drawn either horizontally or vertically (and not diagonally).
- **Circular** view shows the selected device anywhere in the view with other devices that are evenly distributed within the view.

Zoom features

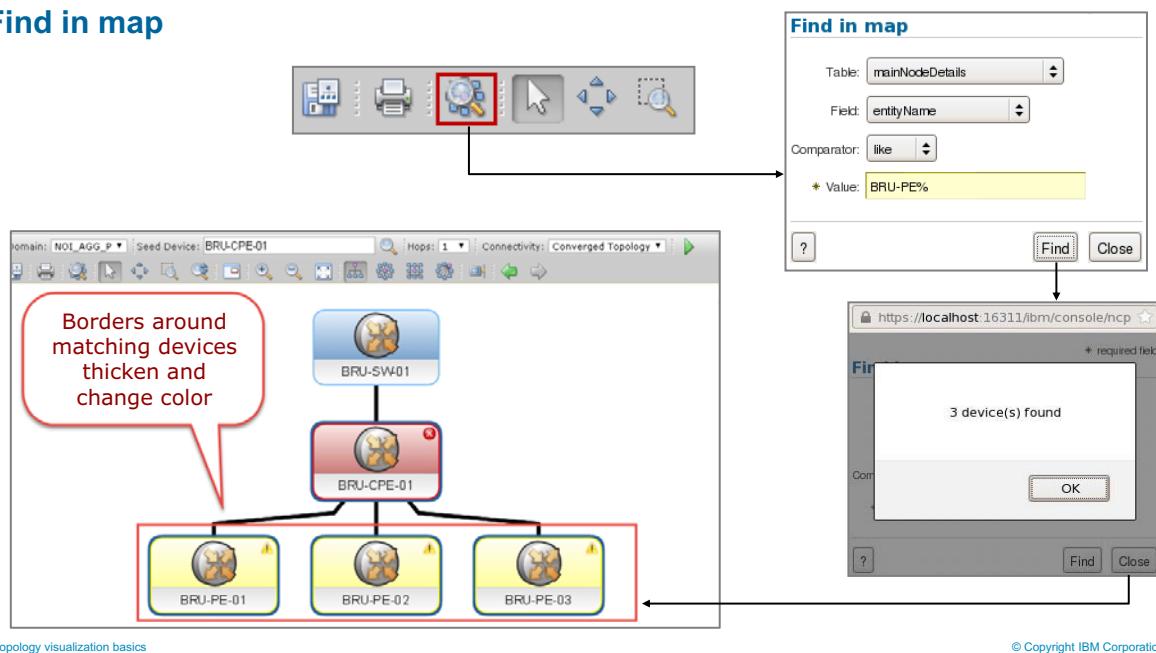
- Zoom in 
- Zoom out 
- Fit map within screen 
- Interactive zoom
 - Click 
 - Drag mouse up and down to zoom in and out
- Select area to zoom 
- Toggle overview
 - Pops up a pan window to quickly scroll through larger map



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-28. Zoom features

Find in map

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-29. Find in map

When many icons populate a network or hop view, it can be difficult to locate a specific device quickly. Click the **Find in map** icon and enter the name of the IP address or host name. You can also enter a portion of this information and use the percent sign (%) as a wildcard character. Tivoli Network Manager highlights each device in the view that matches the search criteria and indicates how many matches were found.

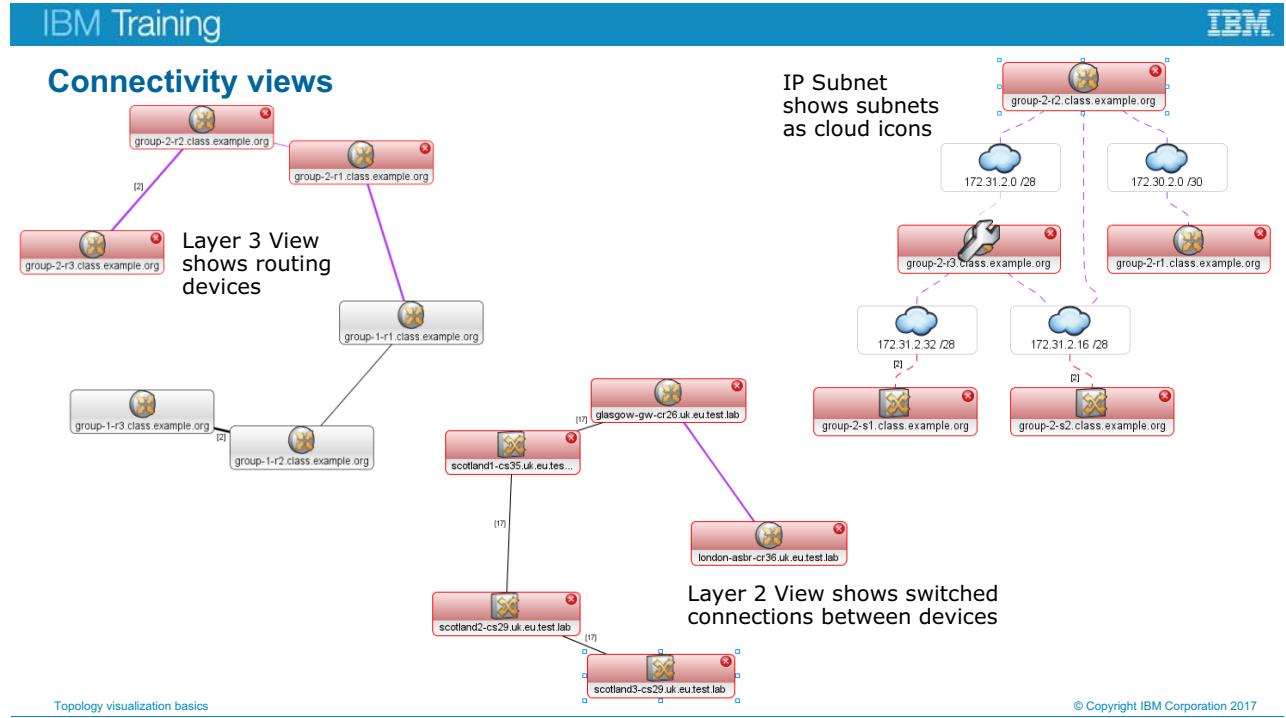


Figure 5-30. Connectivity views

Device icons reflect event severity

The `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/status.properties` file controls how a device icon represents the event status of a device

- It controls:
 - The views where device icons are updated with alert status
 - What icons (*.png) are used to indicate status
 - How often event status updates maps
 - Colors and saturation for icons in different views



The icon color reflects the color of the highest severity alarm for the device

- Properties control how the status is represented

```
# Enable/disable showing the status on the node background
status.node.background=false
# Enable/disable showing the status on the node label background
status.node.label=true
```

After you change **status.properties** or **topoviz.properties**, restart the Jazz SM DASH server

```
/opt/IBM/JazzSM/profile/bin/stopServer.sh server1 -username smadmin -password object00
/opt/IBM/JazzSM/profile/bin/startServer.sh server1
```

- Starting the GUI server does not require user name or password options

Figure 5-31. Device icons reflect event severity

The default behavior of Tivoli Network Manager is to change the color of icons in network maps to reflect the color that is associated with the most severe alert for that device. You can customize the way Tivoli Network Manager shows the status of a device.

In the **status.properties** file, you can enable and disable the display of network status in certain views.

```
# Enable/disable status on view topology maps and in table views.
status.view.enabled=true
# Enable/disable status on bookmarks trees.
status.tree.bookmarks.enabled=true
# Enable/disable status on libraries trees.
status.tree.libraries.enabled=true
# Enables/disables the "none" status
# By default, the "none" status is considered the same as a value of 0 (clear).
status.none.enabled=true
```

Other parameters in **status.properties** have comments that indicate their proper usage.

The screenshot shows a network visualization tool interface. At the top, there's a header bar with the IBM logo and some navigation icons. Below the header, the title "Network table layout" is displayed. The main area contains two tables. The left table, titled "NOI AGG P IP Default > All Routers", lists various routers with their display names, IP addresses, class names, class types, managed states, and maximum severities. The right table, also titled "NOI AGG P IP Default > All Routers", shows the same data but includes a "Events" column. A context menu is open over the right table, listing options like "Events", "Show Device Structure", "Show Connectivity Information", "Find events between two nodes", "Find in...", "Trace IP Path", "Add To View", "Webtools", "Reports", "Polling and MIB Info", "Discovery...", "Unmanage", "Configuration Management", "Launch To...", and "Ping". The bottom right corner of the interface has a copyright notice: "© Copyright IBM Corporation 2017".

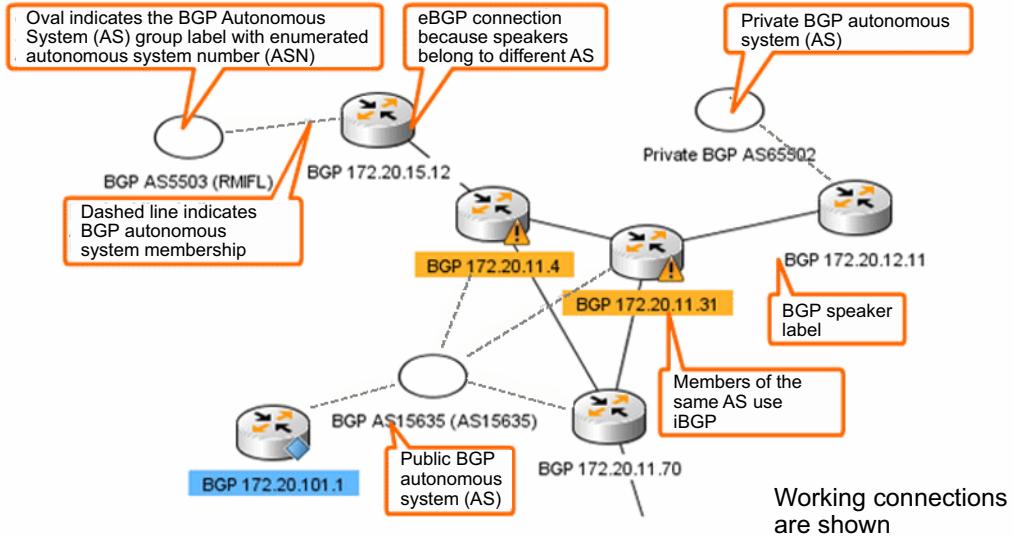
Figure 5-32. Network table layout

The user can now toggle between topology maps and the **network table layout**. This new layout provides a summary table that shows all devices in the current view and provides sorting and filtering capabilities to help operators navigate large network views. The network table layout displays the following summary data for each device in the current view:

- Display Name
- IP Address
- Class Name
- Class Type
- Managed Status
- Maximum Severity
- Alarm Status

The user can search, sort, and filter on any of these attributes. The user can launch tools in-context for any device in this view.

BGP visualization enhancements



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-33. BGP visualization enhancements

In BGP network views, ovals indicate the autonomous system (AS), and dotted lines connect the AS to each device that is a member of that AS.

OSPF visualization enhancements

- Ovals indicate group membership in OSPF areas
- Light dotted lines connect devices to the OSPF areas in which they are members

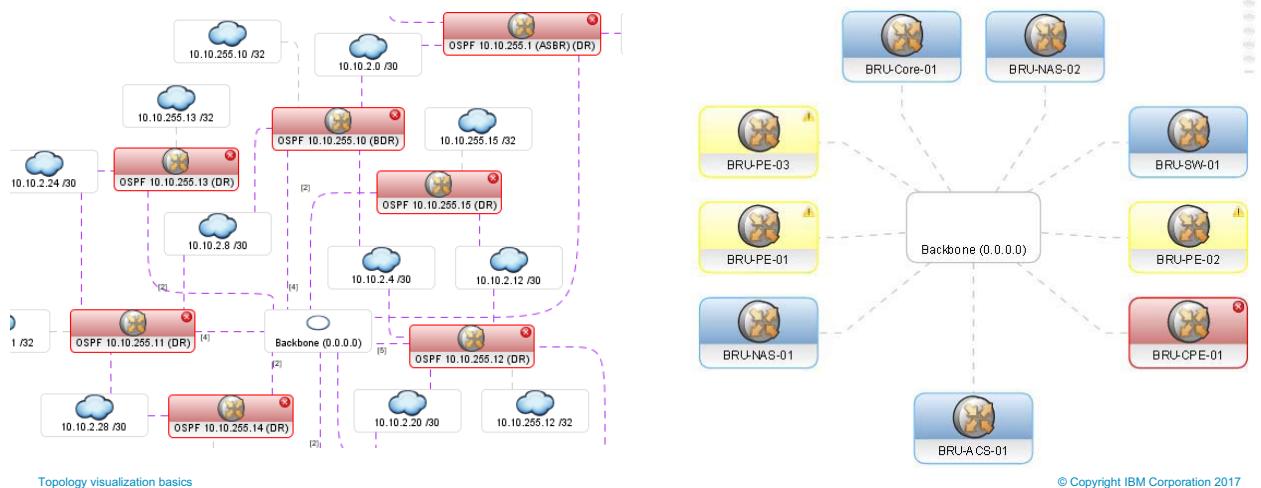


Figure 5-34. OSPF visualization enhancements

Ovals in network views of Open Shortest Path First (OSPF) networks indicate the area zones in the OSPF network. Dotted lines connect devices to the area zones of which they are a member.

A special icon exists for a link state advertisement (LSA). An LSA occurs when two OSPF devices are connected over an Ethernet LAN connection instead of a WAN connection.

Labels can also indicate the function of a device in the OSPF network:

- **DR** = designated router
- **BDR** = backup designated router
- **DROTHER** = router that is not the DR or the BDR
- **ASBR** = autonomous system border router
- **ABR** = area border router

5.4. Network views

The term “Network Views” is somewhat of a misnomer. These views are really collections of devices. For example, a network view can contain all routers, all switches, all Cisco Catalyst 2900 devices, or a multicast distribution tree (MDT). Each network view groups devices together by a common characteristic. Tivoli Network Manager creates many default network views after it completes a discovery. You can also create standard or dynamic network views.

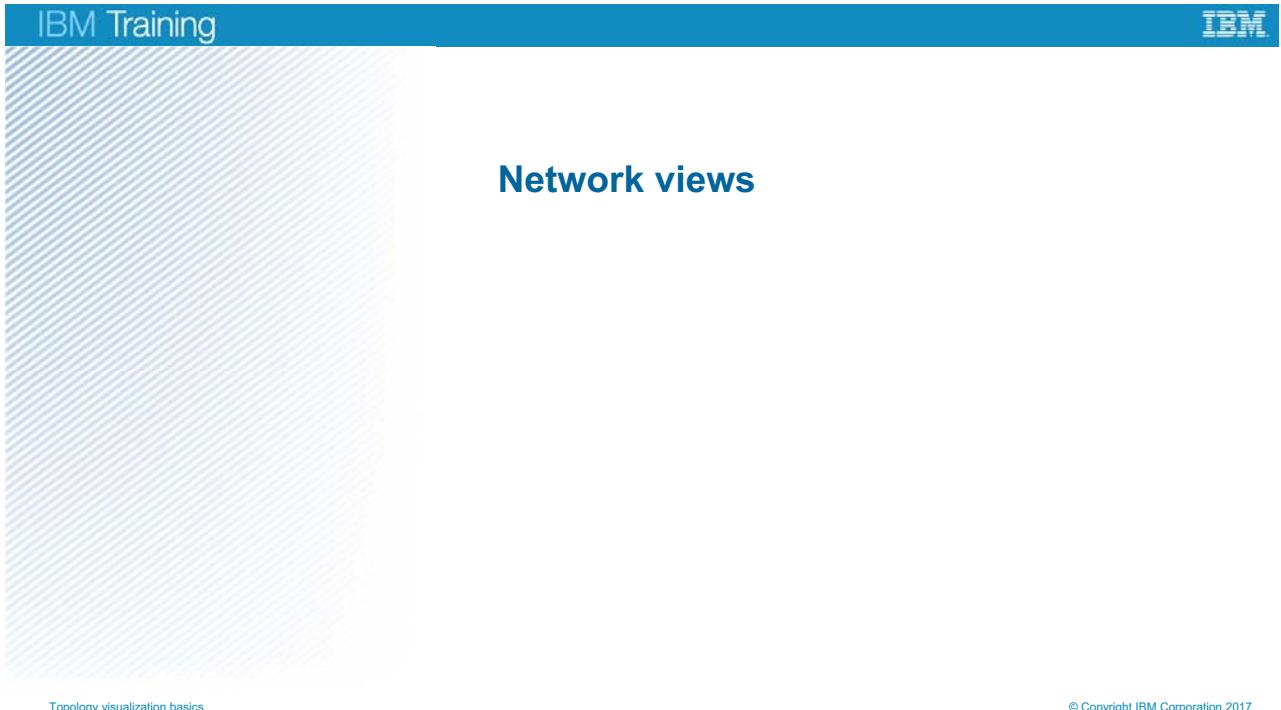


Figure 5-35. Network views

The screenshot shows the Tivoli Network Manager interface. On the left, there's a network topology diagram with various nodes representing routers and switches. Nodes include LON-Core-01, NYC-Core-01, BRU-NAS-02, PAR-Core-01, WAS-Core-01, MOS-Core-01, SYD-Core-01, BRU-ACS-01, BRU-NAS-01, BRU-PE-01, BRU-SW01, BRU-CPE-01, BRU-PE-02, and BRU-PE-03. Lines connect them to show their network connections. On the right, there's a sidebar titled 'itnmadmin Views' with a tree view of available network views. The tree includes categories like 'NOI_000_P IP Default', 'Alert views', 'Unacknowledged Alerts', 'Critical', 'Major', 'Minor', 'All Routers', 'All Switches', 'Custom View', 'Device Classes', 'Cisco36xx', 'Cisco7xxx', 'Linux', 'Manually Added Devices', and 'Monitoring Views'. Some items have red 'X' marks next to them.

Figure 5-36. Default network views (1 of 3)

Tivoli Network Manager 4.2 comes with several default views. A view appears in the network tree only if devices with the related technology are present in the network. Tivoli Network Manager groups show the greatest severity of event in each network view.

Two significant directories contain XML information.



Information

Dynamic view templates are found in the following directory:

```
/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/dynamictemplates
```

Autoprovisioned network views are stored in the following directory:

```
/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/autoprovision
```

- The XML files in the dynamic view templates directory control the creation of dynamic network views. These XML files define the domains and authorized users that are associated with each view.
- The auto-provisioned network views directory is checked every 60 seconds to determine whether Tivoli Network Manager created changes to any network views.

Tivoli Network Manager creates default views for all domains and assigns them to the **itnmadmin** user. Views are also created for the **itnmuser** user.

You can type characters and wildcards in a filter box to reduce the available views to those views that match the criteria you type.

To troubleshoot network views, do the following steps:

1. Review tree table logs and log settings:

- /opt/IBM/netcool/gui/precision_gui/profile/logs/tnm
- ncp_topoviz.0.log
- ncp_topoviz.0.trace

2. Review or change log settings in the `topoviz.properties` file. This file is located in the following directory:

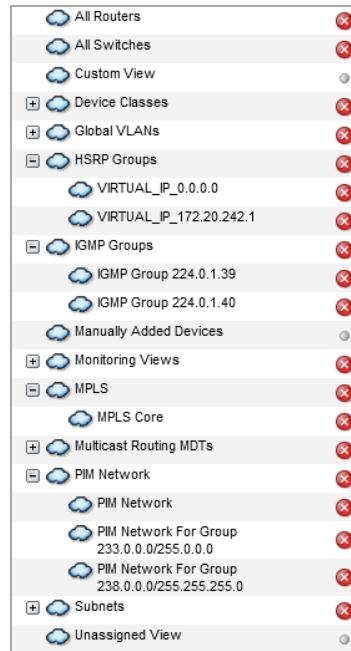
`/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm`

The network views in Tivoli Network Manager 4.2 have these features:

- Large networks render quickly. When discovery completes, Tivoli Network Manager begins caching network views and the status of devices in each view. If you look at a network view before this cached information is prepared, then it can take longer to render.
- The state of the tree is maintained in cache so that the tree loads more quickly within the same user session.
- You can sort, reorder columns, and hide columns within the network views.
- All status icons are hyperlinks that take you to event views.
- **Edit, copy/move, and delete** icons are found on the network view tree table toolbar.

Default network views (2 of 3)

- Custom network partition views
- Devices added with Topology Editor
- Views by device class
- Global VLANs
- Hot Standby Routing Protocol (HSRP) groups
- Multicast views
 - IGMP group views
 - PIM network views
- Subnets



© Copyright IBM Corporation 2017

Figure 5-37. Default network views (2 of 3)

Put views that you create under the **Custom View** portion of the tree.

Default network views (3 of 3)

Monitoring views

- Acknowledged alerts by severity category
- Unacknowledged alerts by severity category
- Chassis ping fail events
- Root cause events
- SNMP poll failures
- Recent interface ping fail events

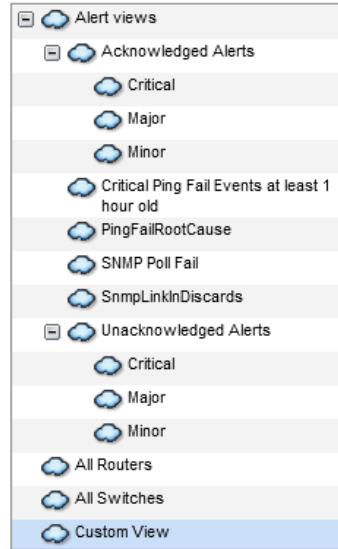
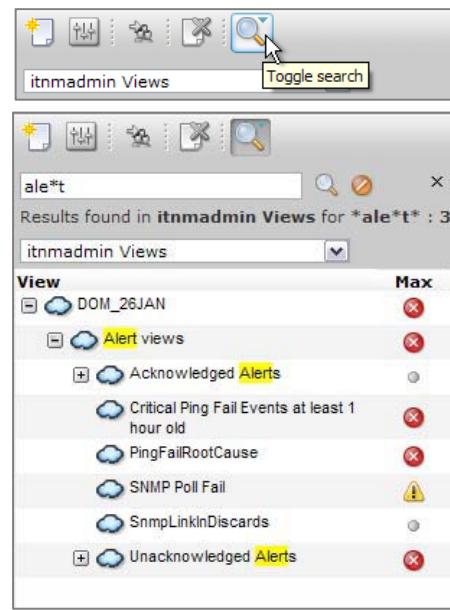


Figure 5-38. Default network views (3 of 3)

Tree table search

- Toggle search icon on the toolbar
 - Search phrase is highlighted in results
 - Search is not case-sensitive
 - Wildcards % or * match 0 or more characters and are supported at each end of the search phrase
- Parent and children views
 - If a parent node matches, all of its children are returned in the filtered tree
 - If any child view matches, the parent is expanded
- A search with no results returns a fully collapsed tree
- Search is available in the following views:
 - Network Views
 - Path View
 - Network Views tree in Poll Policy editor



[Topology visualization basics](#)

© Copyright IBM Corporation 2017

Figure 5-39. Tree table search

As you type each character into the filter box, the list of available views is reduced to those network views that match the characters you type. If a parent view matches the filter, then all of children of that view also are seen in the list of available views.

Tree table search gives the user the ability to search through the items in the tree.

- After you run a search, the user is shown a filtered tree, with matching items highlighted.
- The tree table is not updated. Updates to the tree are turned off while the search results are being shown.

Tree table search is available on the following tree tables:

- Network view trees
- Path view tree
- Network views tree in Poll Policy editor

Network view bookmarks

- Bookmarks enable users to select the network views they use the most and save those views in the **Bookmarks** tab
- When you view the Bookmarks tab, an event status refresh updates only the bookmarked views
 - Updating a smaller number of devices results in enhanced performance of the GUI

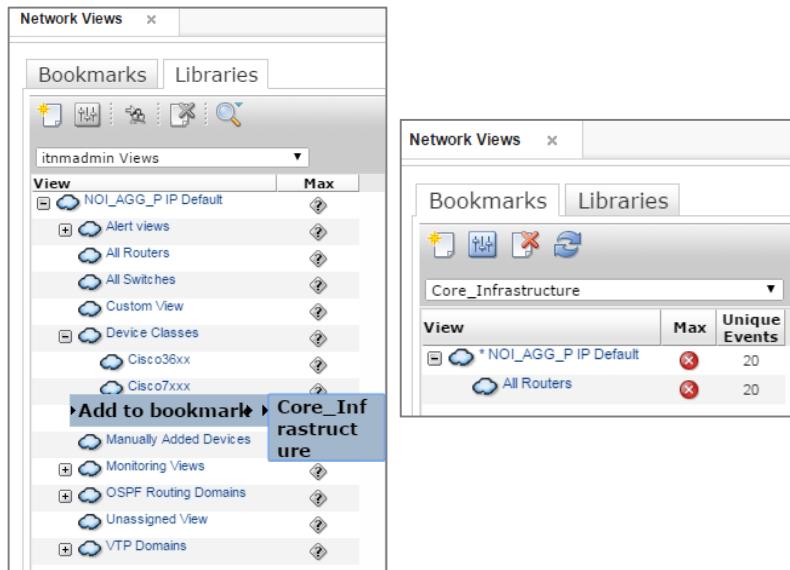


Figure 5-40. Network view bookmarks

Users can add specific network views to a bookmark.

- When IBM Tivoli Network Manager updates the event status on a bookmark view, the update is affecting fewer devices than when someone is trying to view the entire network view tree.
- The use of bookmarks result in enhanced performance of the GUI.
- In the preceding screen graphic, a bookmark has three network views. Event status changes update only these three network views as the operator is looking at this bookmark view.

When the user returns to the **Libraries** tab, event status updates must update all views. The use of the Libraries tab results in slower performance of the GUI.

The **Bookmarks** tab occurs first and is the default selection when a user opens network views.

Creating a container

1. Select **Network Views** from the menu
2. Click the **Libraries** tab
 - Click the **New View** icon
 - Select the user or group to which the new view applies
3. Specify the **Container** type to create a container to hold other views
4. In the **Map Icon** box, specify the graphic files to represent the objects that are inside another object
 - You can also select a **Tree Icon** and **Background image**
 - Click **OK**

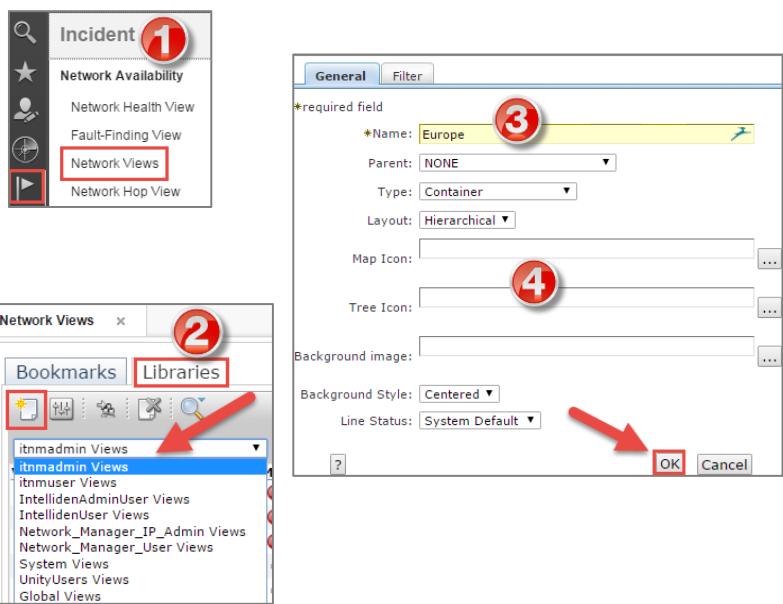
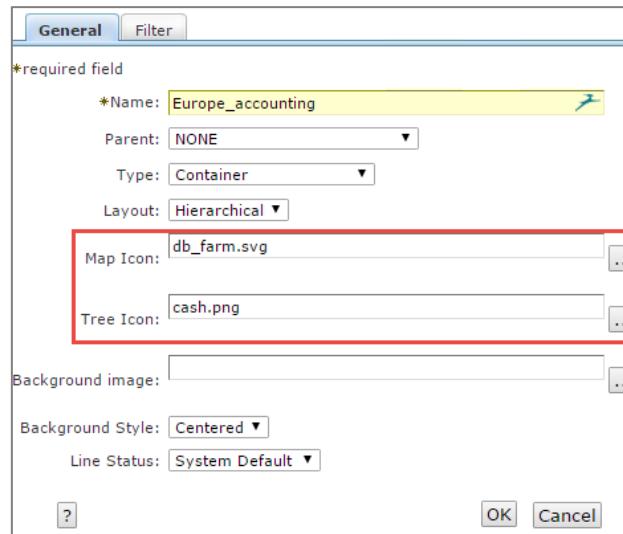


Figure 5-41. Creating a container

A **container** is an item that appears on the root of the network view tree under which you can place network views.

Customizable network view icons (1 of 2)

- Network views can be assigned icons in the tree, on the map, or both
- To enable viewing of the map icon, the associated view must be the child of another view



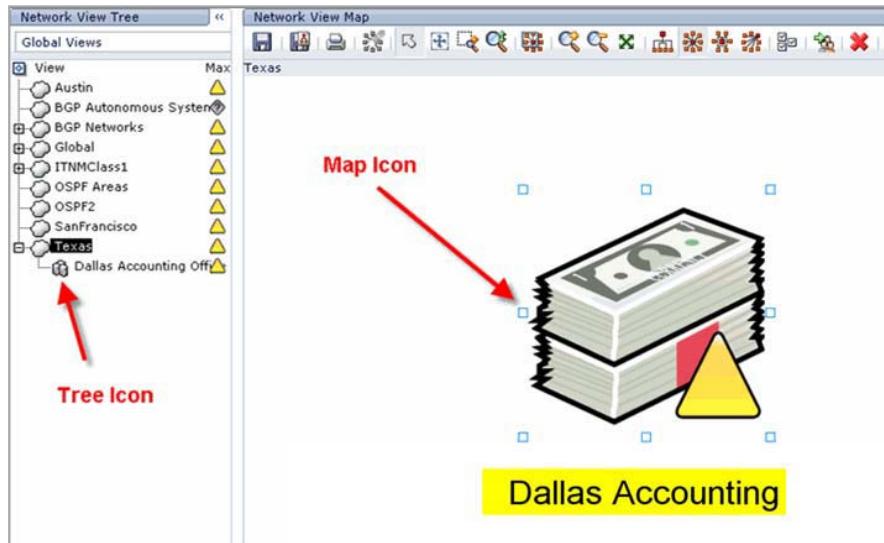
Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-42. Customizable network view icons (1 of 2)

You can assign icons to represent all network views except for items at the root of the tree.

Customizable network view icons (2 of 2)



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-43. Customizable network view icons (2 of 2)

Standard versus dynamic network views

- Standard views require manual explicit creation and deletion
 - User specifies filter criteria or selects from preset filters
 - User explicitly chooses from the resulting list certain views that then appear in the network partition views
 - If all devices in a view cease to exist, the view remains
 - View is explicitly created and must be explicitly deleted
- Dynamic views are automatically created and deleted
 - User picks from dynamic collection preset filters
 - All possible views that match the criteria are automatically created
 - Views are created and deleted automatically
 - If new devices match criteria, new views are automatically created
 - If there are devices in a view are deleted, the view is automatically deleted
 - Individual views cannot be manually deleted

Figure 5-44. Standard versus dynamic network views

When you create a network view, you can create your own filter with query language or select from a predefined filter. Predefined filters include collections, subnet views, and MPLS VPN views. Using predefined collection, subnet, and MPLS VPN views are more efficient than using a filtered view. Use predefined collections to create views whenever possible.

Dynamic views are automatically created and automatically deleted. Standard views are explicitly created and must be explicitly deleted.

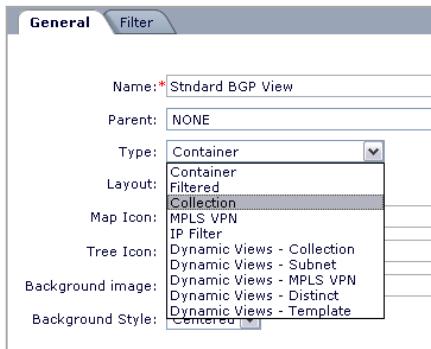
Standard network views

- When you create a standard network view, you explicitly define the values
 - In a collection view, you can see all views that matched the collection criteria and select which views you want
 - For example, you might create a network collection view of Border Gateway Protocol (BGP) autonomous systems (AS)
 - A list of existing autonomous systems is shown
 - You can then select from that list the ones you want in your network view
- After your selection is made, those views are persistent in the network views
- Standard views are not created or deleted automatically

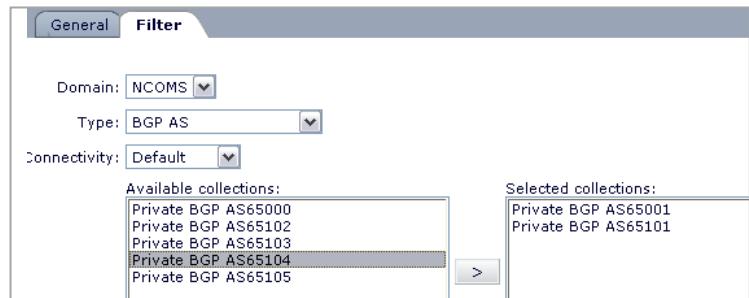
Figure 5-45. Standard network views

When you define filter criteria for a standard view, the GUI indicates what views can be created to match the filter criteria. You choose those views that you want to create.

Creating a standard network view



- The standard network view shows a list of available views that are based on criteria you select
- You pick only the views that you want
- The views are persistent even if they no longer contain devices



[Topology visualization basics](#)

© Copyright IBM Corporation 2017

Figure 5-46. Creating a standard network view

If a created network view no longer has any devices in the view, a standard network view persists even though it is empty. The empty view must be deleted manually.

Dynamic network views

- In a **Dynamic** network view, you can specify a collection or filter criteria
- Tivoli Network Manager then automatically creates every possible view that matches the specified criteria
 - For example, if you create a Dynamic Collection view of BGP autonomous systems, Network Manager automatically creates a view for every one it finds
 - Furthermore, if new BGP autonomous systems appear, views for them are created dynamically
 - If device groupings like BGP autonomous systems no longer contain entities, the view is dynamically deleted
 - After a discovery, if new devices are found that form new groupings, views are dynamically created
- The default views created during the installation are dynamic views

Figure 5-47. Dynamic network views

A number of default network views are created during the discovery process. These views are dynamic network views. If devices no longer exist in a view, the view is automatically deleted. If a new view can be created that matches the filter criteria of the dynamic network view, it is created automatically.

Creating a dynamic network view

- Creates all available views that match the selected criteria
- New devices that meet the collection criteria cause the creation of new views
- When no devices exist to match the view criteria, Tivoli Network Manager deletes the view

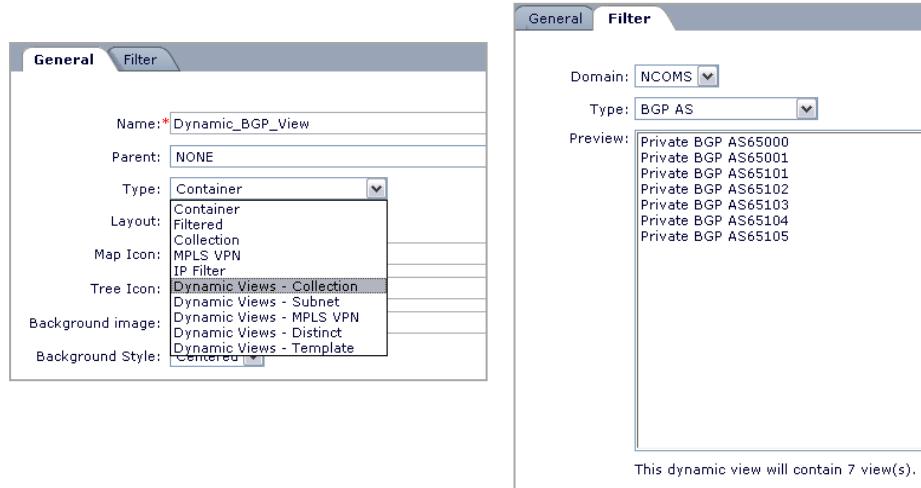


Figure 5-48. Creating a dynamic network view

In this example of creating a dynamic network view, Tivoli Network Manager creates seven network views automatically because that is how many views match the filter criteria. New views are created automatically when devices match the filter criteria and are not already in a network view.

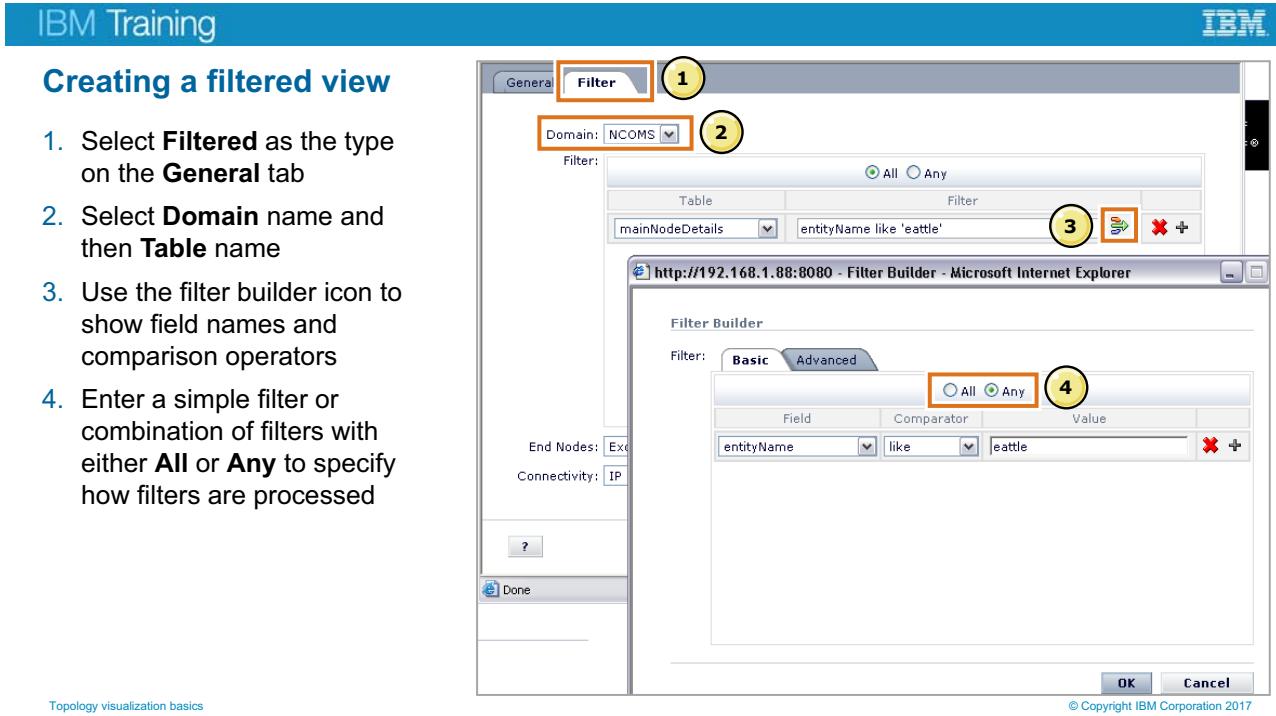
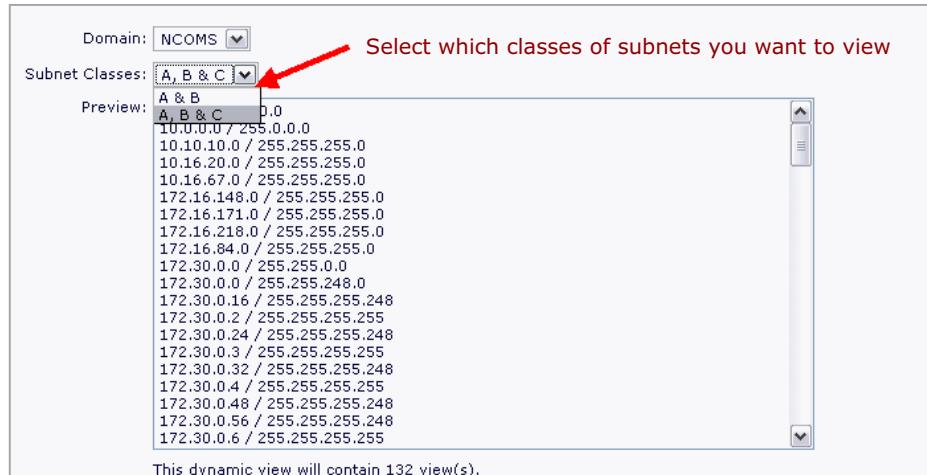


Figure 5-49. Creating a filtered view

Specifying **All** causes the filter to use a Boolean **AND** function. Specifying **Any** causes the filter to use a Boolean **OR** function.

Subnet view



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-50. Subnet view

It is possible to have subnet views that overlap one another due to variations in the subnet mask. When you create subnet views by selecting subnet classes, then a view is created for every possible subnet.

IP filter views (1 of 2)

Provide an easy way to create views that are based on IP address ranges

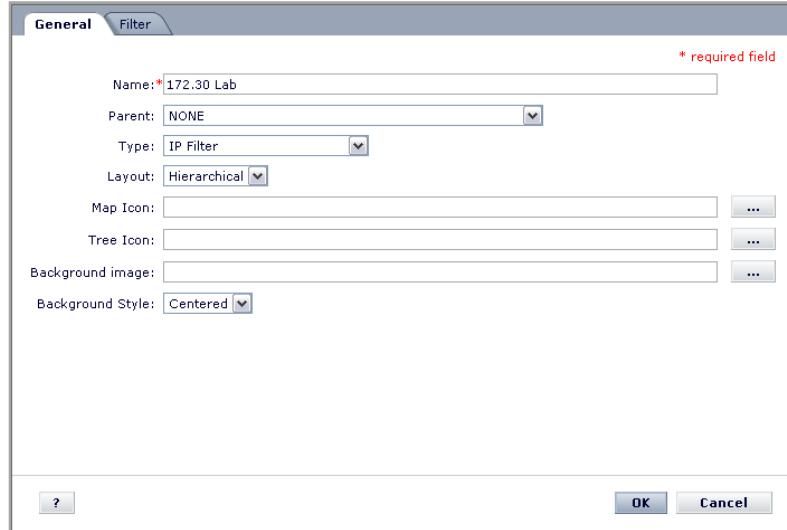
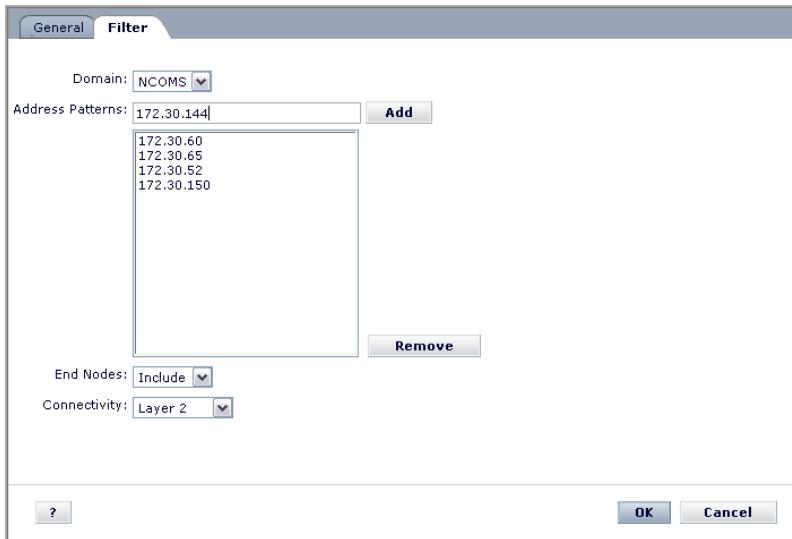


Figure 5-51. IP filter views (1 of 2)

IP filter views (2 of 2)



- Enter as many characters as are necessary to limit the results
- For example, 172.30.15 matches 172.30.15, 172.30.150 – 172.30.159
- To get only the 172.30.150 subnet, enter enough characters so only that subnet is in the results

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-52. IP filter views (2 of 2)

Remember to add enough characters in the **Address Patterns** box so that it matches only the networks for which you want to create views.

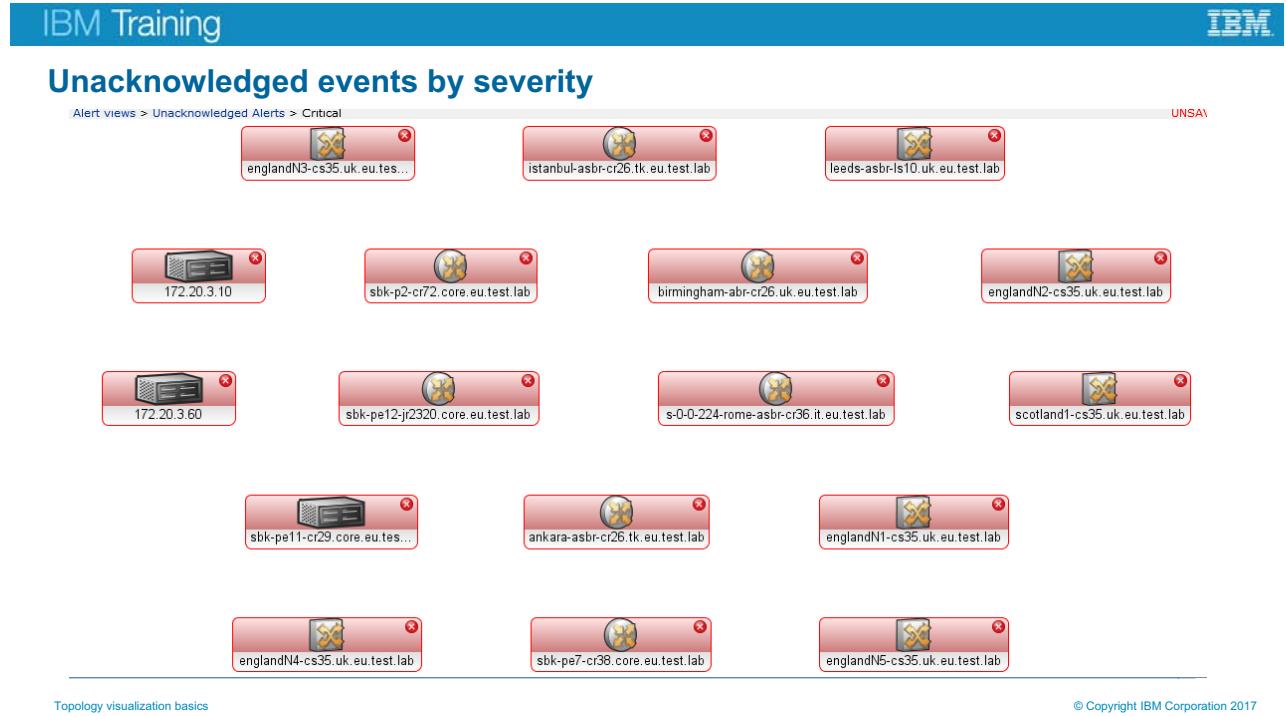


Figure 5-53. Unacknowledged events by severity

This view shows critical unacknowledged events.

IBM Training



All routers

If a switch icon appears in this view, the switch contains a routing blade

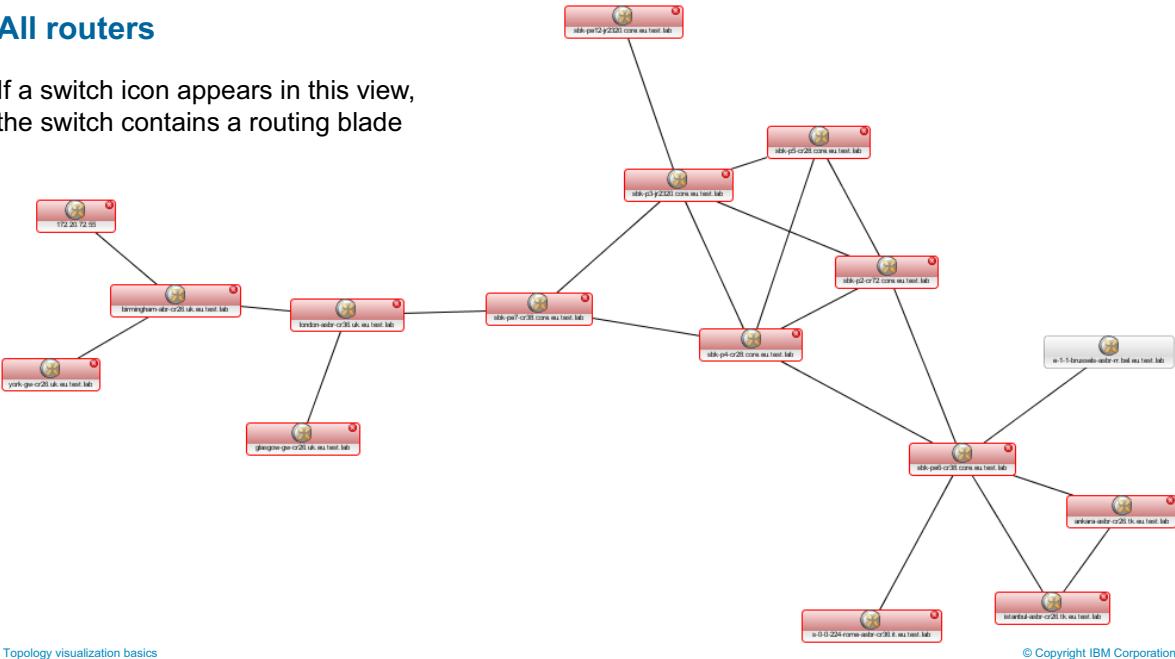


Figure 5-54. All routers

Some switches can also contain a routing card or blade. In this case, the switch can also provide routing capabilities and appears in the **All routers** view.

All switches

- This view shows all switches in the discovery
- A number in brackets next to a link indicates the number of physical links between two connected devices
- For example, the bracketed number seven [7] between two switches indicates that seven physical links exist between the two switches

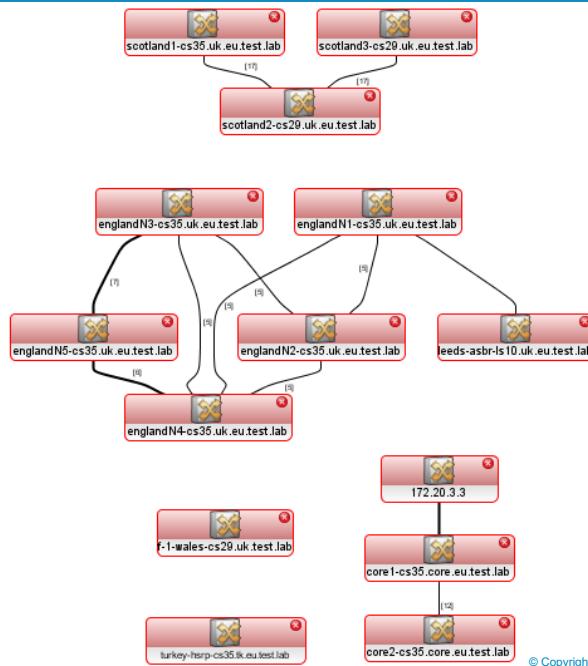
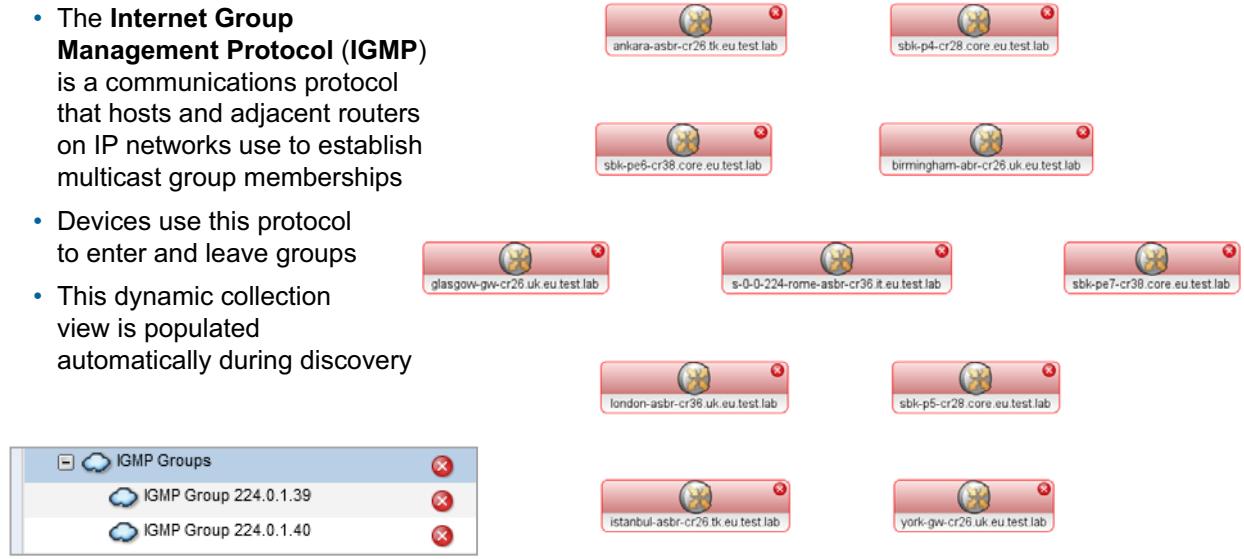


Figure 5-55. All switches

Switches sometimes have several connections between devices to aggregate bandwidth.

IGMP groups view

- The **Internet Group Management Protocol (IGMP)** is a communications protocol that hosts and adjacent routers on IP networks use to establish multicast group memberships
- Devices use this protocol to enter and leave groups
- This dynamic collection view is populated automatically during discovery



Topology visualization basics

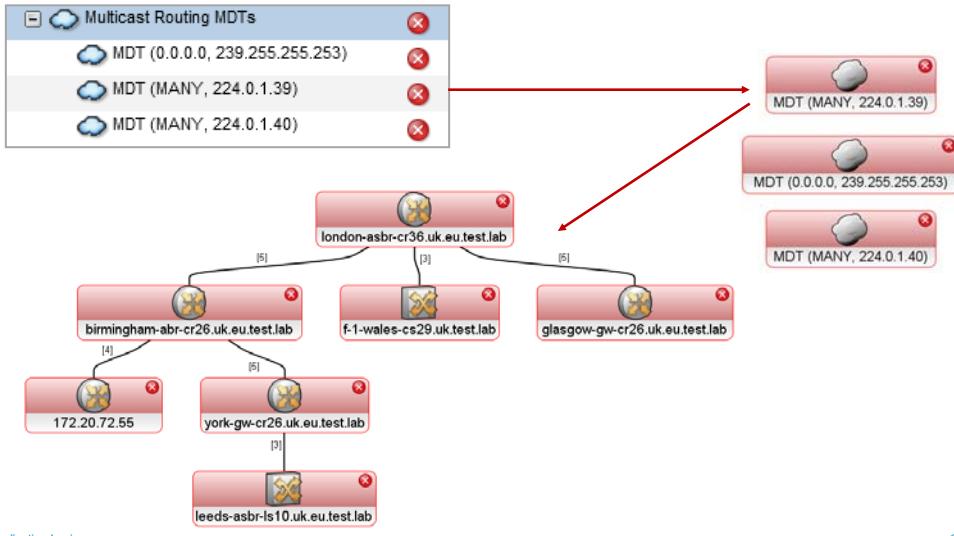
© Copyright IBM Corporation 2017

Figure 5-56. IGMP groups view

All multicast views are created automatically when discovery completes. Multicast network views cannot be created manually.

Multicast routing distribution trees

Views containing multicast distribution trees (MDTs) are created dynamically during discovery



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-57. Multicast routing distribution trees

Double-click multicast network icons with clouds to see the devices within the network view.

Protocol-independent multicast networks

Protocol-independent multicast (PIM) network views are created dynamically during discovery

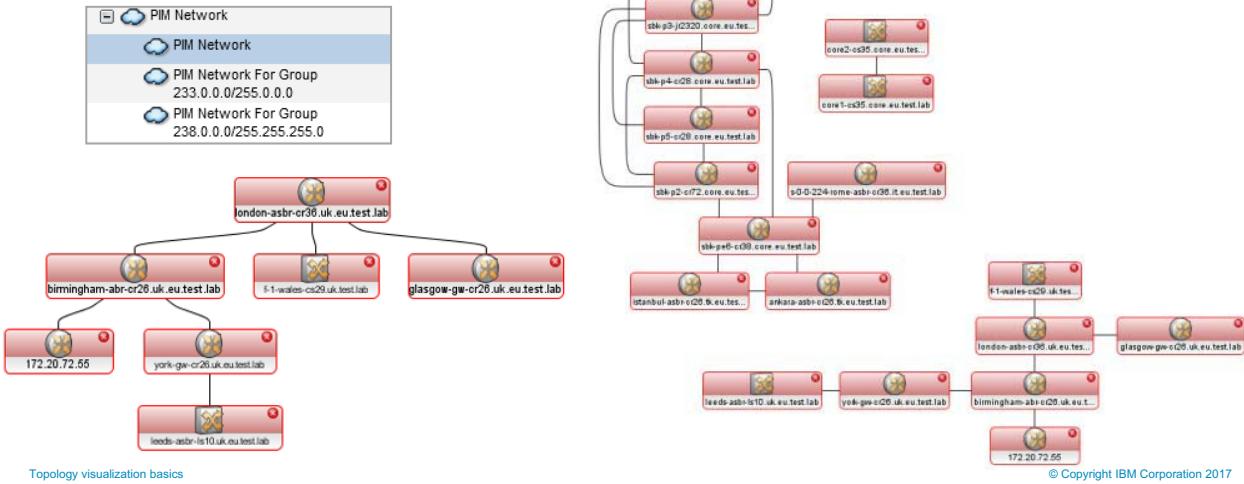
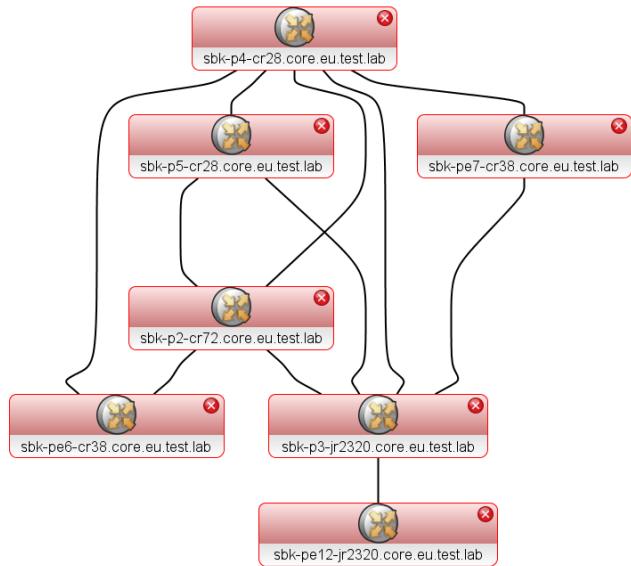


Figure 5-58. Protocol-independent multicast networks

Multicast group views cannot be created manually. These group views are created during the discovery process.

Virtual Private Label Switching

- Shows Virtual Private Label Switching (VPLS) core
- Shows VPLS VPNs if they exist during discovery



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-59. Virtual Private Label Switching

These dynamic network views are created automatically when discovery completes.

Manually created devices

- This dynamic view contains devices that are created with the Topology Editor
- View updates automatically as new devices are updated

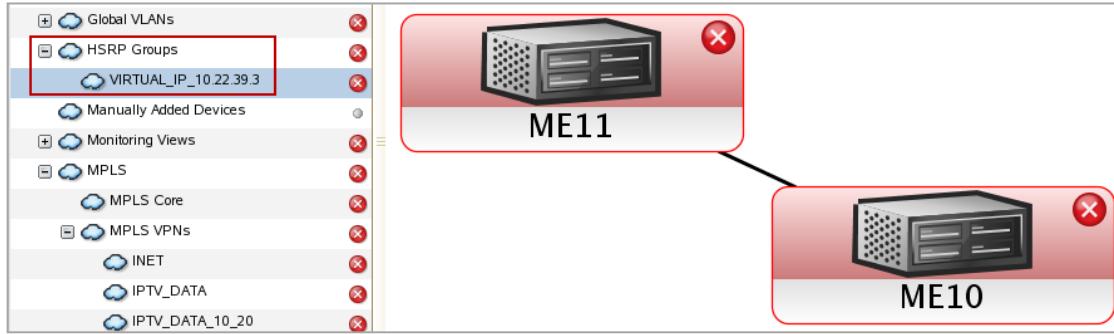


Figure 5-60. Manually created devices

Sometimes you need to create representations of devices or connections manually with the Topology Editor. Each manually created device or connection has an icon that indicates that it was manually created.

HSRP groups

- Hot Standby Routing Protocol (HSRP) groups show devices that are configured to use the same virtual IP address
- This protocol enables one of the devices to function as a hot standby backup unit in case the primary unit fails



Topology visualization basics

© Copyright IBM Corporation 2017

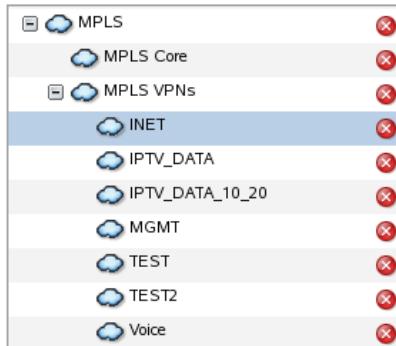
Figure 5-61. HSRP groups

Devices that are configured with HSRP share an IP address.

MPLS Core and VPNs

Network partition views created during discovery for:

- Multiprotocol Label Switching (MPLS) core network
- Each MPLS virtual private network (VPN)



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-62. MPLS Core and VPNs

In this example, the devices in the core of the network are grouped in a rectangle and designated as core devices. This designation makes it easier to see the function of devices in the MPLS VPN.

5.5. Link status and capacity

The Tivoli Network Manager default configuration draws simple black lines to indicate connectivity between devices. However, with some simple modifications, lines can have color to indicate link status and varying width to indicate bandwidth.

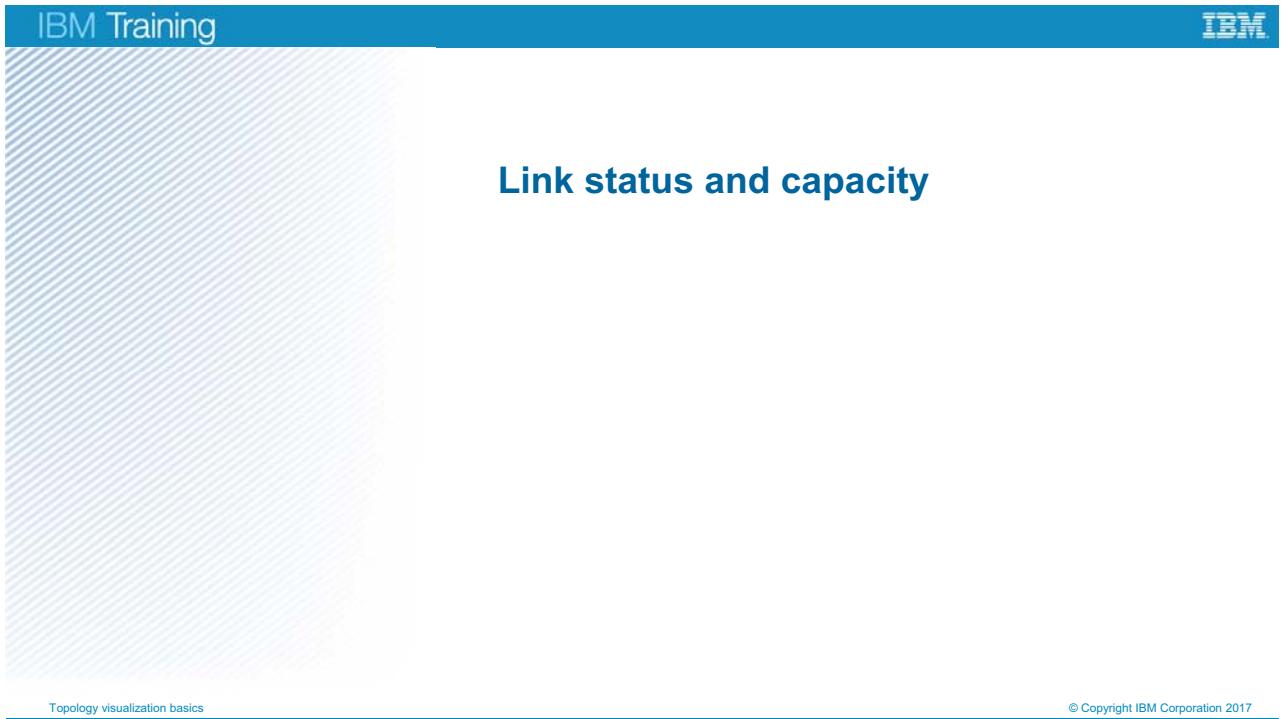


Figure 5-63. Link status and capacity

IBM Training



Link status and capacity

- By representing link status, NOC personnel can understand how critical the link problem is
- Connections with a number indicate the presence of link bundles
 - Right-clicking shows you the interface pairs that are part of the link bundles
 - This consolidation helps reduce onscreen clutter in the topology display

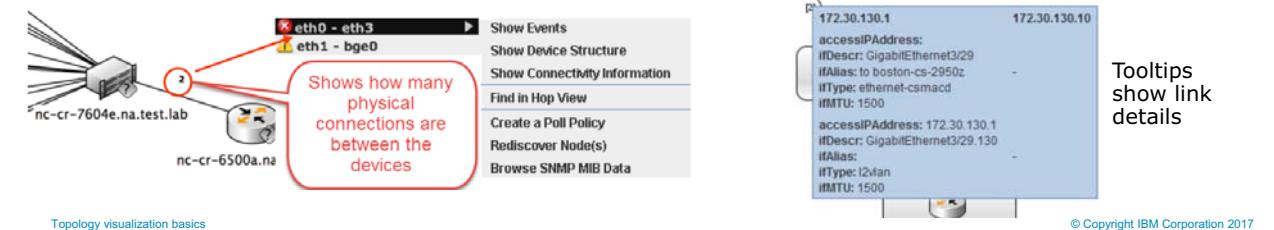


Figure 5-64. Link status and capacity

IBM Tivoli Network Manager 4.2 treats links between devices as interactive user interface objects.

- Link status is shown with colored lines, icons, or both colored lines and icons.
- You can right-click links to access link-related tools.
- Link bandwidth capacity is represented with lines of varying widths

The display of link status is supported in the following views:

- Layer 2 views show the status of all links between two devices. When multiple links exist between devices, right-clicking the link shows each linked pair of interfaces and the user can select from context-sensitive menu options.
- Path views.
- Layer 3 views are where the status color represents the events of greatest severity between connected interfaces.
- MPLS TE views show a status that is the summarization of all Layer 3 connections that support the tunnel.

Right-clicking a selected link presents menu options that provide the following functions:

- View events
- Rediscover devices
- View the device structure
- View the devices in a different view
- Browse SNMP MIB data
- Show detailed information about the connection

Hovering over a link with the mouse cursor shows a tooltip with information about the linked interfaces.

Multiple links between two devices are represented with a single line to reduce screen clutter. A number next to the link indicates the number of physical connections between the two devices. If the option to display link capacity is enabled and multiple connections exist between devices, the aggregate link speed is used for representing the capacity of the link between the devices.

Representing link status

- Tivoli Network Manager 4.2 represents links as objects
- The `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/status.properties` file controls the display of link status
- You can edit this file to specify which of three possible link status representations you want to use
 - none** Simple black lines
 - simple** Represent status with line color of highest severity status
 - detailed** Represent status with line color and severity icon
- You can specify the link status option for the Network View, the Hop View, and the Path View


```
status.netview.linestyle=simple
status.hopview.linestyle=detailed
status.pathview.linestyle=simple
```

Figure 5-65. Representing link status

The **Hop View** usually contains a few devices because you are seeing the specified device and all devices within a limited number of hops. When you enable the representation of link status, it does not add significant rendering time to a Hop View. A **Path View** also represents link status without a discernable performance difference.

But some **Network Views** can contain hundreds or thousands of device icons and links. With many links in a view, enabling the representation of link status can add to the time necessary to render a view. The views are first drawn in the normal manner (with no representation of link capacity or status). After this initial drawing, the links are updated to reflect status (if this option is enabled for the Network View).

Network maps are loaded in two phases:

- The topology loads immediately with data from the topology database. No link status is shown.
- Then, the Web GUI plug-in of Netcool Operations Insight dashboard downloads event information for each device and the link that is represented in the map from the Tivoli Netcool/OMNIbus ObjectServer. Topoviz updates the map to reflect link and device status.
 - The status update takes longer when many nodes and links are represented on the map.
 - To reduce time for status update, configure the display for fewer hops.
 - You can also limit link status representation to the Hop View and Path View, which both tend to have fewer devices than network views.

At the bottom of this **status.properties** file, you can see the default values:

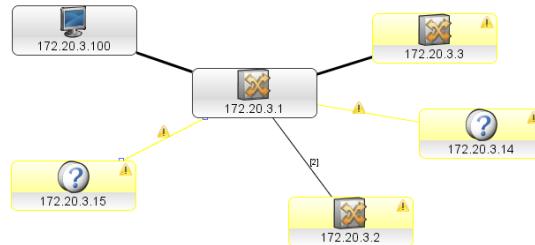
```
# Display status of connection between nodes
# possible values: none, simple, detailed, default is simple
status.netview.linestyle=simple
status.hopview.linestyle=simple
# Applies to both MPLS TE and IP Paths
status.pathview.linestyle=simple
```

Link status display

A colored line with or without a status icon represents the status of a link

Severity Name	Value	Icon
Critical	5	🔴
Major	4	🟠
Minor	3	🟡
Warning	2	🔵
Indeterminate	1	🟤
Clear	0	🟢

Status icons



Link Severity Styles

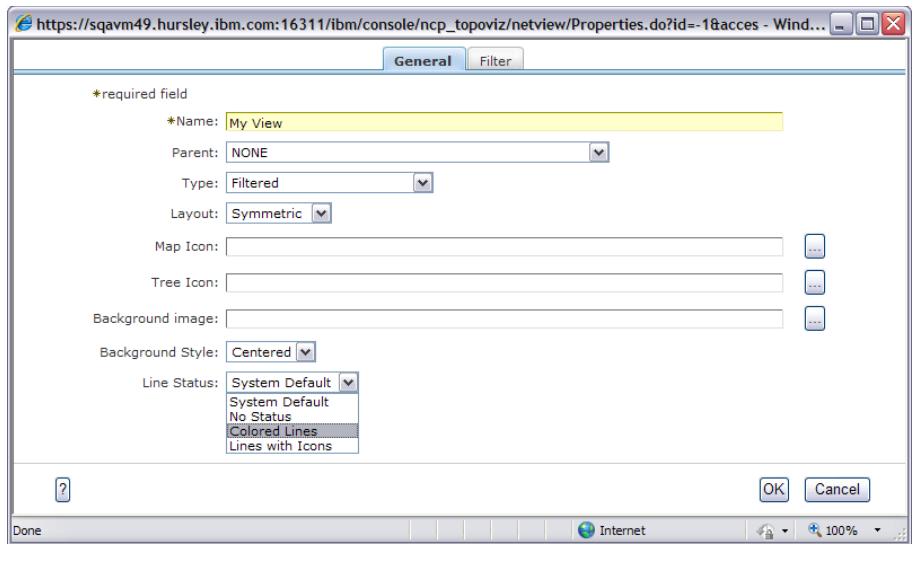


Figure 5-66. Link status display

Specify link status options on a network view

You can configure the status of links for each network view

- A setting for a particular network view overrides the configured system default setting for link status
- Select **Line Status** when you create or edit a network view



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-67. Specify link status options on a network view

When you create a network view, selecting **System Default** causes the view to use the global setting for line status. However, you can select **Colored Lines** or **Lines with Icons** for a particular network view. Selecting one of these values causes the created network view to use the selected setting instead of the system default setting.

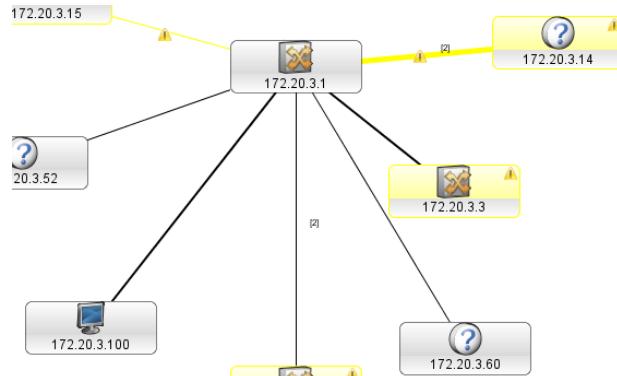
Representing link capacity

- `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/topoviz.properties` contains options for representing link capacity:
 - To enable link capacity representation, set the following values to true
 - To disable link capacity representation, set the following values to false
 - `topoviz.hopview.showlinkcapacity=true`
 - `topoviz.networkview.showlinkcapacity=true`
- You can specify up to five capacity thresholds each with a different line thickness
 - These pairs are called pipes
 - Each pipe has two entries in **topoviz.properties**
 - `topoviz.pipe.1.threshold=100M`
 - `topoviz.pipe.1.thickness=2`
 - `topoviz.pipe.2.threshold=2G`
 - `topoviz.pipe.2.thickness=4`

Figure 5-68. Representing link capacity

Link capacity

- Tivoli Network Manager can represent the relative capacity of links with lines of varying thickness
- Configured in the `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/topoviz.properties` file
- You can enable or disable the representation of link capacity independently for the **Hop View** and **Network Views**



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-69. Link capacity

The **Hop View** usually contains a few devices because you are seeing the specified device and all devices within a limited number of hops. When you enable the representation of link capacity, it does not add significant rendering time to a hop view. A **Path View** also represents capacity without a discernable performance difference.

But some **Network Views** can contain hundreds or thousands of device icons and links. With many links in a view, enabling the representation of link capacity can add to the time necessary to render a view. The views are first drawn in the normal manner (with no representation of link capacity or status). After this initial drawing, the links are updated to reflect capacity (if this option is enabled for the network view).

Network maps are loaded in two phases:

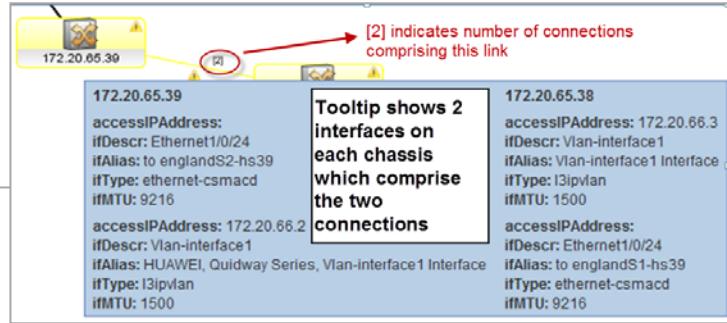
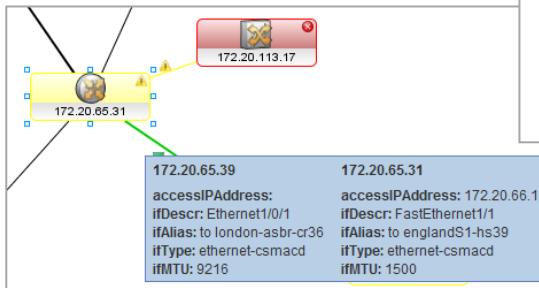
- The topology loads immediately with data from the topology database. No link status is shown.
- Then, the Web GUI plug-in of Netcool Operations Insight dashboard downloads event information for each device and the link that is represented in the map from the Tivoli Netcool/OMNIbus ObjectServer. Topoviz updates the map to reflect link and device status.
- You can minimize the time that it takes to load and update network maps.
 - The status update takes longer when many nodes and links are represented on the map. To reduce time for status update, configure the display for fewer hops.
 - You can also limit link status representation to the Hop View and Path View, which both tend to have fewer devices than network views.

The feature to represent links as objects records log messages to the common Topoviz log files:

- `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm/ncp_topoviz.0.log`
- `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm/ncp_topoviz.0.trace`

Link tooltips

- Hover the mouse over a link to open a tooltip for a configurable time period
 - The tooltip shows interface information about each end of the link
- Where a link represents multiple connections, the tooltip includes interface information about each end of the connection that comprises the link



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-70. Link tooltips

The tooltip default settings show the tooltip for only 3 seconds. If you did not have sufficient time to view the tooltip details, you can move your mouse off the icon or link and then hover over it again. This action causes the tooltip to reappear.

You can also configure the tooltip properties in the `topoviz.properties` file. This file is located in the following directory:

```
/opt/IBM/netcool/gui/precision_gui/profile/etc/trm
```

If you set either the `topoviz.tooltip.timefire` or the `topoviz.tooltip.timestay` property, you must set them both. For example, to have the tooltip show after hovering over an icon or a link for 1 second and then remain on-screen for 8 seconds, set these values in the `topoviz.properties` file:

```
# Allows tooltip timings to be customized. Values are in milliseconds.  
# Note: None or both values should be set.  
topoviz.tooltip.timefire=1000  
topoviz.tooltip.timestay=8000
```

Contextual link tools

Right-click a specific link to see the tools available for that link

- **Show Link Events** opens a new window that shows events that determine the link status
- **Manage or Unmanage**
- **Ping from the server**

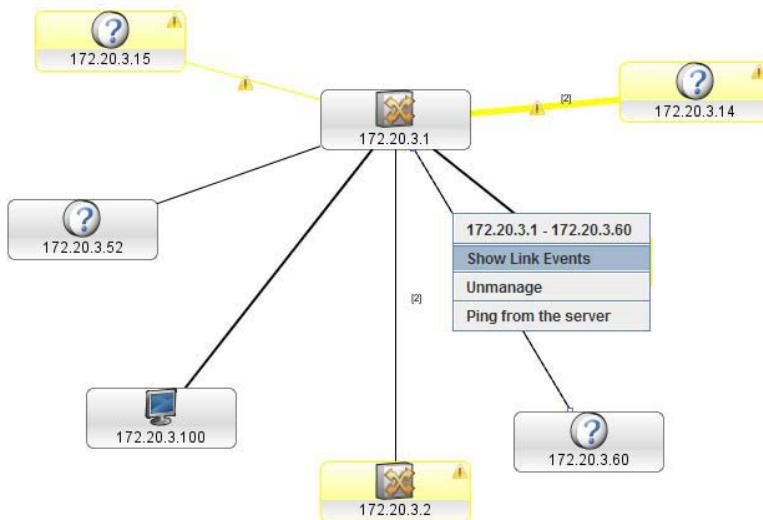


Figure 5-71. Contextual link tools

When you unmanage a link, Tivoli Network Manager unmanages the interfaces at both ends of that link. The **Ping from the server** tool causes the Netcool Operations Insight dashboard server to send an ICMP packet to the selected device. In another tool, you can select an option to **ping from this host**. This tool causes the user's workstation to send a ping to the selected device.

Contextual tools on link bundles

- Right-clicking a link that is composed of multiple connections shows a list of those connections
 - Each connection in the list shows the interface endpoint information and the status of each connection
 - From the list, you can click the connection that is the source of the network error
- The user selects the specific connection and is then presented with the list of contextual link tools

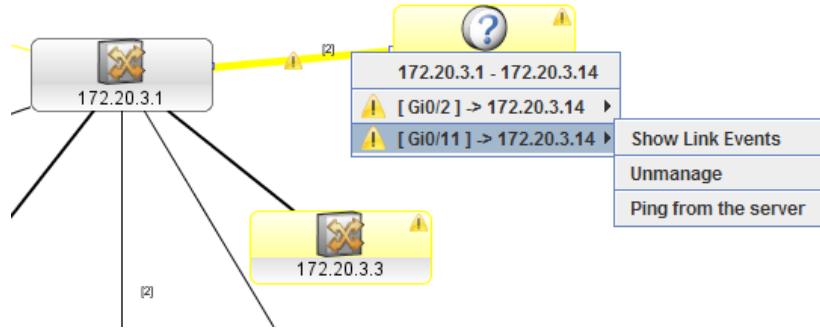


Figure 5-72. Contextual tools on link bundles

Topology visualization basics

© Copyright IBM Corporation 2017

Show link events tool

The **Show Link Events** tool opens a new browser window that contains the AEL showing only events that contribute to the link status

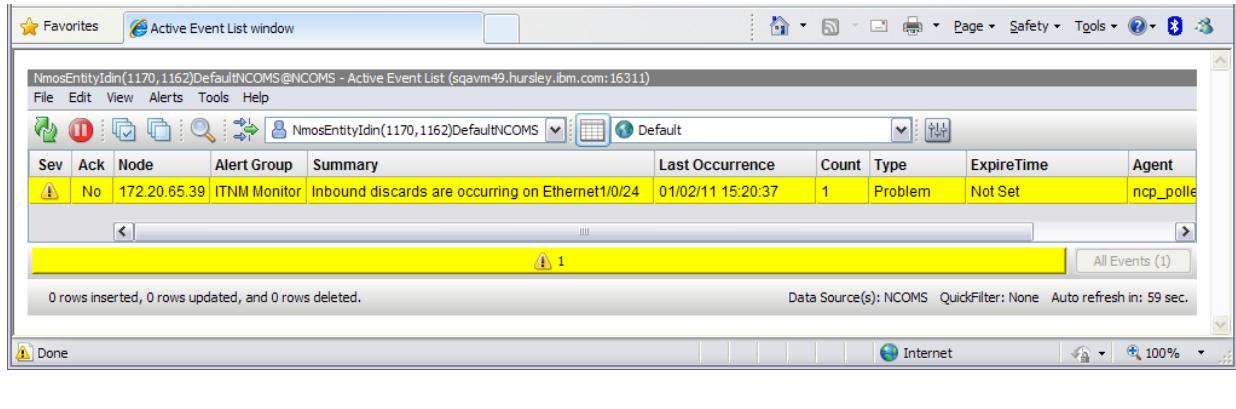


Figure 5-73. Show link events tool

When you use the **Show Link Events** tool, the events are launched in another tab.

Tabular layout option

- Selecting the **Table view** icon changes the view from device icons to a table
- The table view can be sorted
- Useful for viewing large numbers of devices



NOI_AGG_P IP Default > All Routers				
Display Name	IP Address	Class Name	Class Type	Managed State
BRU-PE-01	10.10.255.15	Cisco36xx	Router	Managed
WAS-Core-01	10.10.255.5	Cisco7xxx	Router	Managed
BRU-PE-02	10.10.255.16	Cisco36xx	Router	Managed
SYD-Core-01	10.10.255.6	Cisco7xxx	Router	Managed
BRU-PE-03	10.10.255.17	Cisco36xx	Router	Managed
BRU-SW-01	10.10.255.10	Cisco36xx	Router	Managed
BRU-Core-01	10.10.255.1	Cisco7xxx	Router	Managed
BRU-ACS-01	10.10.255.11	Cisco36xx	Router	Managed
PAR-Core-01	10.10.255.2	Cisco7xxx	Router	Managed
MOS-Core-01	10.10.255.7	Cisco7xxx	Router	Managed
BRU-NAS-01	10.10.255.13	Cisco36xx	Router	Managed
BRU-CPE-01	10.10.255.12	Cisco36xx	Router	Managed
LON-Core-01	10.10.255.3	Cisco7xxx	Router	Managed
BRU-NAS-02	10.10.255.14	Cisco36xx	Router	Managed
NYC-Core-01	10.10.255.4	Cisco7xxx	Router	Managed

Topology visualization basics

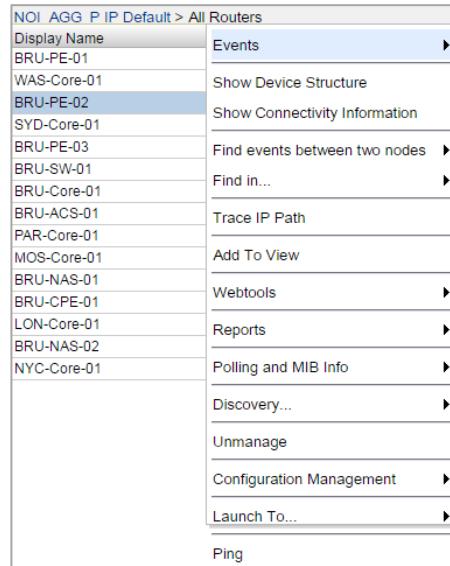
© Copyright IBM Corporation 2017

Figure 5-74. Tabular layout option

Select the tabular layout icon to switch from the graphical view to a table view.

Standard right-click options apply to tabular view

- Double-click a display name to open the Device Structure browser
- Right-click a device name to see the contextual menu



Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-75. Standard right-click options apply to tabular view

Tabular layout overview (1 of 2)

- Easier-to-comprehend data from network views that contain many devices
- The following data is shown for each entity:
 - Display label
 - IP address
 - Class name
 - Class type
 - Managed status
 - Maximum severity
- You can change the way data is displayed in the table
 - Lexical sorting on alphanumeric column data
 - Sort by severity of associated events
 - When you create or edit a network view, you can select **Tabular** as the default display option

Figure 5-76. Tabular layout overview (1 of 2)

Tabular layout overview (2 of 2)

- Maximum severity of events is updated in the tabular view on event notification
- Similar in appearance to AEL view, but rows are not colored
- Right-clicking a row of data in the table starts a contextual tool menu
- Double-clicking a row of data in the table starts a default tool

Figure 5-77. Tabular layout overview (2 of 2)

Limitations

- If you have a selected node in a tabular or map layout, that node is not selected when you switch to the other layout
- You can change the order of columns and their size, but you cannot save these options as user preferences
- The tabular view represents only chassis (**EntityType=1**)
 - It does not represent interfaces or connections
- You can print the table or save as an image

Figure 5-78. Limitations

Troubleshooting tabular views

- Icons that are associated with severities are defined in:
 - `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/status.properties`
- Useful log files include the following files:
 - `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm/ncp_topoviz.0.log` (can also be `*.1.log`)
 - `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm/ncp_topoviz.0.trace`
 - `/opt/IBM/JazzSM/profile/logs/server1/SystemErr.log`
 - `/opt/IBM/JazzSM/profile/logs/server1/SystemOut.log`
- Currently, it is not possible to export a table to a spreadsheet file
 - However, you can run a report on the network and save it in an Excel spreadsheet format

Figure 5-79. Troubleshooting tabular views

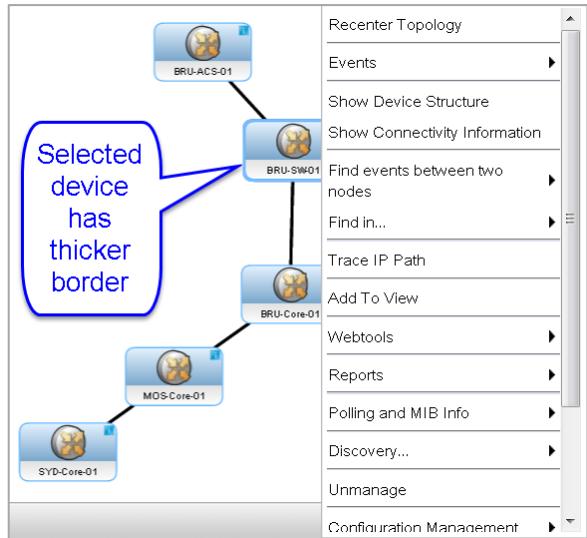
5.6. Tools

When you right-click a device in a topology view or an alarm in the event view, you can select from several tools.



Figure 5-80. Tools

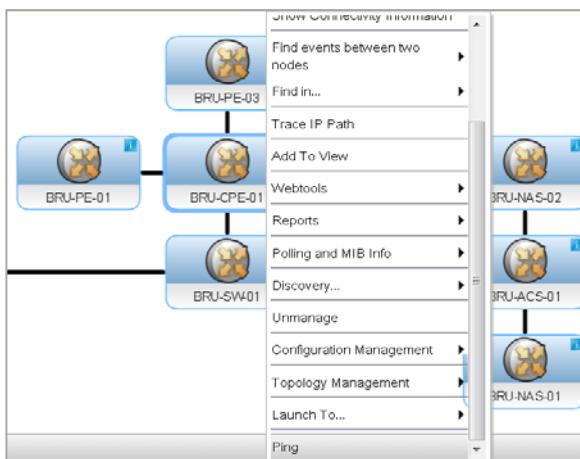
Topology visualization tools (1 of 3)



- **Recenter Topology** shows the map that is centered around the selected device
- **Events** option shows events for the selected device or starts **Event Analytics** for this device
- **Show Device Structure** starts Device Structure browser
- **Show Connectivity Information** shows a list of connections from the device
- **Find events between two nodes** shows connectivity-related events between two selected nodes

Figure 5-81. Topology visualization tools (1 of 3)

Topology visualization tools (2 of 3)



- **Find in** shows device in either Hop View or Network Views
- **Trace IP Path** starts the **Trace Network Path** tool that shows you the devices along a path between two selected devices
- **Add to View** adds a device to a custom network view
- **Webtools** can open a separate window with generic diagnostic tools and vendor-specific tools
- **Reports** menu option shows a list of reports that might include the device
 - Poll-based reports rely on stored polling data
- **Polling and MIB Info** shows historical poll data for the selected device, creates a poll policy, starts an SNMP MIB browser, or configures a *real-time* MIB graph for this device

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-82. Topology visualization tools (2 of 3)

You can add or edit the menu items in the menus that are displayed when you right-click a device or subnet from a topology map.

You can associate tools with menu items to enable network operators to right-click a device, link, or subnet and run a script or a third-party web application. Tools can also be invoked from components in the Device Structure browser, such as network ports, from the Tools menu. Any new menu items that you define appear after the default menu items.

To add or edit an item in a menu for a device, event, subnet, link, or component within the Device Structure browser, complete the following steps:

1. Edit an XML file in the `$NMGUI_HOME/profile/etc/tnm/menus` directory.
2. Configure the elements and attributes of the menu definition to define the name, filters, label, and other properties of the menu item. Use the reference information about the XML elements and attributes available for menu items and the example that is provided to help you define the menu item.
3. Add the menu item that you defined to the appropriate menu type in the `topoviz.properties` file in the `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm` directory.
 - Menus that are launched from devices: `topoviz.menu.device = menu-id`
 - Menus that are launched from subnets `topoviz.menu.subview = menu-id` where `menu-id` is an identifier that points to the top level of a menu hierarchy that is defined in an XML file in the `$NMGUI_HOME/profile/etc/tnm/menus` directory. Subnets require a separate menu because the tools that are run on subnets operate on all nodes that are contained within the subnet.

Topology visualization tools (3 of 3)

- | Discovery Status | | | | | | | | | |
|------------------|--------------|------------|---------------------|--------------------------|--------------------------|--------------------------|--------------------------|----------------------------|--|
| Entity Name | Access IP | Class Name | Interface Filtered? | Entity Created | Last Discovered | Last Modified | Last Reboot | Current | |
| BRU-PE-03 | 10.10.255.17 | Cisco36xx | No | 2016-08-11
22:13:44.0 | 2016-08-11
22:11:33.0 | 2016-08-11
22:13:44.0 | 2016-08-10
23:14:03.0 | 2016-08-23
19:07:33.255 | |
- Discovery** shows a discovery status summary or starts a rediscovery of the selected devices
 - Unmanage** disables monitoring and polling on the devices and can also disable root cause analysis for the selected devices
 - Configuration Management** starts the Tivoli Netcool Configuration Manager
 - Topology Management** tool is used to manually add devices or connections to your topology
 - Launch To** menu option can be used to launch other software
 - Ping** sends ICMP packet to the selected device
-
- The screenshot shows the Tivoli Netcool interface with several dropdown menus expanded:
- Discovery...**: Sub-options include "Show Discovery Overview" and "Rediscover Node(s)".
 - Configuration Management**: Sub-options include "ITNCM Reports", "ITNCM Wizard", "ITNCM Launch", and "Device Activity Sequence".
 - Launch To...**: Sub-options include "TADDM / CCMDB...", "View Details", and "View History".

[Topology visualization basics](#)

© Copyright IBM Corporation 2017

Figure 5-83. Topology visualization tools (3 of 3)

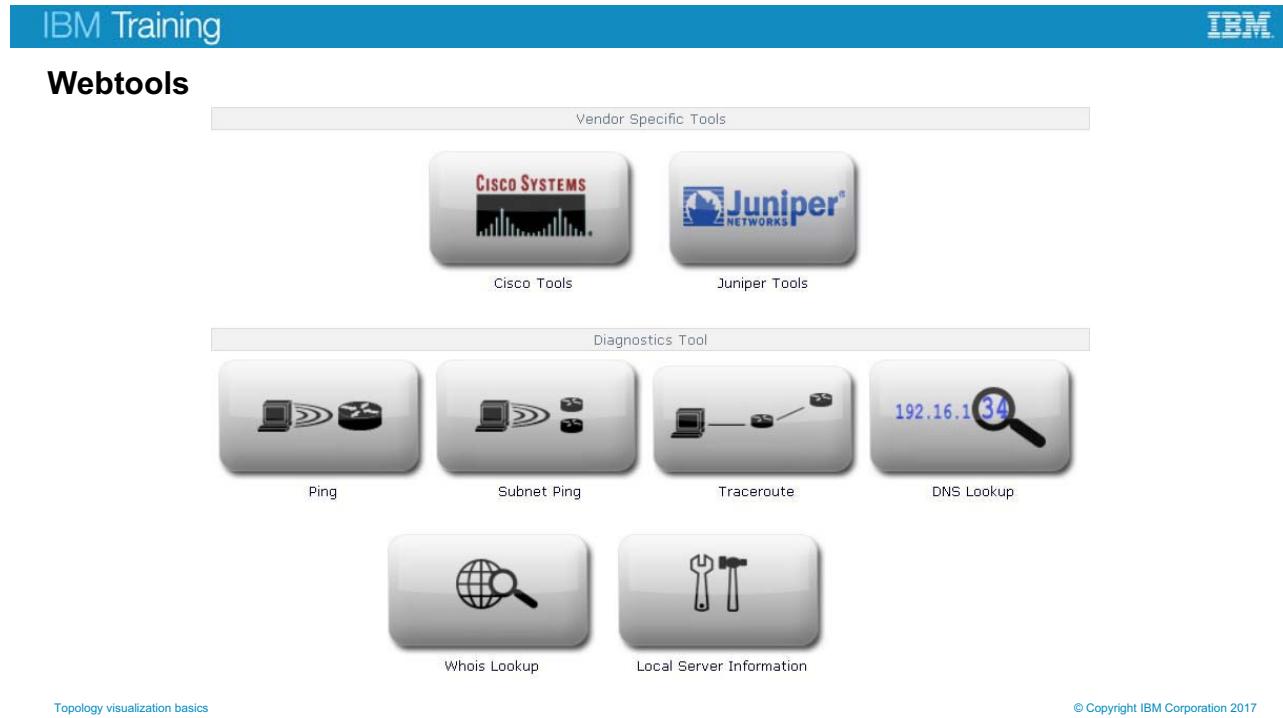


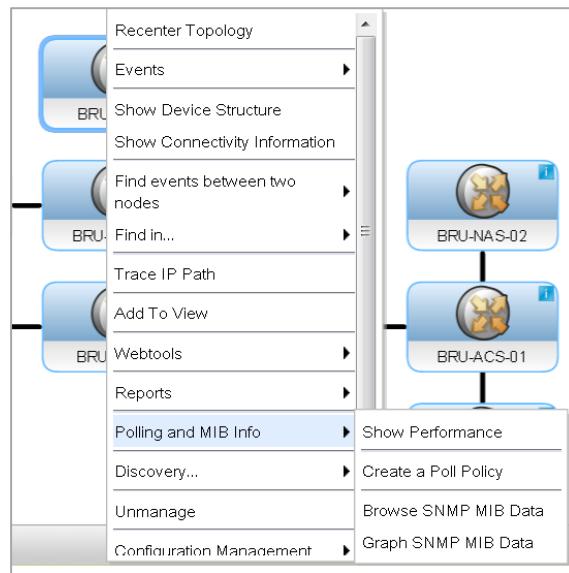
Figure 5-84. Webtools

The Webtools include vendor-specific tools and normal diagnostic tools.

Polling and MIB Info tool

This new tool has multiple options:

- **Show performance** shows historical poll data for selected devices
- **Create a poll policy** launches a poll policy configuration tool to create a poll policy specific to the selected devices
- **Browse SNMP MIB Data** starts an SNMP MIB browser on the selected device beginning in the MIB tree at the device ObjectID
- **Graph SNMP MIB Data** starts the real-time SNMP MIB Grapher utility that you can use to configure and show values of up to two MIB values simultaneously in real time



[Topology visualization basics](#)

© Copyright IBM Corporation 2017

Figure 5-85. Polling and MIB Info tool

Browse SNMP MIB data

The screenshot shows the 'Browse Host MIB' window with the following details:

- MIB Tree:** The left pane displays the MIB tree under the domain 'NCOMS'. A red arrow points from step 4 to the 'Method' dropdown menu at the top right of the main panel.
- Query Results:** The right pane shows the 'SNMP Query Results' table with columns 'Name' and 'Value'. The method selected is 'Walk'.
- Information Panel:** At the bottom, there is a panel titled 'MIB Variable Information' showing details for the selected MIB module:

Name	juniperMIB
OID	1.3.6.1.4.1.2636
Type	
Module	1.3.6.1.4.1.2636

Instructions:

1. Select host IP address
2. Select part of MIB tree or OID for query
3. Select Walk, Get, Get Next, or Get Table method
4. Click green arrow to start the query

Results are shown in the window

Figure 5-86. Browse SNMP MIB data

Ping tool

The **Ping** tool shows results of pinging selected device from the DASH host in a browser window

The screenshot shows two windows side-by-side. On the left is a list of network devices with their IP addresses. On the right is a ping result window.

Display Name	IP Ad	Actions
BRU-PE-01	10.10	Events ▾
WAS-Core-01	10.10	Show Device Structure
BRU-PE-02	10.10	Show Connectivity Information
SYD-Core-01	10.10	Find events between two nodes ▾
BRU-PE-03	10.10	Find in... ▾
BRU-SW-01	10.10	Trace IP Path
BRU-Core-01	10.10	Add To View
BRU-ACS-01	10.10	Webtools ▾
PAR-Core-01	10.10	Reports ▾
MOS-Core-01	10.10	Polling and MIB Info ▾
BRU-NAS-01	10.10	Discovery... ▾
BRU-CPE-01	10.10	Unmanage
LON-Core-01	10.10	Configuration Management ▾
BRU-NAS-02	10.10	Launch To... ▾
NYC-Core-01	10.10	Ping

Network Manager Webtool

Result of pinging specified target
2016-08-19T20:33:49

BRU-Core-01 [ip: 10.10.255.1] is alive (packet return time: 1239.85 ms)

Figure 5-87. Ping tool

© Copyright IBM Corporation 2017

Exercise: Topology visualization basics

Topology visualization basics

© Copyright IBM Corporation 2017

Figure 5-88. Exercise: Topology visualization basics

Complete the exercise for this unit.

Review questions

1. What are the three options for showing link status?

2. Which type of network view is automatically populated and is automatically deleted when no devices exist in that view?

3. Can MPLS views be created manually?

Figure 5-89. Review questions

Write your answers here:

Unit summary

- Use the Hop View
- Use Network Views
- Search for network devices
- Display link status and indicate bandwidth
- Create Dynamic Network Partition views
- Create an Internet Protocol (IP) filtered view

Figure 5-90. Unit summary

Review answers

1. What are the three options for showing link status?
 - Simple view
 - Colored line
 - Detailed view (colored line with status icon)

2. Which type of network view is automatically populated and is automatically deleted when no devices exist in that view?
 - A dynamic view

3. Can MPLS views be created manually?
 - **MPLS views cannot be created manually. They are created after network discovery completes.**

Figure 5-91. Review answers

Unit 6. Advanced visualization

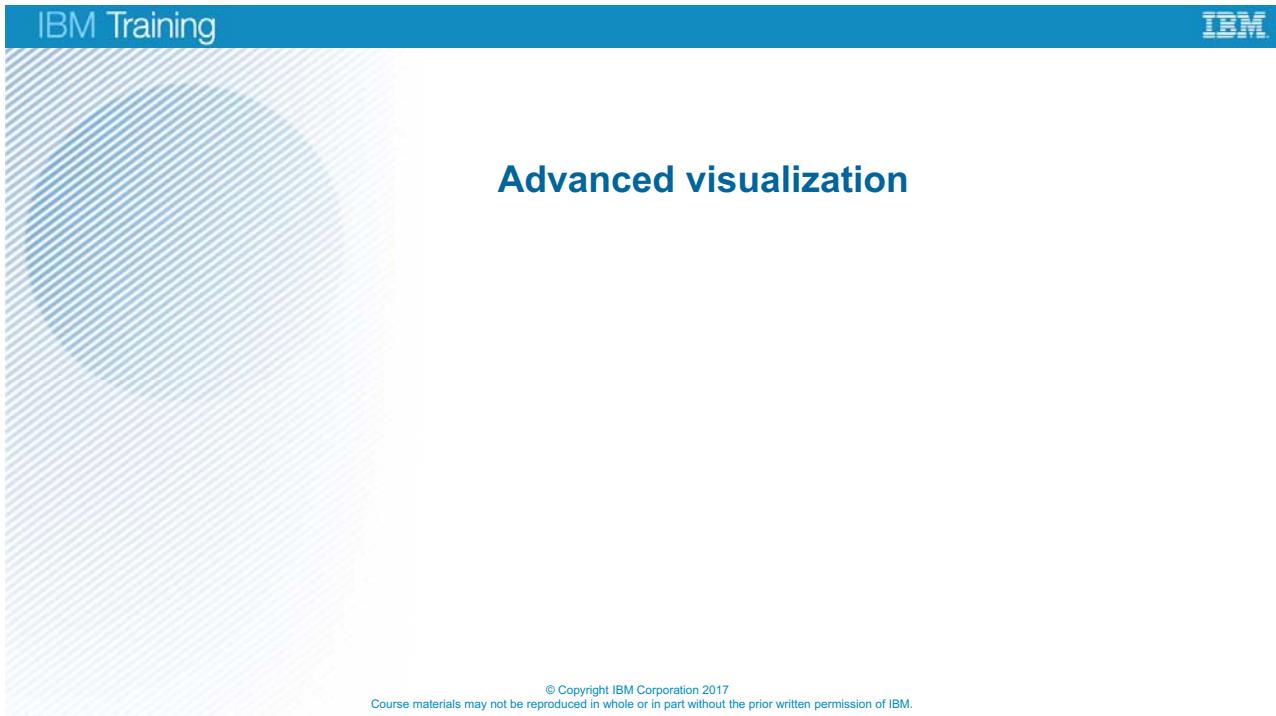


Figure 6-1. Advanced visualization

Estimated time

01:30

Overview

Tivoli Network Manager includes several specialized views to provide more detailed information for diagnostic purposes. These views include Path Views, the Device Structure browser, and the SNMP MIB browser. A Topology Editor enables you to manually add devices and connects that you were not able to discover due to access restrictions. A new feature of Tivoli Network Manager 4.x is the cross-domain network view. This network view contains devices from more than one domain in the same network map. This unit introduces these tools and how you can use them to enhance your understanding of your network.

How you will check your progress

- Complete checkpoint questions
- Complete student exercises

Unit objectives

- Launch an IPv4 Path View
- Use the Topology Editor to manually add devices and connections
- Use the Device Structure browser
- Create a cross-domain network view

Figure 6-2. Unit objectives

6.1. Path views

Path views have two types. The IPv4 path view shows that you selected beginning and end points and all devices through a certain gateway between those end points. The MPLS Traffic Engineer (TE) path view shows you devices along a designated traffic engineering tunnel for an MPLS network.

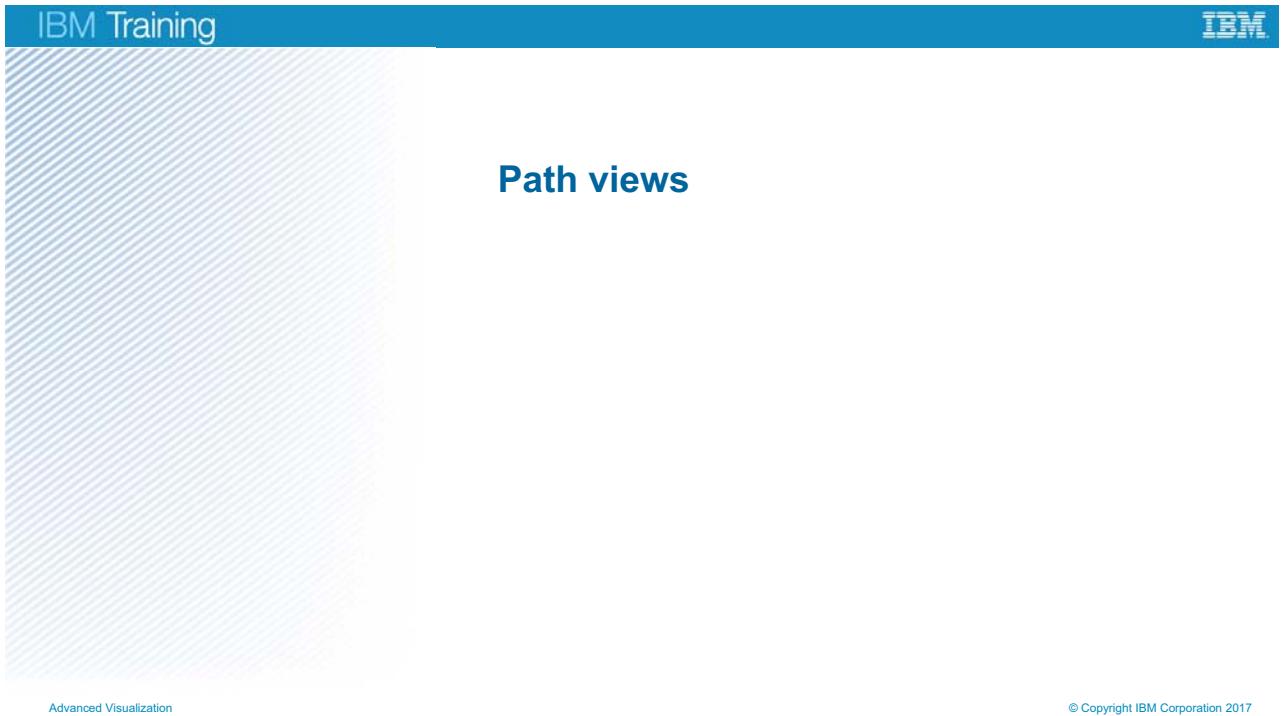


Figure 6-3. Path views

Path views overview

Tivoli Network Manager 4.2 has two types of path views:

- System paths are created automatically during discovery
 - Multi-protocol Label Switching Traffic Engineering (MPLS TE)
 - Virtual circuits
- User-created paths
 - Users can create on-demand IPv4 path views between any two devices in the discovery
 - By specifying different gateway devices between the path endpoint, you can see different routes between the devices
 - User paths can be created from: Network views, Network Hop View, Structure browser, Event Viewer, Active Event List

Current event status information for devices on the path is shown

- Right-click tools can be run from entities within a path view
- A path view administration window shows details of the path

Figure 6-4. Path views overview

The **Path View** helps you to do the following tasks:

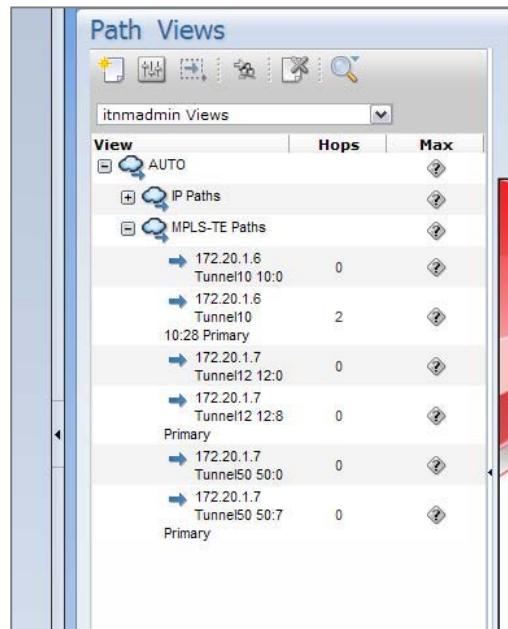
- Monitor, visualize, and verify complex network paths
- Discover network paths on demand by initiating a trace path with an intuitive right-click on selected network devices
- Aid operators in identifying the point along a network path that is subject to error conditions, and remediating the problem

Network engineers can use network path information to verify network configuration in complex IP networks.

- User-defined paths are not automatically retraced.
 - The retrieved path can vary depending on network routing behavior at the time of execution.
 - The path needs to be manually retraced to keep it up-to-date with the newest network routing information.
- Users can create only IPv4 path views.

MPLS TE paths

- Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) paths are generated during discovery
 - Only if they exist in the discovered network
 - Cannot be created manually with the path GUI
- Available MPLS TE paths views are listed in the navigation tree for **Path Views**



Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-5. MPLS TE paths

User-defined paths (1 of 2)

With path tools, you can do these tasks:

- Show the path between two devices
- Edit a path
- Retrace a path (run the tool again without changing any parameters)
- Delete a path

Tool options

- Create a path between devices through a different gateway
- Determine the return path for a defined path
- Do an out-of-band trace for a defined path
 - This procedure forces the use of discovered access IP addresses if available
- Ping verify each hop of the defined path to verify that each hop is reachable

Figure 6-6. User-defined paths (1 of 2)

Create a path view from a Hop or Network View

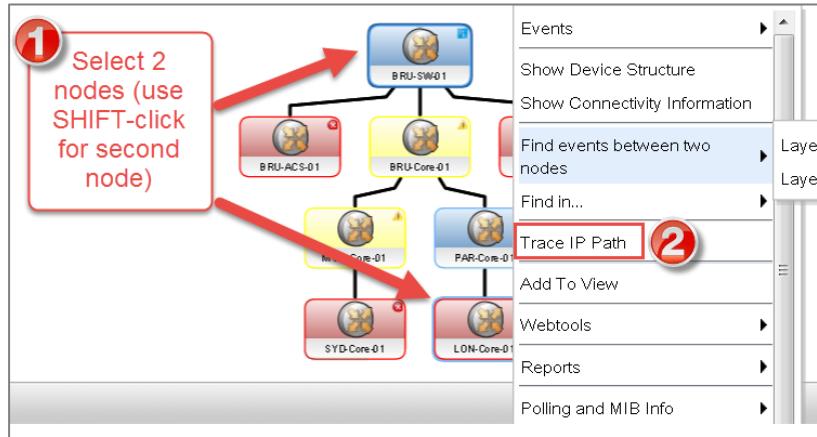


Figure 6-7. Create a path view from a Hop or Network View

Create a Path View from the menu

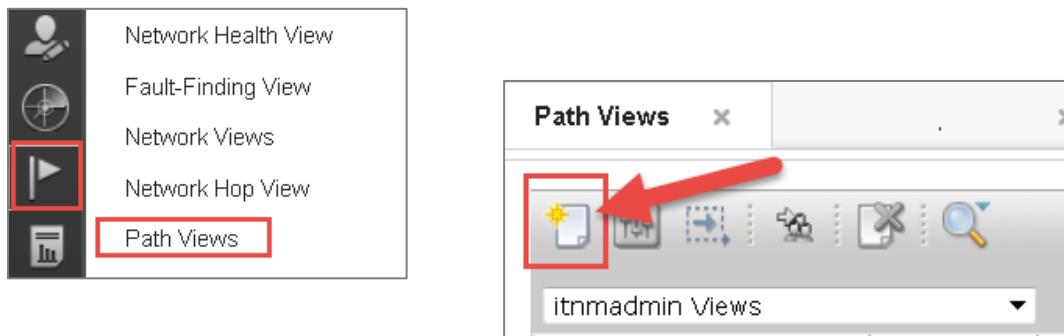


Figure 6-8. Create a Path View from the menu

Specify nodes and gateway

- If you selected two nodes in a Hop View or Network View, the selected nodes appear in the **Start** and **End** fields
 - Otherwise, enter them manually
 - (You can click **Browse** to search for a node)

The screenshot shows the 'Trace Network Path' dialog box. It has a 'Domain' dropdown set to 'NOI_AGG_P'. Under 'Name', there is a text input field. Below it, 'Start' is set to 'BRU-SW-01' with a 'Browse...' button, and 'End' is set to 'LON-Core-01' with a 'Browse...' button. A 'Via' input field is also present with a 'Browse...' button. At the bottom, there's an 'Options' section with three checkboxes: 'Get Return Path', 'Perform Out-Of-Band Trace', and 'Ping Verify Each Hop'.

- If you want to specify the gateway between two devices, enter that information into the **Via** field

The screenshot shows the same 'Trace Network Path' dialog box but with different values. The 'Name' field contains 'BrusselsToLondon'. The 'Start' field is 'BRU-ACS-01' with a 'Browse...' button. The 'End' field is 'LON-Core-01' with a 'Browse...' button and a 'Swap' button. The 'Via' field is 'BRU-Core-01' with a 'Browse...' button. In the 'Options' section, the 'Get Return Path' checkbox is checked. The copyright notice '© Copyright IBM Corporation 2017' is visible at the bottom right.

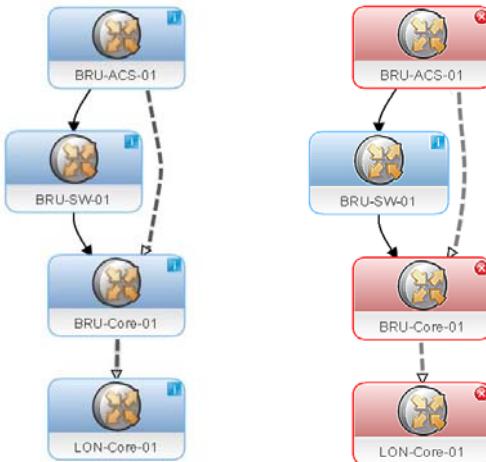
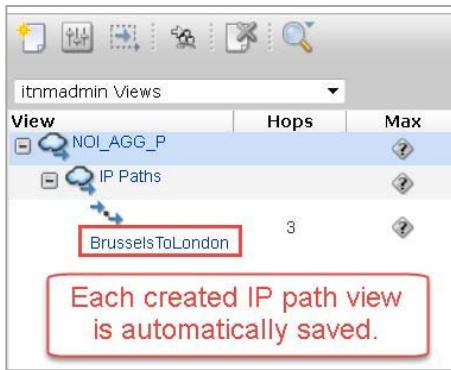
Figure 6-9. Specify nodes and gateway

The following options are available:

- Browse** uses the Entity Search to help you specify the **Start**, **End**, and **Via** nodes.
- Swap** swaps the start and end nodes.
- Via** and **Get Return Path** are mutually exclusive options. The **Via** option forces the path trace to go through the specified node. When your network is configured to use one path outbound and another path inbound, select the **Get Return Path** option to see the connectivity in each direction.
- Save and Trace** creates the path that is then shown in the GUI.

User-defined paths (2 of 2)

- The path view updates icon background colors as event severity changes
- The broken line shows the selected start, end, and gateway devices



Dashed line shows the values you selected. Solid line shows path trace.

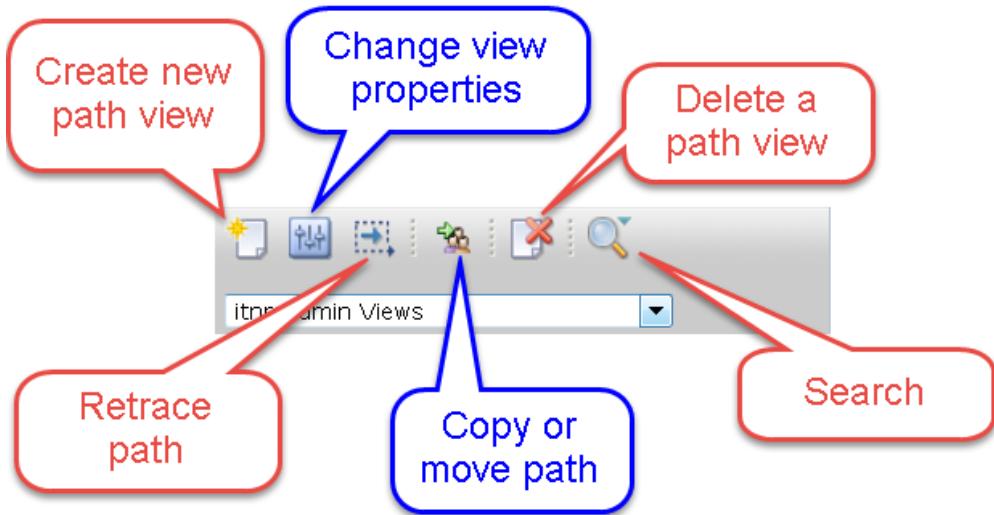
Figure 6-10. User-defined paths (2 of 2)

The broken line shows the selected start and end points and the specified gateway.

Also, the dashed line can change colors when service-affecting events (SAE) occur along the path.

In some cases, when you create a user-defined path, it is not possible to trace the full path between start and end devices. This problem can occur when path trace prerequisites are not met. For example, one of the devices along the path might not provide SNMP access. Another reason that it might not be possible to trace the full path is that one of the devices along the path does not provide routing information. If Network Manager is unable to trace a full path, then it creates a partial path that is made up of a subset of the nodes along the path. Partial paths are highlighted in the Path View Administration GUI with a warning icon in the Trace Status column.

Path View administration



Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-11. Path View administration

6.2. Topology editing tools

At times, you cannot discover a portion of your network. For example, you might not have access rights to a firewall or to intervening third-party networks that connect your location to another location. If you need to add only a few icons to represent these devices, or a few connections to build links between discovered network devices, you can use the Topology Editor to do so with a simple interface.



Figure 6-12. *Topology editing tools*

Graphical Topology Editor

- This feature provides a GUI tool that you can use to manually do the following tasks:
 - Add nodes to a network topology
 - Delete nodes from a network topology
 - Add links between existing nodes in a network topology
 - Delete links from a network topology
- You can visually identify manually created nodes and links
- You can have a specific view that shows only the manually created objects

Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-13. Graphical Topology Editor

The purpose of the Topology Editor is to provide the Tivoli Network Manager network administrator with a GUI allowing the augmentation of discovered network topologies with manually added nodes and links. This GUI also supports the removal of links and nodes. In some environments, network devices and links are not automatically discoverable. Network administrators might also want to include planned devices in their topology maps. A complete network topology model provides end-to-end visibility of the network infrastructure for network operations, and powers the root cause analysis capabilities of Tivoli Network Manager IP.

The functionality that is provided by this feature comprises the following use cases:

- Allow the manual addition of nodes to a network topology
- Allow the manual deletion of nodes from a network topology
- Allow the manual addition of links between existing nodes in a network topology
- Allow the manual deletion of links from a network topology

Users are able to visually identify manually created nodes and links by locating the special small icon that indicates manual creation. One network view shows all manually created device icons.

You can also augment the topology by using the **PresetLayer.stch** stitcher.



Important

This feature updates Network Manager's network topology model only. IBM Tivoli Netcool Configuration Manager can be used for configuring network devices.

Use cases for the Topology Editor

Sometimes you need to represent a device in the topology that Tivoli Network Manager did not discover

Tivoli Network Manager might not discover a device due to one of the following reasons:

- It is a known device that is connected over unreliable high-latency link
- You know that a device exists and is connected but you do not manage it
 - For example, you might want to visualize the link to a device in customer premise but do not manage that device
- The device can be legacy or prototype equipment with limited or faulty SNMP MIB support

Figure 6-14. Use cases for the Topology Editor

Add a device with the Topology Editor

- With this utility, you can edit a topology by manually adding and removing devices and connections
- You access this utility by right-clicking inside the Hop View
- An icon indicates a manually added entity
- The **Manually Added Devices** network view shows all manually added devices

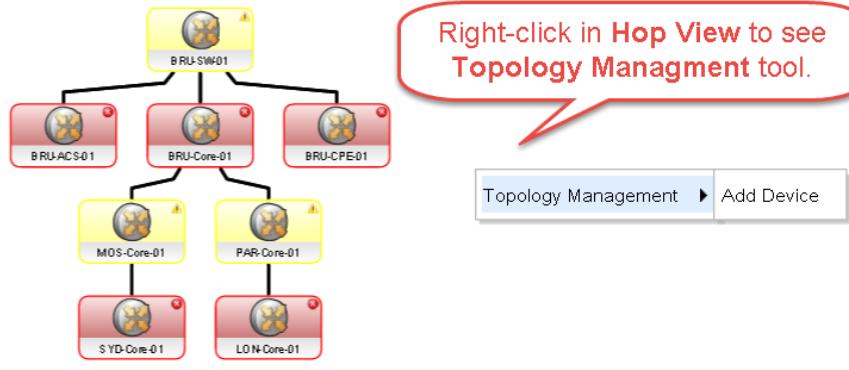


Figure 6-15. Add a device with the Topology Editor

You might need to use the Topology Editor to create representations of devices or links that the discovery process did not find. Several factors can prevent discovery of devices or links.

- A device was outside the scope.
- A device denied SNMP access by the Tivoli Network Manager server.
- A device belonged to a third-party network to which you had no access.

In large geographically diverse networks, the network map often contains several islands of connectivity. Connections between these clusters of devices often go through third-party networks to which Tivoli Network Manager has no access. When you create these devices and links, they can be included in root cause analysis calculations if you designate the device as capable of IP forwarding. You can also add connections between discovered devices. If two sites have a VPN that connects them, then Tivoli Network Manager shows the connectivity properly. If a third-party network intervenes between sites, you can use the Topology Editor to create links between interfaces on the discovered devices on each end of the connection.

Use the Add Device Wizard

- Enter **entityName** and **displayLabel**
- Only fields with asterisk (*) are essential
- To include a manually created device in root cause analysis, set the **ipForwarding** field to **forwarding**

Add Device Wizard

Entity Details

Please enter the relevant entity details for this device.

domain:	NOI_AGG_P
entityName:	MyManuallyAddedDevice1.example.class.com
entityType:	Chassis
displayLabel:	MyManuallyAddedDevice1
description:	
reason:	

Buttons: ? Cancel << Back Next >> Finish

Add Device Wizard

Chassis Details

Please enter the relevant chassis details for this device.

className:	Cisco74x
sysObjectid:	
sysName:	MyManuallyAddedDevice
sysDescr:	My fake device 1
sysLocation:	LoneStartState
sysContact:	Ima Geek 555-1212
ipForwarding:	forwarding
serialNumber:	123456789
modelName:	Cisco 7416
accessIPAddress:	10.10.255.47
accessProtocol:	IPv4
DNSName:	

Annotations: Set IpForwarding to include the device in root

Buttons: ? Cancel << Back Next >> Finish

Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-16. Use the Add Device Wizard

Confirm device details

Add Device Wizard

Confirm Details

Please confirm that the following details are correct for this device.

domain:	NOI_AGG_P
entityName:	MyManuallyAddedDevice1.example.class.com
entityType:	Chassis
displayLabel:	MyManuallyAddedDevice1
description:	
reason:	
className:	Cisco74x
sysObjectid:	
sysName:	MyManuallyAddedDevice
sysDescr:	My fake device 1
sysLocation:	LoneStartState
sysContact:	Ima Geek 555-1212
ipForwarding:	forwarding

Cancel << Back Next >> **Finish**

Advanced Visualization

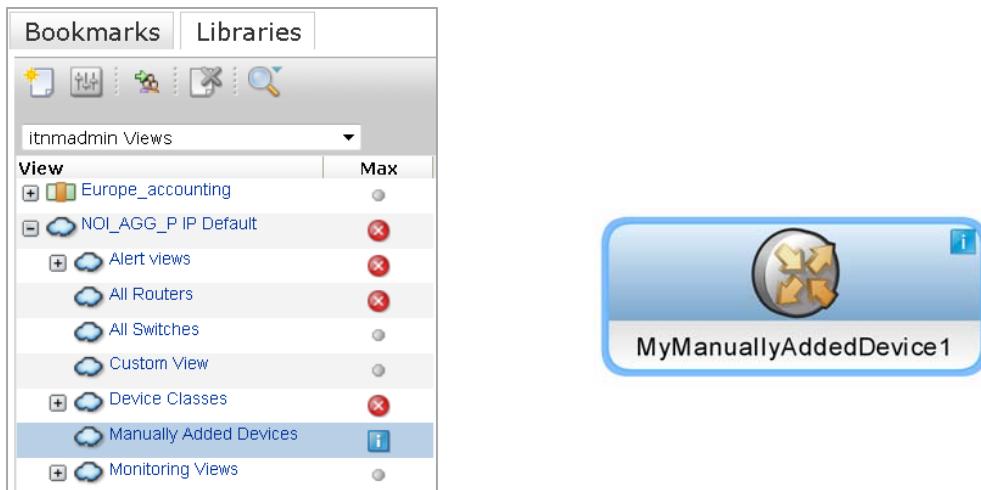
Add Device Wizard

Success

Device MyManuallyAddedDevice1.example.class.com has been successfully added.
Device has been added to the NOI_AGG_P domain.

Figure 6-17. Confirm device details

Manually Added Devices network view



Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-18. Manually Added Devices network view

All manually added devices appear in the **Manually Added Devices** network view. You can quickly use this view to see all manually created devices.

Delete Device

- The **Delete Device** tool deletes a manually added device from the topology database
 - Device data is removed from the topology database
 - Started from the right-click menu in the Hop View
- A device selection screen shows manually created devices
 - User checks boxes for one or more manually created devices to select devices for deletion
 - User confirms the list that shows all devices that are selected for deletion
- After you click **Finish**, you are shown a screen that confirms that Tivoli Network Manager successfully deleted the devices

Figure 6-19. Delete Device

Manually added devices must be manually deleted. They are not automatically deleted out of the database after a preset number of discoveries fail to find them.

Add connection

The add connection wizard can be started from the right-click menu

- Scenario 1: When two devices are selected
 - The first screen lists the two devices to connect
 - A check box can swap the devices between **From Device** and **To Device**
- Scenario 2: When one device is selected
 - The first screen lists one device to be connected and tells the user to click **Next** to choose the second device
 - When you click **Next**, the entity search window starts
 - Search for the device to connect and select it
- On the **Connection Details** screen, the user selects whether to connect the devices at the interface level or the device level

User confirms the configuration, and the map shows the connection

Figure 6-20. Add connection

Add a network connection

- In the **Hop View**, right-click one or both of the devices to which you want to add a connection
 - If you select two devices, both devices are in the **Device Selection**
 - If you select one device, Tivoli Network Manager prompts you for the second device
 - You can use the entity search to find and select the device name
- In the **Device Selection** window, the device you selected is shown
- Click **Next** to select the other connection endpoint

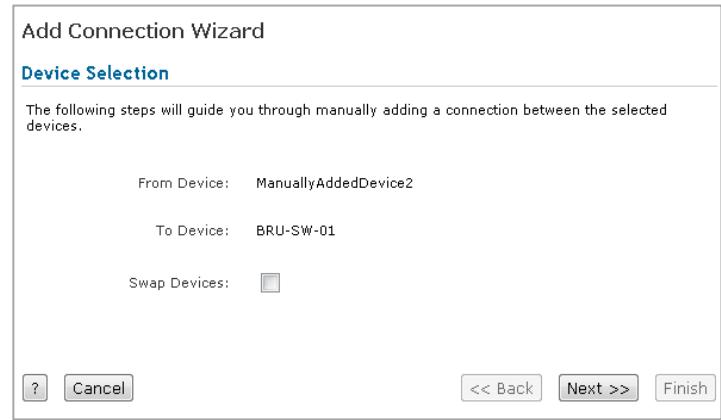
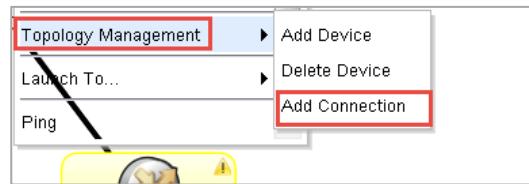


Figure 6-21. Add a network connection

You can create a connection from within the Hop View in one of three ways:

- Right-click an empty area of the Hop View and specify each end of the connection.
- Right-click one device in the Hop View and specify the second end of the connection.
- Select two devices in the Hop View. Right-click to build a connection between the selected devices.

If Tivoli Network Manager discovered the device on each end of the connection, you can select the interface on each end. However, if you manually created one of the devices, that device does not have any interfaces to select. Build the connection to the device itself.



Note

If you want to create devices with interfaces, use the **PresetLayer.stch** stitcher instead of the Topology Editor.

Specify connection details

- You can use the Topology Editor to build a connection between two discovered devices when connectivity is missing
- Select the correct interface on each device
- Specify connection speed so Tivoli Network Manager can properly represent link bandwidth

The screenshot shows the 'Add Connection Wizard' interface. On the left, a 'Success' step indicates a connection was added successfully. On the right, the 'Connection Details' step is displayed, showing fields for From Device (ManuallyAddedDevice2), From Interface (None), To Device (BRU-SW-01), To Interface (Fa0/1), Connectivity (Layer 1), Speed (1000000 bits/sec), and Reason (Lab exercise). Navigation buttons like << Back, Next >>, and Finish are visible.

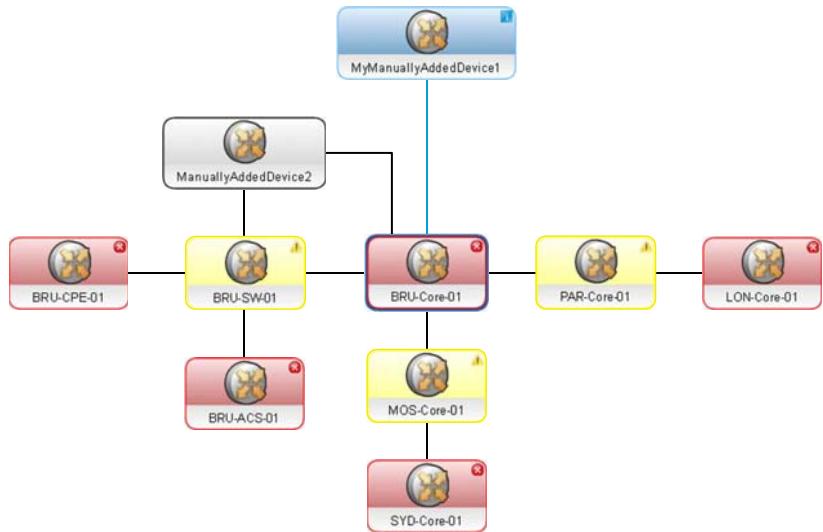
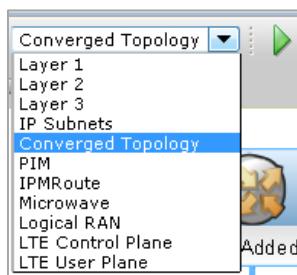
Add Connection Wizard	
Success	Layer 1 connection ManuallyAddedDevice2 -> BRU-SW-01 [Fa0/1] was successfully added. To view the connection the topology will need to be recentered with the following settings: Seed: ManuallyAddedDevice2 Connectivity: Layer 1
<input type="button" value="Recenter & Close"/> <input type="button" value="Close"/>	
© Copyright IBM Corporation 2017	

Figure 6-22. Specify connection details

The most common use of the Topology Editor is to build connections between discovered devices when Tivoli Network Manager did not discover all connections because of some access restriction. When you specify connection details, you can also specify the layer in which to add the connection and can add a speed in bits per second.

View added devices and connections

- Added devices and connections appear in the network topology
- Make sure to select the correct topology layer to see added connections



© Copyright IBM Corporation 2017

Figure 6-23. View added devices and connections

Troubleshooting the Topology Editor

Diagnostic logs for GUI are on the Jazz / DASH server

- `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm/ncp_topoviz.0.log`
- `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm/ncp_topoviz.0.trace`

Manually added device information storage

- Manually added devices are part of a manual topology domain
 - This classification prevents normal domain operations (such as network discovery) from deleting manually added entities
- Currently, if you restore your network topology cache to an empty database, the target domain owns the manually added entities
 - In this case, manually added entities do not re-create properly
 - They do not appear in the **Manually Added Devices** network view
 - Customers are unlikely to face this issue
 - If you manually added devices and connections to your topology, tell IBM Tivoli support personnel

Figure 6-24. Troubleshooting the Topology Editor

6.3. Device Structure browser

You can use the Device Structure browser to see the details of what is inside a particular device chassis. In this view, you can see the interfaces and protocol endpoints as they run on a device. You can also manage or unmanage specific interfaces on the device in this tool.

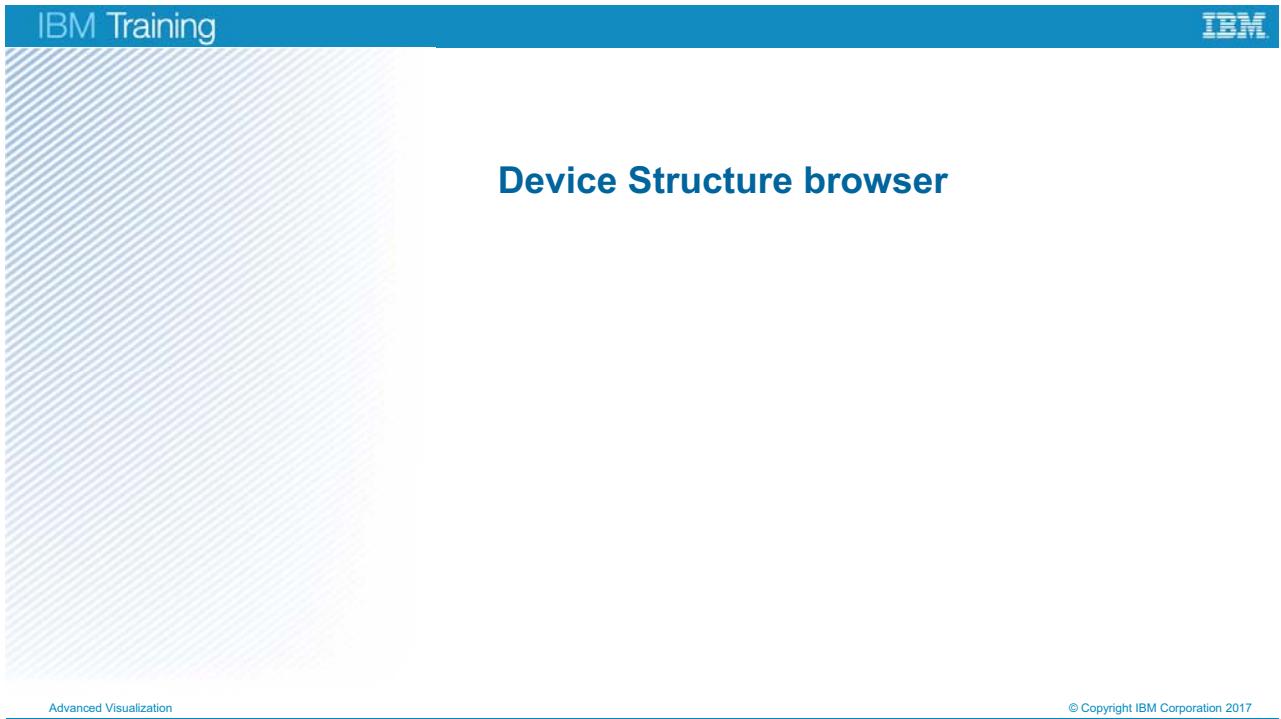


Figure 6-25. Device Structure browser

Device Structure browser description

The Device Structure browser is an information display and diagnostic tool for network devices

- It helps you pinpoint the cause of a network issue down to the component level

Using the Device Structure browser, you can do the following tasks:

- Examine the structure of a selected device and the data that is associated with its internal components
 - This data can be used to support the resolution of network problems
- Track the alert status for each device
- Use the diagnostic and information retrieval tools available in the Device Structure browser
 - With these tools, you can test the selected device or component
- Manage or unmanage a chassis or interface

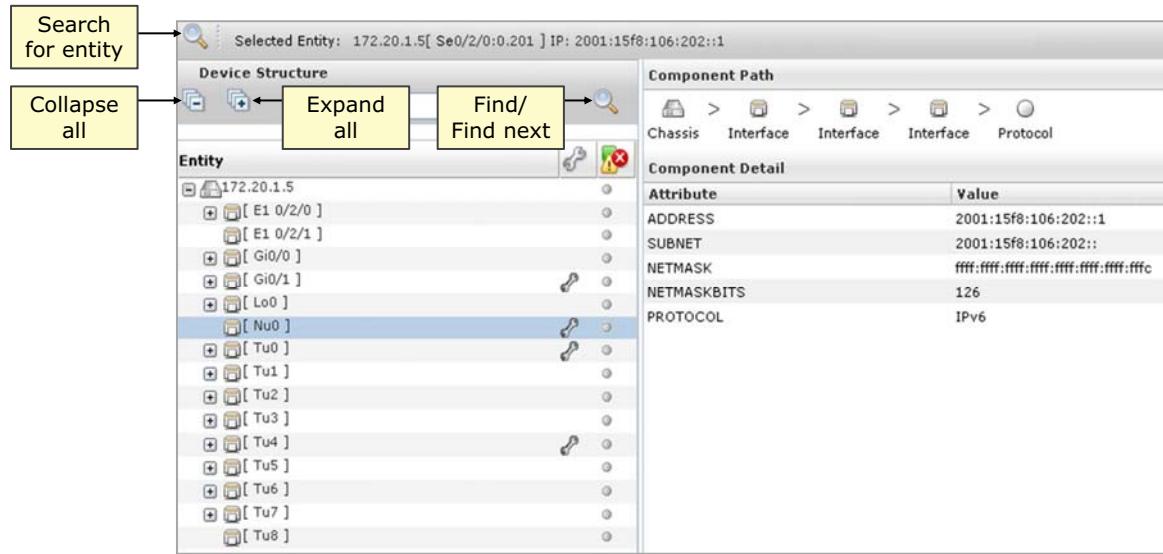
Figure 6-26. Device Structure browser description

Device Structure browser

- See which ports or interfaces are running a particular technology
- Understand the dependencies that a particular technology has on physical components within a device
- View configuration and state data that relates to a particular technology
- Device Structure browser uses NCIM device containment model and technology data
 - NCIM **containment**, **protocolEndPoint**, and **dependency** core model relationship tables are used
 - **\$NCHOME/etc/precision/ModelNcimDb.cfg** maps the Object Query Language (OQL) database to NCIM database and determines NCIM table content and relationships between entities
 - All data for a device that is found in the **entityDetails** table of the NCIM database is shown in the Device Structure browser

Figure 6-27. Device Structure browser

Structure browser tree



Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-28. Structure browser tree



Information

You can disable the Structure browser from displaying the status icon that indicates whether an interface is managed or not. To make this configuration change, edit the `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/structurebrowser.properties` file on the Jazz / DASH GUI server.

Change the **structurebrowser.showManagedStatus** to **false**.

```
# Display/Hide managed status information from the tree.
structurebrowser.showManagedStatus=false
```

Collapsing and expanding structure browser tree

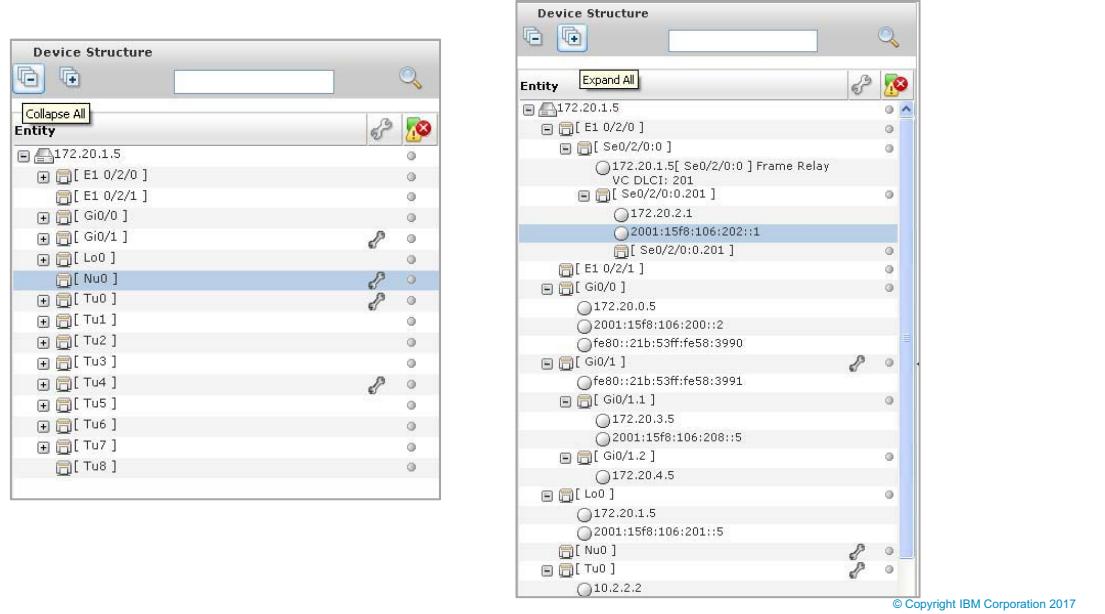


Figure 6-29. Collapsing and expanding structure browser tree

The wrench icons on an interface indicate that the interface is unmanaged.

Structure browser basic search

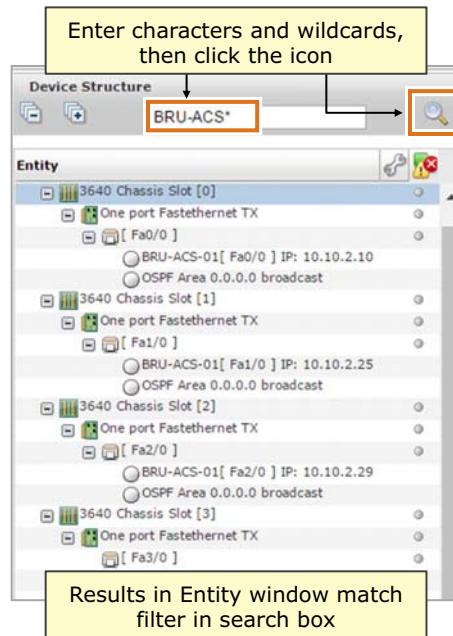
- Users enter the search criteria in the input box and click the search icon
 - Branches are expanded as necessary to show matches
 - Searches are conducted in the forward direction only
 - To reset the search, select the root node and click **Find / Find Next** icon
- The search is conducted by using a predefined fixed set of database columns, looking for equality or likeness with the user-entered string
 - The query searches the following fields:

ifName	accessProtocol	entPhysicalVendorType
ifDescr	accessIPAddress	from table interface
ifAlias	duplex	address
ifTypeString	DNSName and	protocol
ifPhysAddress	subnet from	subnet
	ipEndPointTable	

Figure 6-30. Structure browser basic search

Device Structure browser search function

- Wildcard characters that are supported are the percent sign (%) and the asterisk (*)
 - Wildcard characters are supported at both ends of the search string
- For example, ***ipv*** or **ipv*** or %**ipv%** or **ipv%** as the search string searches for both IPv4 and IPv6 entities
- The searching is not case-sensitive
 - Internally, the entered value and columns in the query are converted to lowercase characters and then compared
 - For example, **ipv4**, **IPV4**, and **iPV4** are all treated identically



© Copyright IBM Corporation 2017

Figure 6-31. Device Structure browser search function

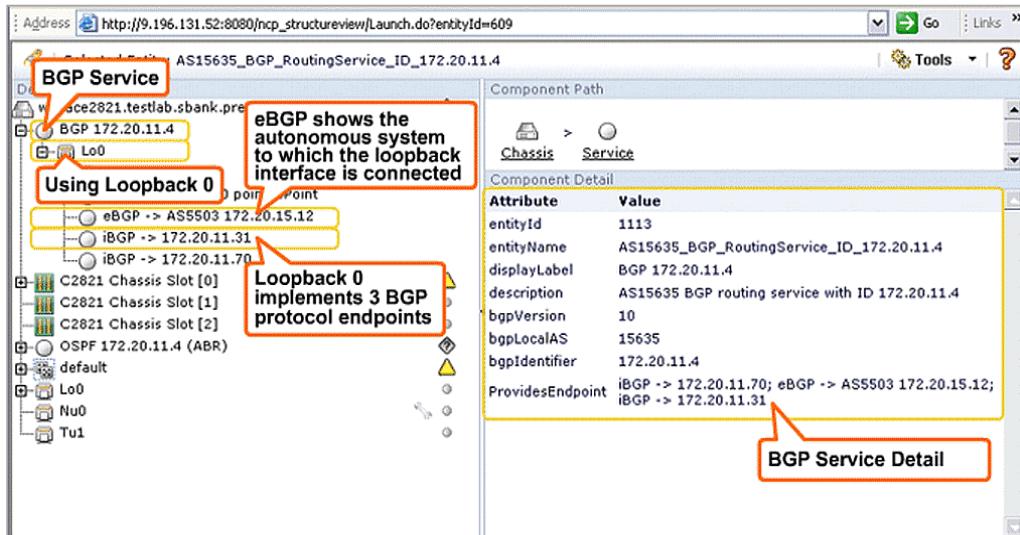
Structure browser troubleshooting

- Log files
 - Found in `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm`
 - In previous versions, the log files for the structure browser were under the `$TIPHOME` directory
 - Default files
 - `ncp_structurereview.0.log`
 - `ncp_structurereview.0.trace`
 - Log settings
- `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/structurebrowser.properties`

Figure 6-32. Structure browser troubleshooting

Tivoli Network Manager processes write log and trace files to the `$NCHOME/log/precision` directory. However, the Device Structure browser is part of the GUI, so the directory for log and trace files is the `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm` directory.

Device Structure browser BGP support

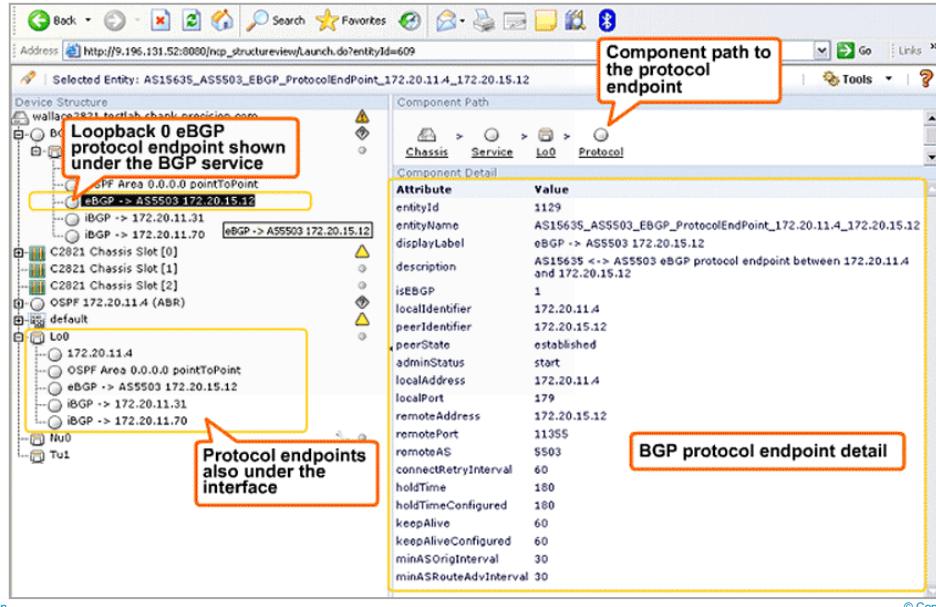


Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-33. Device Structure browser BGP support

Device Structure browser BGP protocol endpoints



© Copyright IBM Corporation 2017

Figure 6-34. Device Structure browser BGP protocol endpoints

Device Structure browser OSPF support

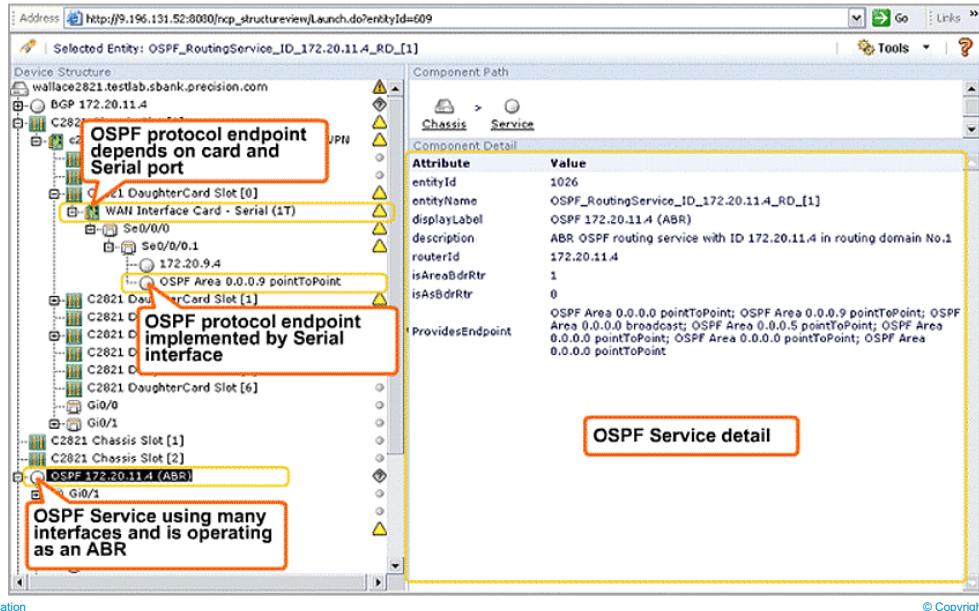
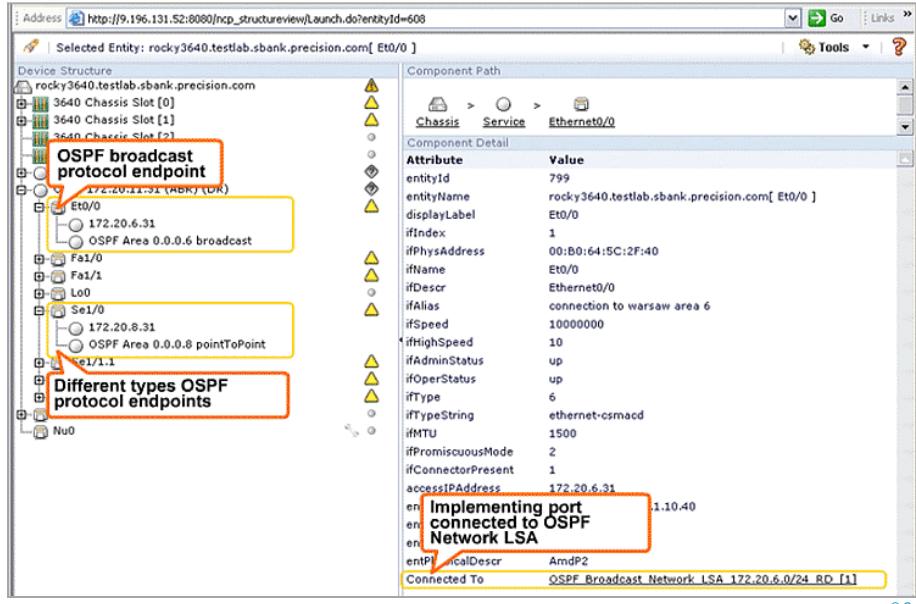


Figure 6-35. Device Structure browser OSPF support

Structure browser OSPF protocol endpoints

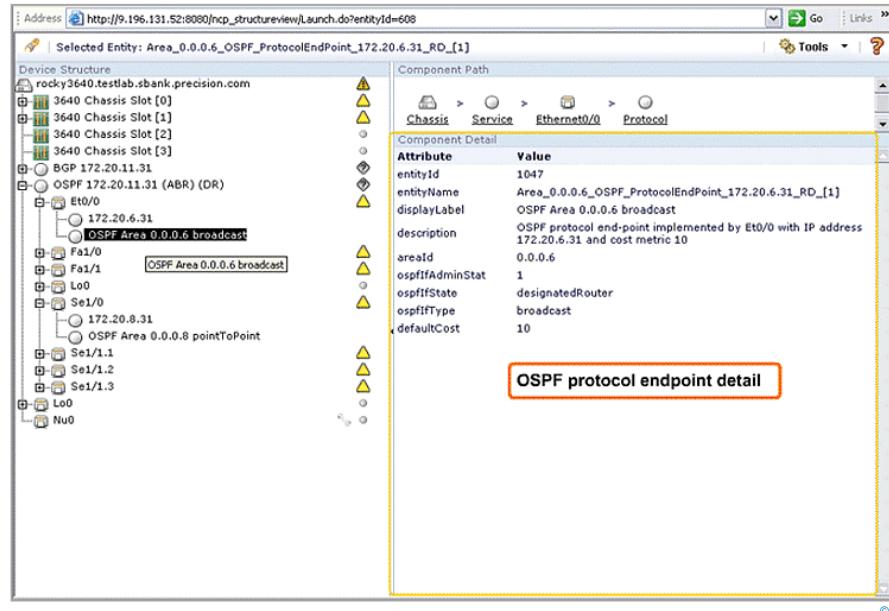


© Copyright IBM Corporation 2017

Figure 6-36. Structure browser OSPF protocol endpoints

An LSA is a link state advertisement. When two OSPF devices are connected by an Ethernet cable, the connection is called a link state advertisement.

Device Structure browser OSPF



© Copyright IBM Corporation 2017

Figure 6-37. Device Structure browser OSPF

Device Structure browser VLAN support

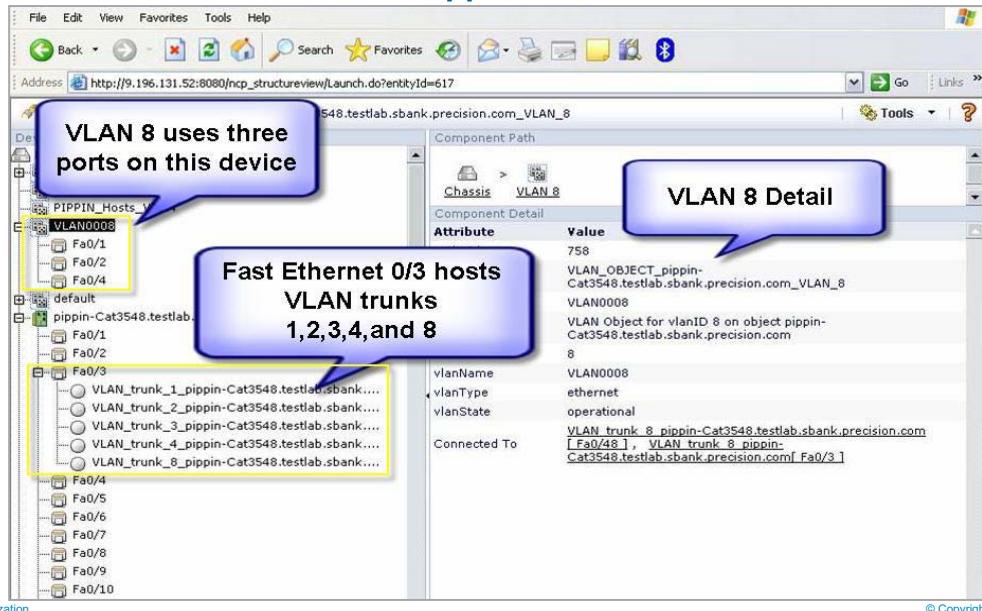


Figure 6-38. Device Structure browser VLAN support

Device Structure browser VRF support

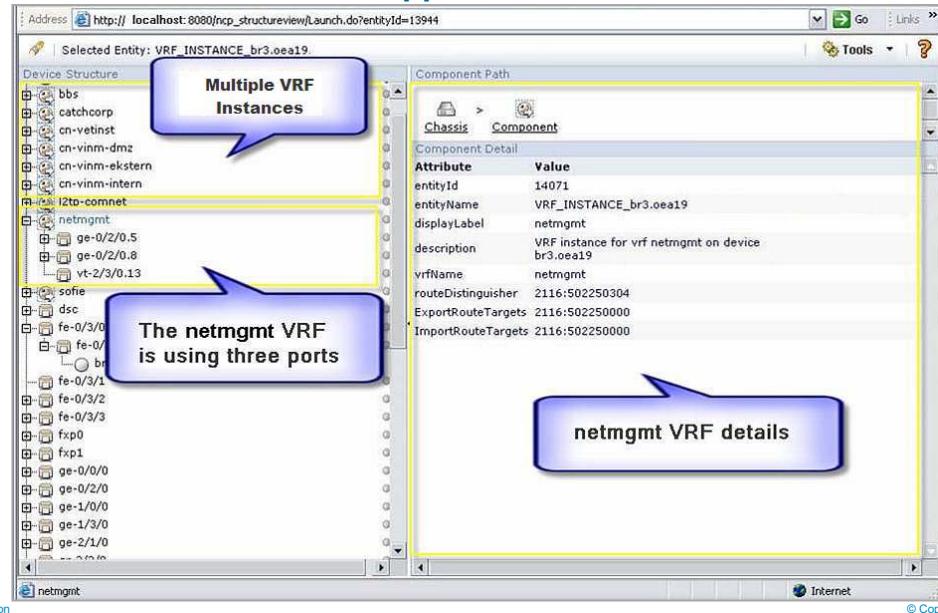
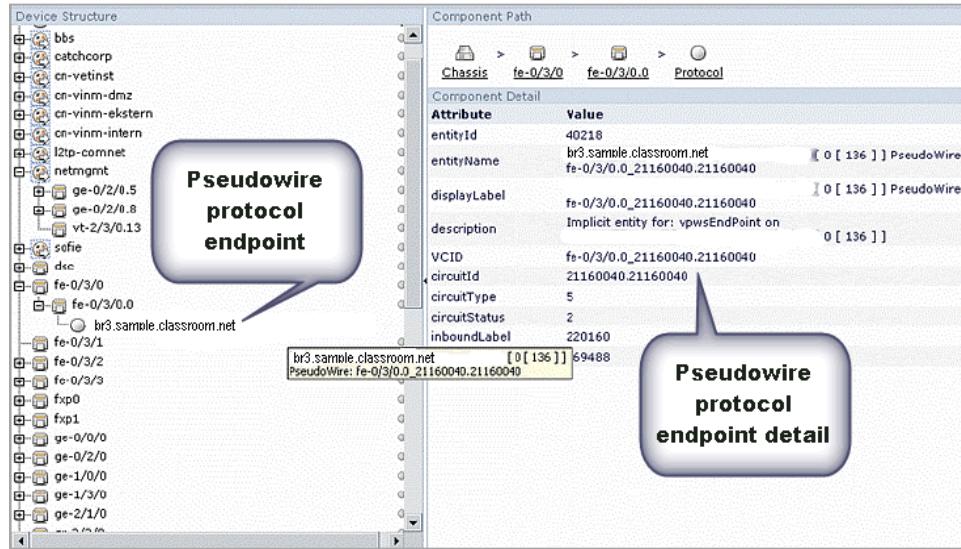


Figure 6-39. Device Structure browser VRF support

Device Structure browser support for pseudowire



Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-40. Device Structure browser support for pseudowire

Device Structure browser data flow

1. The discovery process writes to the DNCIM database
2. The DNCIM database is an embedded **sqlite** database that has the same table structure as the NCIM database
3. Discovery information is sent to the MODEL service
4. The MODEL service writes to the NCIM database and the NCIMCache database
 - **NCIM** is the primary repository for the topology GUI data
 - **NCIMCache** is used by the gateway to enrich events and do root cause analysis
 - Inside the NCIM database, the **entityDetails** table contains information for the Device Structure browser
 - Extra information can be added to the **entityDetails** table by modifying the `$NCHOME/etc/precision/DbEntityDetails.cfg` file

Figure 6-41. Device Structure browser data flow

Device Structure browser tables (1 of 2)

- **Entity:** This main table stores general information about the entity, such as **mainNodeEntityId**, **entityType**
 - The population of this table requires the use of the Entity agent
- **EntityData:** This table stores basic information about every discovered entity in the network
- **Contains:** This table stores the containment information of the entities
- **EntityType:** This table stores the type of the entity
- **mainNodeDetails:** This table stores the details about the main nodes only

Figure 6-42. Device Structure browser tables (1 of 2)

Device Structure browser tables (2 of 2)

- **Chassis, interface, ipEndpoint:** Stores specific information for each entity type
- **\$NCHOME/etc/precision/ModelNcimDb.cfg:** This configuration file controls the population of fields in the NCIM database
- **\$NCHOME/etc/precision/DbEntityDetails.cfg:** Customize this file to add fields to the Device Structure browser or add custom fields to the topology GUI
- **/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/ncimMetaData.xml:** This file determines what tables and fields are available choices for creating filtered network partition views

Advanced Visualization© Copyright IBM Corporation 2017

Figure 6-43. Device Structure browser tables (2 of 2)



Note

Use the `DbEntityDetails.cfg` file for adding custom fields to network maps or the Device Structure browser. The `ModelNcimDb.cfg` file is complex and crucial to the functioning of the product. Changing `DbEntityDetails.cfg` is less complex and has less potential for affecting other key processes. You must still edit the `ncimMetaData.xml` file for new custom fields to be available in pull-down menus.

Network view bookmarks

- With bookmarks, users can select the network views that they use the most and save those views in the **Bookmarks** tab
- When you view the Bookmarks tab, an event status refresh updates only the bookmarked views
 - Updating a smaller number of devices results in enhanced performance of the GUI

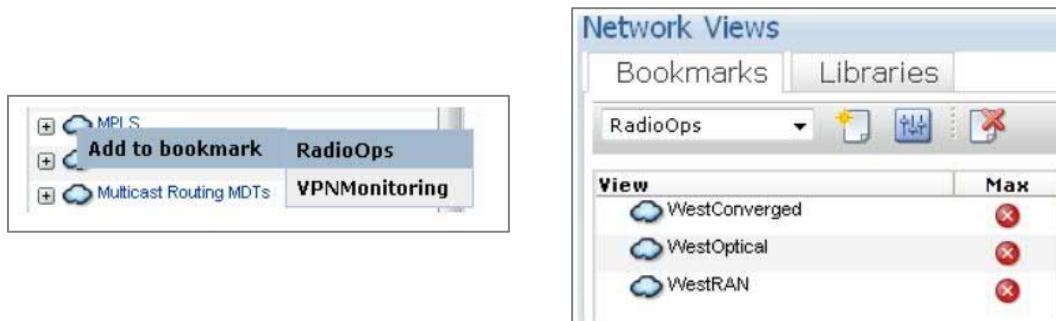


Figure 6-44. Network view bookmarks

Users can add specific network views to a bookmark.

- When IBM Tivoli Network Manager updates the event status on a bookmark view, the update is affecting fewer devices than when someone is trying to view the entire network view tree.
- The use of bookmarks results in enhanced performance of the GUI.
- In the preceding screen graphic, a bookmark has three network views. Event status changes update only these three network views as the operator is looking at this bookmark view.

When the user returns to the **Libraries** tab, event status updates must update all views. The use of the Libraries tab results in slower performance of the GUI.

The **Bookmarks** tab occurs first and is the default selection when a user opens network views.

6.4. Adding classes and icons

Sometimes manufacturers use new sysObjectID numbers (also called OID numbers) when they release a new version of operating software for existing device models. Tivoli Network Manager might show an “unknown” icon for a device that was previously represented with a router or switch icon due to the change in the OID number of the device. It is easy to update the Active Object Class (AOC) file for the device so that the GUI again uses the correct icon for the device. Some customers also want to customize the topology GUI to add their own icons for particular devices. This lesson shows how to update class and icon information in Tivoli Network Manager.

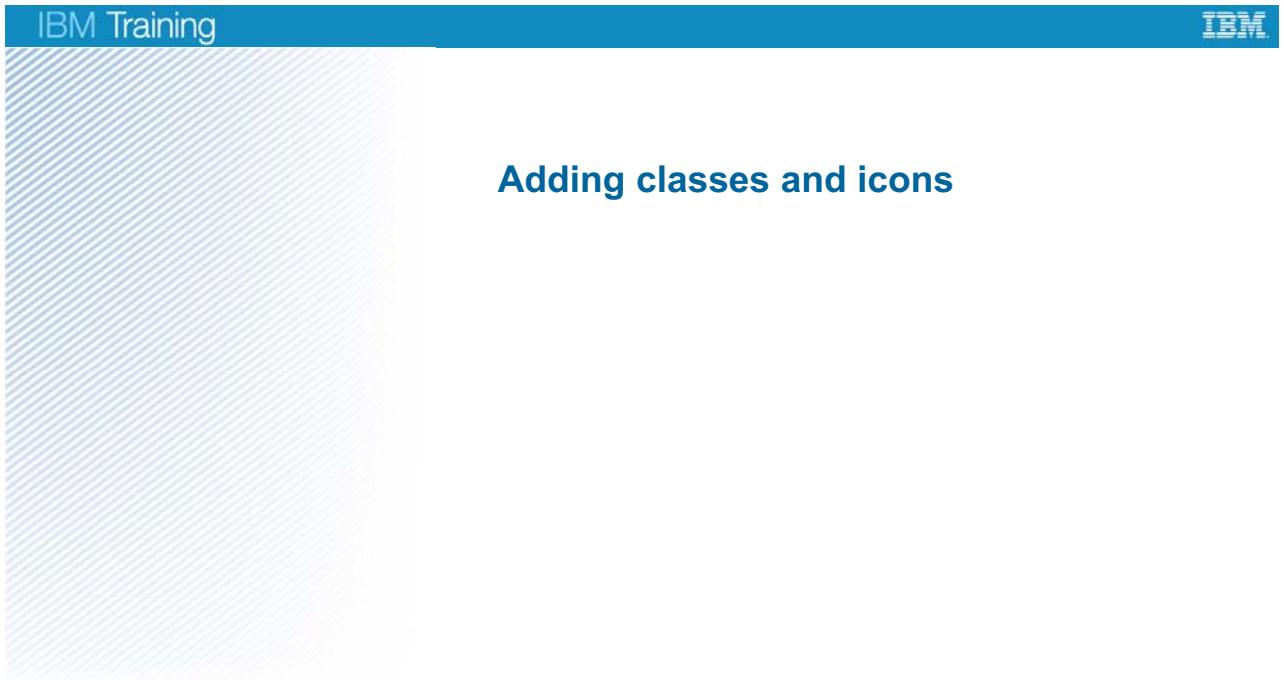


Figure 6-45. Adding classes and icons

Adding a device class (1 of 2)

- If your network contains a device for which no device class exists, Tivoli Network Manager represents this device with an icon that indicates the device is an unknown type



- You can add a class file that corresponds to the OID of that device

The class files are located in the `$ITNMHOME/aoc` directory

- When discovery runs again, devices that match the new class you created are now assigned to that class

Figure 6-46. Adding a device class (1 of 2)

Tivoli Network Manager uses a device icon with a question mark when it cannot determine what type of device it is. This inability to classify a device is because of one of the following conditions:

- A discovery finder found the device, but the discovery agent did not have access to the device.
- The OID number of the device did not match any OID number in one of the Active Object Class (AOC) files in the `$ITNMHOME/aoc` directory.

The AOC files determine the **Class** to which a device is assigned and what icon is used to represent that device. Sometimes an operating system upgrade to a router or switch can change the OID number of the device. Tivoli Network Manager might not recognize the new OID number. Update the AOC file to add the new OID number. When you restart the MODEL process, Tivoli Network Manager assigns the correct icon.

Adding a device class (2 of 2)

- Copy the Active Object Class (AOC) file from a similar device and rename the copied file to match your new device


```
cd $ITNMHOME/aoc
cp CiscoCat19xx.aoc NewYumDumSwitch.aoc
```
- Edit the new file to change the OID number in the filter and make sure correct **visual_icon** value is specified
- Save the file and run a new discovery

Change names and OID

```
/*
// File : NewYumDumSwitch.aoc
// Created for lab exercise ***
active object 'NewYumDumSwitch'
{
    super_class = 'CiscoNonRoutingSwitch';
    instantiate_rule = "EntityOID = '1.3.6.1.4.1.9999.888.77.6'";
    visual_icon = 'switch';
};
```

Device has correct icon after the new AOC file is created

Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-47. Adding a device class (2 of 2)

```
/*
// File: NewYumDumSwitch.aoc
//
// ***Created for lab exercise ***
active object 'NewYumDumSwitch'
{
    super_class = 'CiscoNonRoutingSwitch';
    instantiate_rule = "EntityOID = '1.3.6.1.4.2.9999.888.77.6'";
    visual_icon = 'Switch';
};
```

Adding icons

- Create 48 x 48 Scalable Vector Graphics (SVG) file
- On the Jazz / DASH server, do these steps:
 1. Copy icon to the `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/resource` directory
 2. Edit and save the `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/topoviz.properties` file
 - Add a line that describes the icon:
`topoviz.image.classname=iconname.extension`
 - Example:
`topoviz.deviceicon.InferredNetwork=light_cloud.svg`
`topoviz.deviceicon.HPPrinter=printer.svg`
`topoviz.deviceicon.PseudoNode=net1.svg`
`topoviz.image.NewYumDumSwitch=NewYum.svg` ← Must be the same name as the file in the **resource** directory
 3. Restart the GUI server (Jazz / DASH) by running these commands on that server:
 - `/opt/IBM/JazzSM/profile/bin/stopServer.sh server1 -username smadmin -password password`
 - `/opt/IBM/JazzSM/profile/bin/startServer.sh server1`
- Starting the GUI server does not require user name or password options

Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-48. Adding icons

Versions of Tivoli Network Manager before version 4.2 used Tivoli Integrated Portal to show topology information, do administrative tasks, and show event information to the users and administrators. Tivoli Integrated Portal required Java.

Tivoli Network Manager 4.2 and later versions abandon the use of Tivoli Integrated Portal and Java. Instead, the GUI is supplied by the Jazz server with the Dashboard Application Services Hub (DASH).

In small network environments, it is possible to run the Netcool Operations Insight base package and Tivoli Network Manager on the same physical computer. However, in most production deployments, you have at least three servers:

- A Netcool/OMNIbus ObjectServer, Jazz (with DASH)
- A DB2 or Oracle database server
- The Tivoli Network Manager server

6.5. Cross-domain network views

A domain is a collection of Linux processes in Tivoli Network Manager. Sometimes it is necessary to break up a discovery into smaller discoveries. This fragmenting of the network discovery can happen due to sizing, geographic, or political constraints within an organization. Even if discovery and monitoring activities are distributed across multiple servers, operations center personnel need to be able to view devices in one map. You can use Tivoli Network Manager 4.x to create a cross-domain network view so that a map can contain devices from more than one domain.

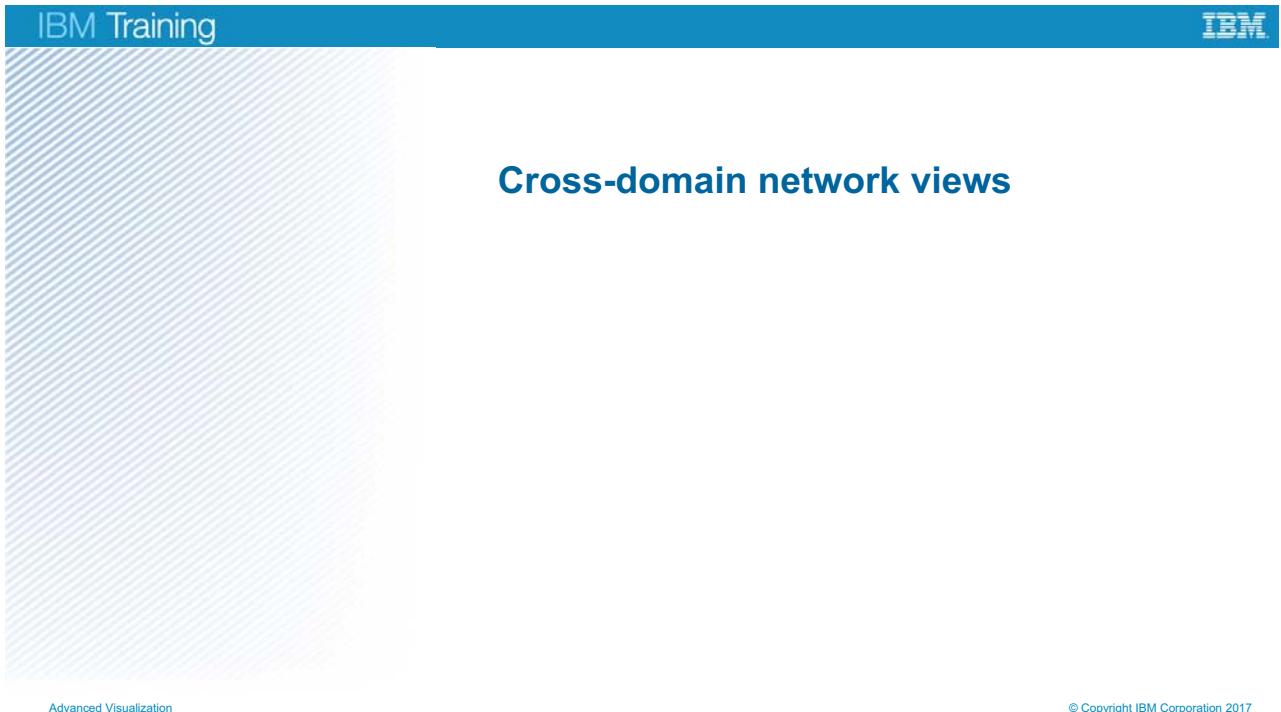


Figure 6-49. Cross-domain network views

Cross-domain processing and visualization

- Multiple domains are discovered
- Stitchers find links between domains and create cross-domain connections
- The topology GUI can render cross-domain topologies
- Root cause analysis remains limited to contiguous network maps within a domain

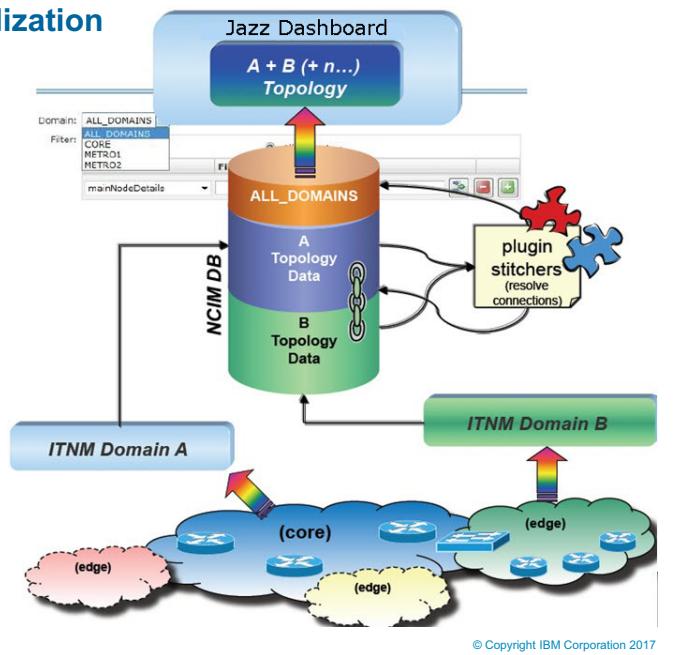


Figure 6-50. Cross-domain processing and visualization

The preceding slide illustrates that devices in different domains are discovered separately. The discovery data remains separate inside the NCIM database. Cross-domain stitchers determine the links between the domains. These stitchers use information these kinds of information:

- Remote neighbor information on interfaces
- Class /30 subnets (these subnets can have only two IP addresses)
- Information from the **PresetLayer** stitcher
- Manually instantiated links

An aggregated network view shows devices from all domains.

Prepare configuration files

- Go to the `$NCHOME/etc/precision` directory
- Make a domain-specific copy of the primary domain `ConfigItnm.domainName.cfg` and `DbLogins.domainName.cfg` files for the second domain:

```
cd $NCHOME/etc/precision
cp ConfigItnm.NOI_AGG_P.cfg ConfigItnm.NOI_AGG_P2.cfg
cp DbLogins.NOI_AGG_P.cfg DbLogins.NOI_AGG_P2.cfg
```

- Stop IBM Tivoli Network Manager:

```
itnm_stop
```

- Run the `domain_create.pl` script to clone the existing domain:

```
cd $ITNMHOME/scripts/perl/scripts
ncp_perl domain_create.pl -domain NOI_AGG_P2 -clone NOI_AGG_P
```

[Advanced Visualization](#)

© Copyright IBM Corporation 2017

Figure 6-51. Prepare configuration files

Each domain needs domain-specific configuration files. The `ConfigItnm.cfg` file contains information about how the IBM Tivoli Network Manager connects to the OMNIbus ObjectServer. The `DbLogins.cfg` file contains information about how IBM Tivoli Network Manager connects to databases. Without this information, the new domains cannot properly connect to the NCIM and other databases.

The default `DbLogins.cfg` file does not contain the proper information to authenticate to databases. When you create a new domain, make sure to copy the domain-specific file from the primary domain. For example, copy `DbLogins.NOI_AGG_P.cfg` (rather than `DbLogins.cfg`) to `DbLogins.NOI_AGG_P2.cfg`.

Enable cross-domain processing

- Edit the `DiscoConfig.domainName.cfg` file for each domain
- Change the value of `m_EnableCrossDomainProcessing` to 1

```

    1,      // Do we want to send stitcher events to omnibus.
    0,      // Should we assign private address spaces to VRF interfaces?
    0,      // The display label assignment mode for main nodes
    0,      // Do we want to run BuildInterfaceName to provide custom int naming
    1,      // Do we want to perform cross domain processing ?
    0,      // Do we wish to refresh the discovery process on each full cycle
);

```

- Alternatively, change the `DiscoConfig.domainName.cfg` file for the primary domain and then make domain-specific copies of this file for each other domain
- Start IBM Tivoli Network Manager for each domain:

`itnm_start`

- Run a full discovery in each domain so that cross-domain stitchers can process the links between the domains

Figure 6-52. Enable cross-domain processing

Cross-domain processing is not enabled by default.

To enable cross-domain processing, change each `DiscoConfig.domainName.cfg` file to set `m_EnableCrossDomainProcessing` to 1.

```

m_EnableCrossDomainProcessing,
...
1, // Do we want to perform cross-domain
// processing?

```

After you enable cross-domain processing, run a full discovery of each domain to create the cross-domain connections.

Cross-domain processing stitchers

Name	Description
AggregationDomain	Handles processing required to update aggregation domain after discovery
AggregationDomain**	Creates collection entities and manages the aggregation domain
LinkDomains	Master link domain stitcher responsible for starting plug-ins and creating cross-domain connections
LinkDomainsViaBGPSessions	Connects domains via BGP data
LinkDomainsViaCDP	Connects domains via CDP data
LinkDomainsViaOSPF	Connects domains via OSPF data
LinkDomainsViaPIM	Connects domains via PIM data
LinkDomainsViaPseudoWires	Connects domains via pseudowire data
LinkDomainsViaSlash30Subnet	Connects domains via /30 subnets
LinkDomainsViaAgentXRemoteField	Connects domains via a set of adjacency data from return tables of a set of discovery agents
LinkDomainsLoadInterfaceDescriptionMatches	Literal and regular expression search-based cross-domain connections
LinkDomainsLoadPresetConnections	User-defined preset connections
LinkDomains**	Various utility stitchers

[Advanced Visualization](#)

© Copyright IBM Corporation 2017

Figure 6-53. Cross-domain processing stitchers

These stitchers process various types of connectivity information to create cross-domain links.

Create default views for the new domain (1 of 2)

- In the Network Views, click the icon to create a new view
- Enter the new domain name and select **Dynamic Views - Template** as the type

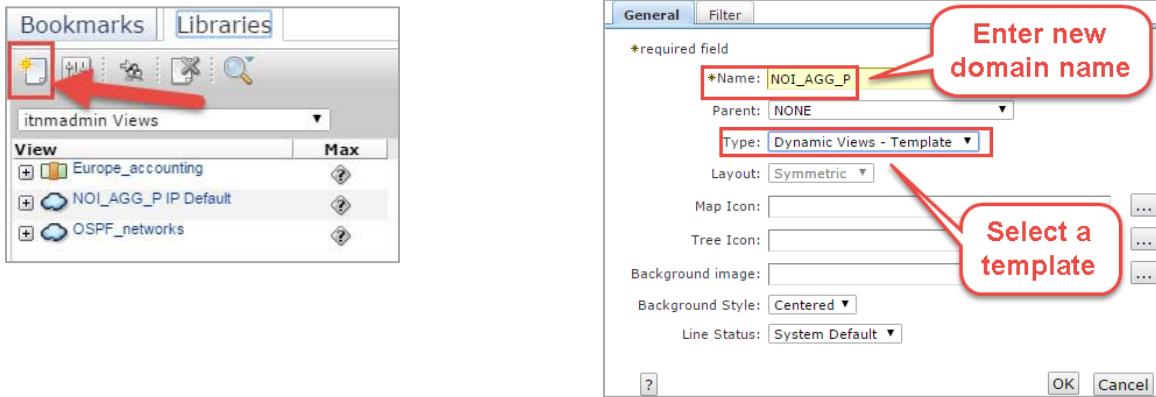


Figure 6-54. Create default views for the new domain (1 of 2)

For each domain you add, create a new set of default network views. Select **Dynamic Views – Template** as the type.

Create default views for the new domain (2 of 2)

- Click the **Filter** tab
- Select the new domain, the **IP Default** template, and click **OK**
 - This action creates default network views for the new domain

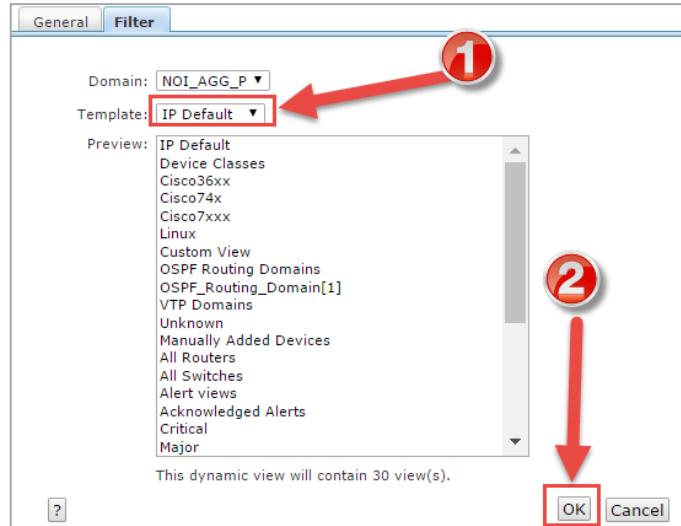
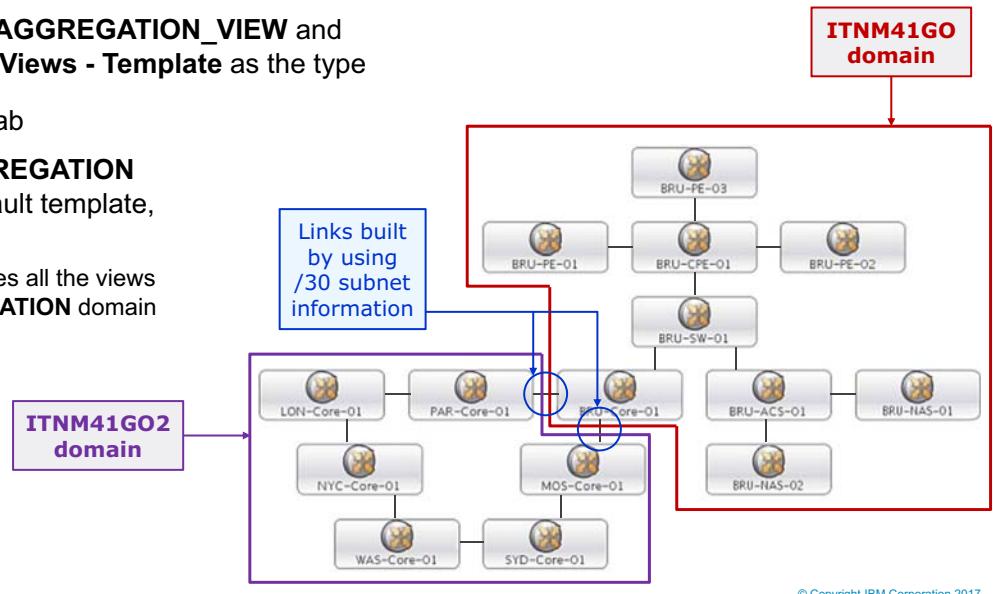


Figure 6-55. Create default views for the new domain (2 of 2)

Selecting the **IP Default** template on the filter tab causes all network views to be generated for the new domain.

Create the AGGREGATION_VIEW

- Enter the name **AGGREGATION_VIEW** and select **Dynamic Views - Template** as the type
- Click the **Filter** tab
- Select the **AGGREGATION** view, the IP Default template, and click **OK**
 - This action creates all the views for the **AGGREGATION** domain

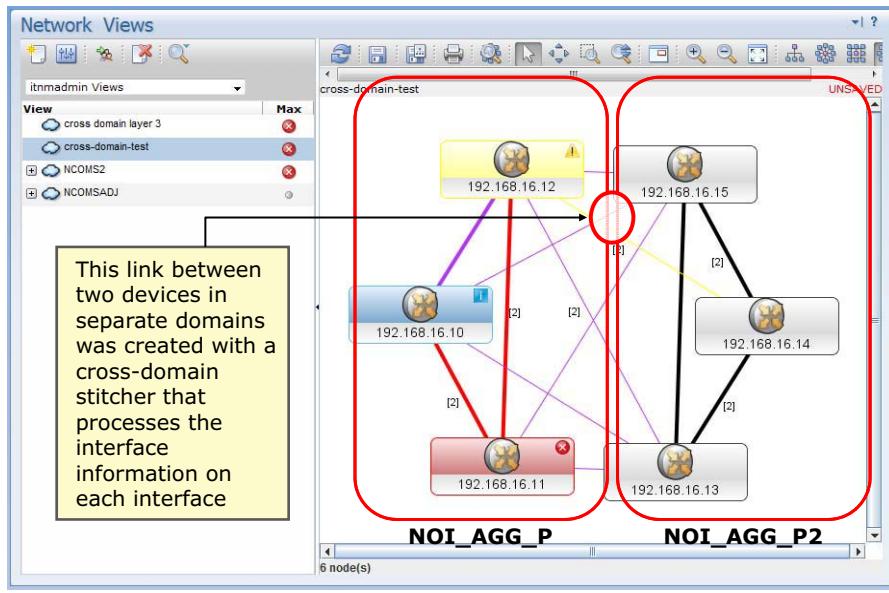


© Copyright IBM Corporation 2017

Figure 6-56. Create the AGGREGATION_VIEW

In the same way that you create views for each added domain, create a set of default views for the **AGGREGATION** view.

Cross-domain Layer 2 view



© Copyright IBM Corporation 2017

Figure 6-57. Cross-domain Layer 2 view

The preceding graphic shows devices from two separate discovery domains. Tivoli Network Manager used the remote neighbor information on interfaces to build the cross-domain link. The ovals that highlight the two domains are for illustrative purposes only.

Cross-domain Layer 3 view

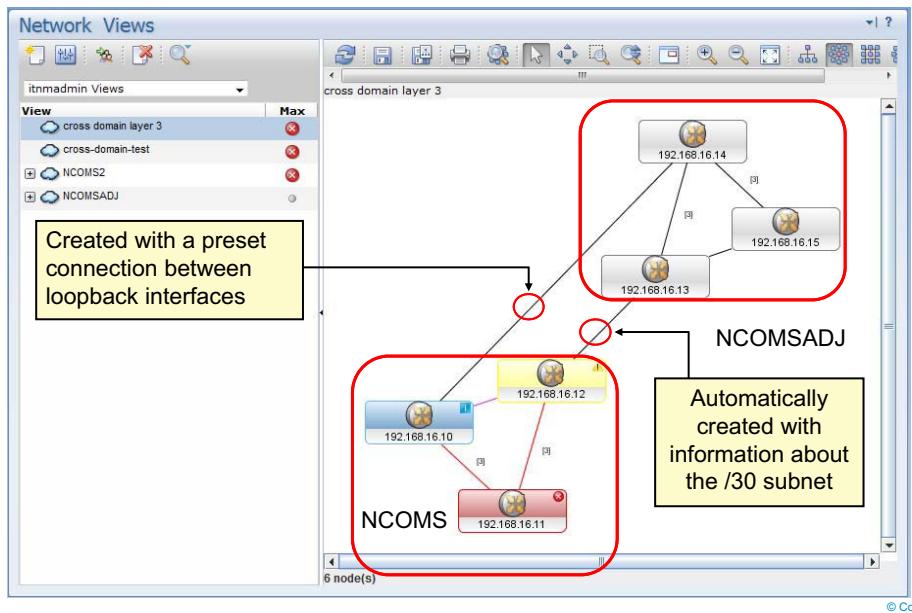


Figure 6-58. Cross-domain Layer 3 view

The preceding graphic shows two links between the separate domains. Tivoli Network Manager derived one of the links from information in the **PresetLayer** stitcher. It derived the second link by subnet addressing. Two interfaces each had an address on a /30 subnet. Since a /30 subnet can have only two IP addresses, Tivoli Network Manager built a link between the two interfaces that existed in separate domains.

Aggregation domain is for visualization only

The AGGREGATION domain is not available for use with the following tools:

- Network polling
- Management database access
- **Network Hop View** toolbar domain menu
- Entity search (**Structure Browser** and network views)
- Path views (the path tool cannot trace a path across domains)
- SNMP MIB browser, SNMP MIB graph
- Discovery status
- Discovery configuration
- Reports

Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-59. Aggregation domain is for visualization only

The **Aggregation** domain is only for visualization purposes. For that reason, it is not available to Tivoli Network Manager configuration tools, reports, or in path views.

Topic summary

After you complete this unit, you can do the following tasks:

- Clone a domain to create a second domain
- Create a cross-domain network view

Figure 6-60. Topic summary

Exercise: Advanced visualization

Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-61. Exercise: Advanced visualization

Complete the exercises for this unit.

Review questions

1. How are MPLS TE views created?
2. What is one reason that an error message can appear when you are creating a path view?
3. What file can you modify to add more information to the Device Structure browser?
4. How can you easily see all manually added devices?

Figure 6-62. Review questions

Write your answers here:

Unit summary

- Launch an IPv4 Path View
- Use the Topology Editor to manually add devices and connections
- Use the Device Structure browser
- Create a cross-domain network view

Advanced Visualization

© Copyright IBM Corporation 2017

Figure 6-63. Unit summary

Review answers

1. MPLS TE views are created during the discovery process. These views cannot be created manually.
2. An error message can appear when you create a path view if a device in the path is not included in the discovery.
3. To add more information to the Device Structure browser, modify the `$NCHOME/etc/precision/DbEntityDetails.cfg` file.
4. You can see all manually added devices in the Manually Added Devices network view.

Figure 6-64. Review answers

Unit 7. The Network Health Dashboard

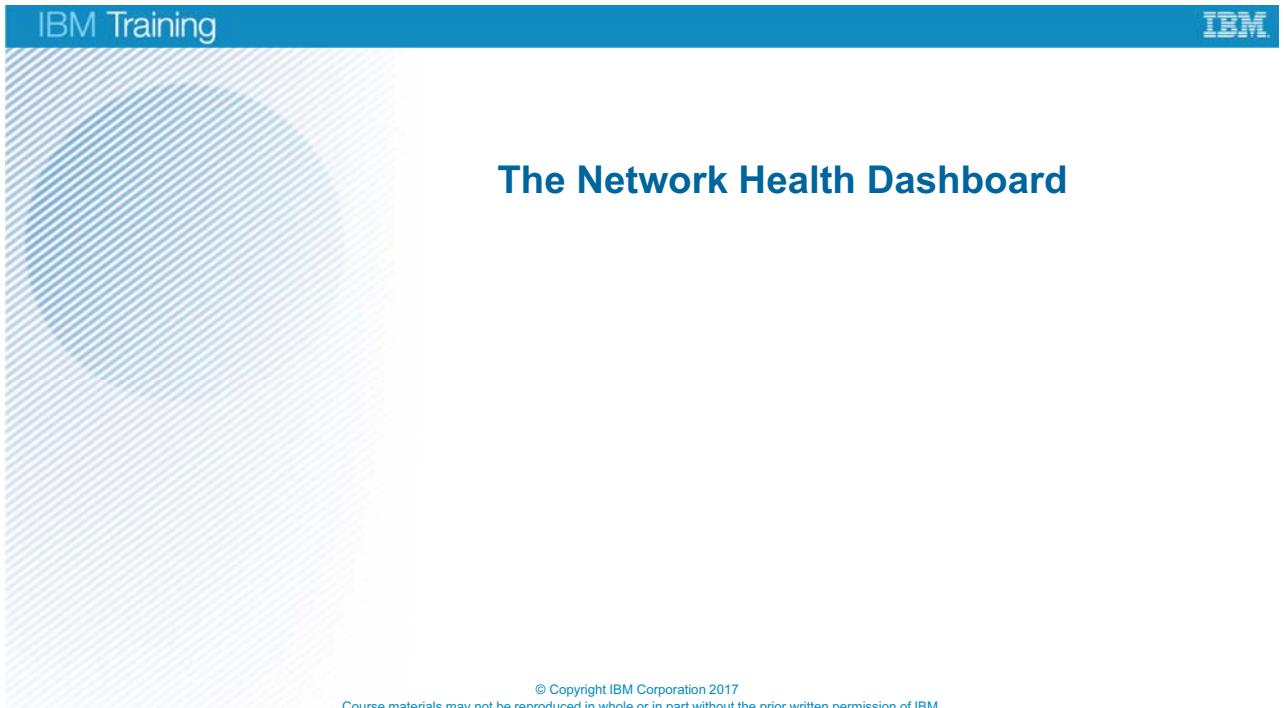


Figure 7-1. The Network Health Dashboard

Estimated time

00:15

Overview

This unit explains the Network Health Dashboard and then explains the basic steps to creating custom dashboards.

Unit objectives

- Use the Network Health Dashboard

The Network Health Dashboard

© Copyright IBM Corporation 2017

Figure 7-2. Unit objectives

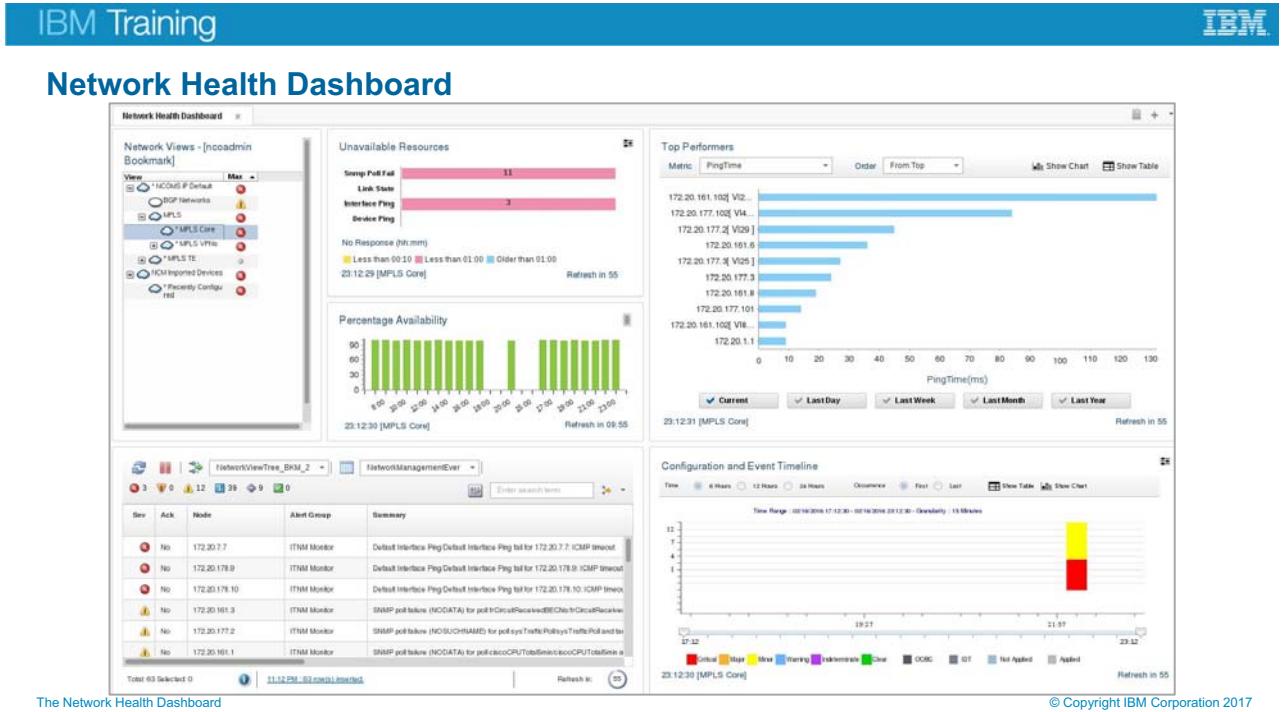


Figure 7-3. Network Health Dashboard

When you install Tivoli Network Manager with Netcool Operations Insight, you can add the Network Health Dashboard to existing dashboards.

Select a network view

- Select a bookmarked view from the upper-left corner of the Network Health Dashboard
- Data in the remainder of the Network Health Dashboard represents only the devices in the selected network view

Network Views - [ncoadmin Bookmark]

View	Max
NCM Imported Devices	X
* Recently Configured	X
NCOMS IP Default	X
* MPLS TE	○
MPLS	X
* MPLS VPNs	X
* VC_1999	X
* sirius	X
* shield	X
* OMNI_CP	X
* lunar	X
* Gekko&co	X
* benthic_spoke	X
* ACME	X
* MPLS Core	X
BGP Networks	⚠

Select a network view. All other data in the Network Health Dashboard applies to the selected network view.

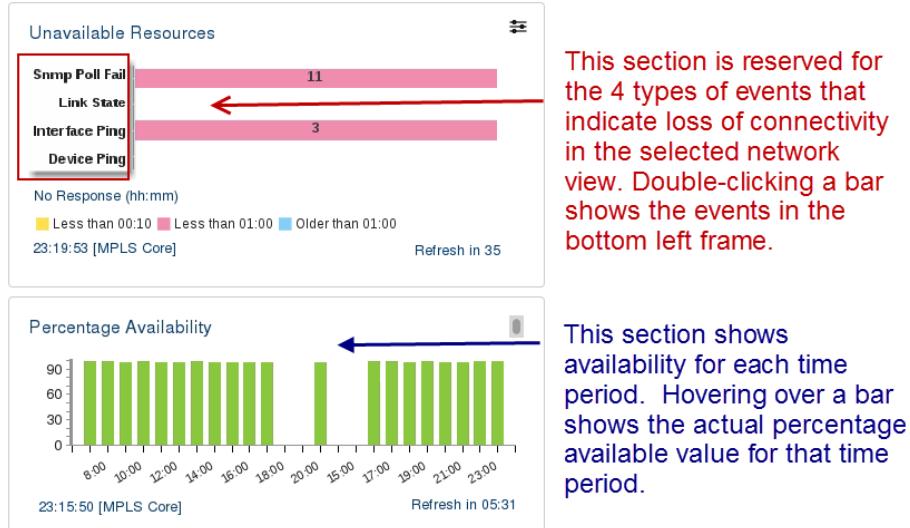
The Network Health Dashboard

© Copyright IBM Corporation 2017

Figure 7-4. Select a network view

Tivoli Network Manager comes with many default network views. Most users do not need all of those views. Users are encouraged to bookmark those views that they use most often. By looking at bookmarks instead of the network views, users can quickly locate the views that they need. Also, fewer database updates are needed to update the status icons because users are looking for a subset of the total number of network views. The Network Health Dashboard works only with user bookmarked views to maintain optimal performance of the view.

Resource availability for selected network view

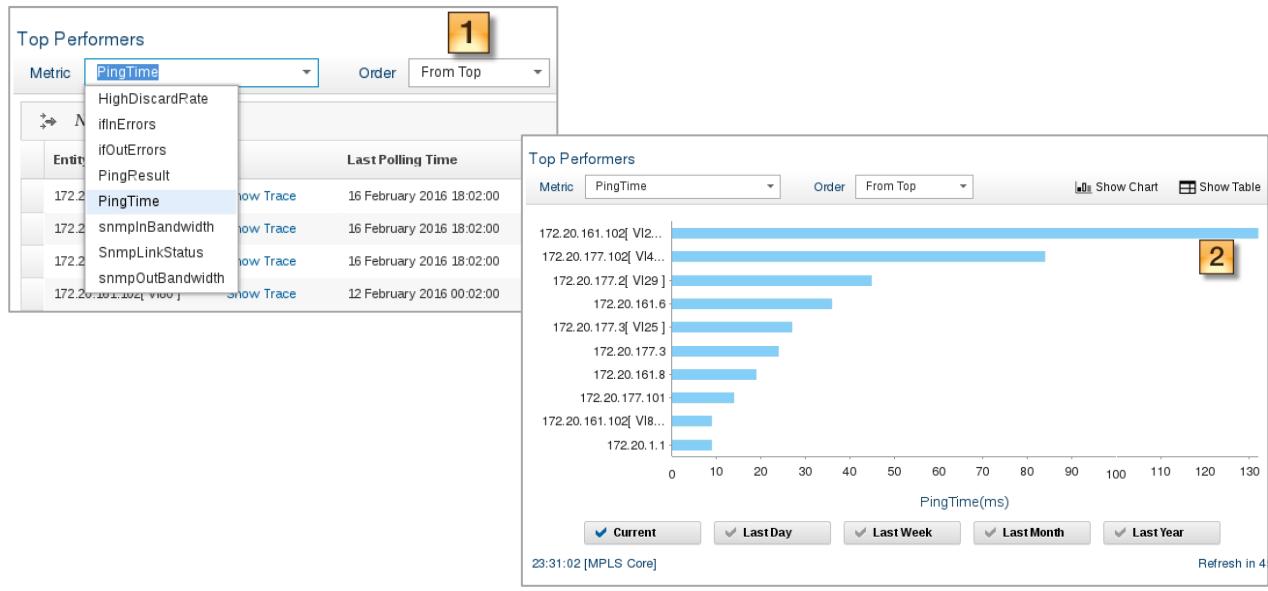


The Network Health Dashboard

© Copyright IBM Corporation 2017

Figure 7-5. Resource availability for selected network view

Top Performers listing for selected network view (1 of 2)



The Network Health Dashboard

© Copyright IBM Corporation 2017

Figure 7-6. Top Performers listing for selected network view (1 of 2)

1. Select from a list of traffic statistics that are being polled and having the poll results stored in the **ncpolldata** database.
2. A graphical view shows the top performers for that statistic (sort order can be changed).

The screenshot shows a web-based dashboard interface. At the top left is a blue header bar with the text "IBM Training". At the top right is the "IBM" logo. Below the header is a title "Top Performers listing for selected network view (2 of 2)". The main area contains a table titled "Top Performers" with columns: Entity Name, Last Polling Time, and Value. The table lists various network entities with their last polling time and a numerical value. At the bottom of the table are five buttons: "Current", "Last Day", "Last Week", "Last Month" (which is highlighted with a red border), and "Last Year". To the right of the table is a timestamp "23:31:30 [MPLS Core]" and a "Refresh in 54" button. At the very bottom of the dashboard are five buttons: "Current", "Last Day", "Last Week", "Last Month" (highlighted with a red border), and "Last Year". A copyright notice "© Copyright IBM Corporation 2017" is at the bottom right.

Entity Name	Last Polling Time	Value
172.20.161.102[VI25]	Show Trace	16 February 2016 21:02:00
172.20.177.102[VI45]	Show Trace	16 February 2016 22:02:00
172.20.177.2[VI29]	Show Trace	16 February 2016 22:02:00
172.20.161.6	Show Trace	16 February 2016 22:02:00
172.20.177.3[VI25]	Show Trace	16 February 2016 22:02:00
172.20.177.3	Show Trace	16 February 2016 22:02:00
172.20.161.8	Show Trace	16 February 2016 22:02:00
172.20.177.101	Show Trace	16 February 2016 21:02:00
172.20.1.1	Show Trace	16 February 2016 21:02:00

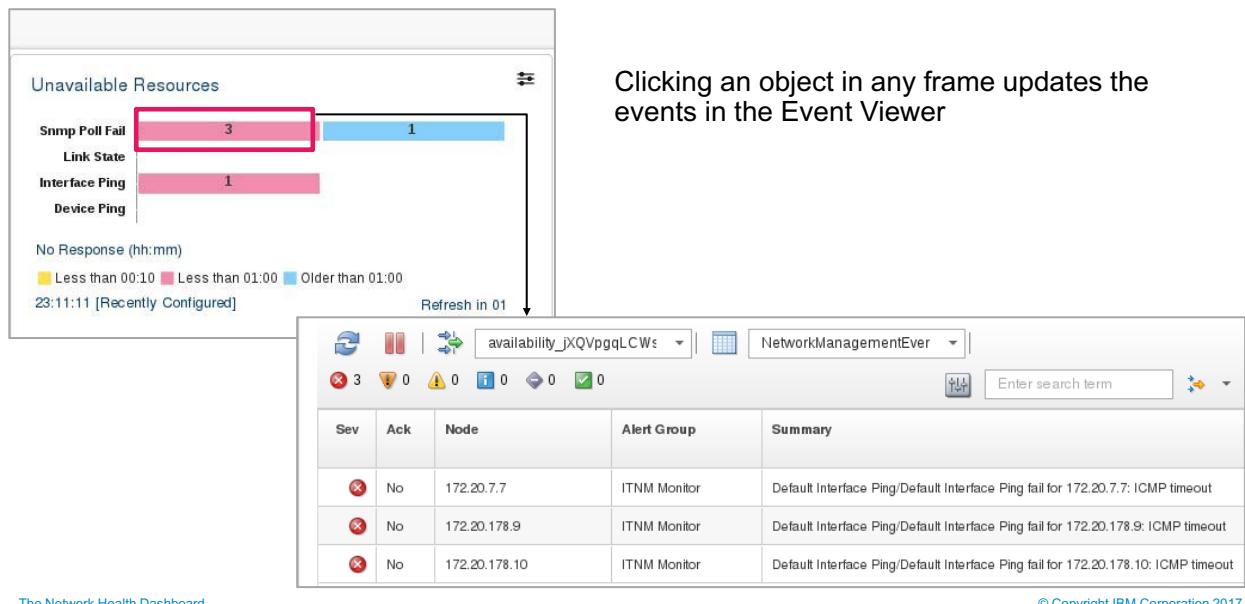
Figure 7-7. Top Performers listing for selected network view (2 of 2)

3. Select **Show Table** to see the same data in tabular format.
4. Change the collection period to see results for a different time period.

Information

Polling data is summarized every 15 minutes. Therefore, a device might be a top performer in the **Current** view but not even be seen in the **Last Day** or **Last Week** view. If the top device in the Current view is not in other views, it is because the device statistics in the current time were not yet rolled into the aggregated data for other longer time periods. After the next summarization of polling data, the other views should then contain data from that device.

Selecting component in a frame changes event data frame



The Network Health Dashboard

© Copyright IBM Corporation 2017

Figure 7-8. Selecting component in a frame changes event data frame

Do events correlate to configuration changes?

- Select the network view that shows devices with recent configuration changes
- Look at the resulting time bar to see whether event changes correspond with large numbers of events
- When events occur immediately after configuration changes, investigate to determine whether a causal relationship exists



Figure 7-9. Do events correlate to configuration changes?

Create custom dashboards for status views of critical services

- Dashboards can draw data from multiple databases
- You configure database connections
- **Wires** are connections that are built between frames or separate pages
- Wires can update target frames or start target pages when data is selected in one frame
- Select widgets from the dashboard library to create tables, status indicators, and graphs on the dashboard page
- Save the dashboard configuration



The Network Health Dashboard

© Copyright IBM Corporation 2017

Figure 7-10. Create custom dashboards for status views of critical services

Dashboards can use event, topology, and log data

- Each operational area in your network operations center can have custom dashboards that pertain to their respective area of responsibility
- Using the widgets library, custom dashboards can be built in minutes

Summary view of virtualized environment, including provisioning, monitoring, patch, and backup

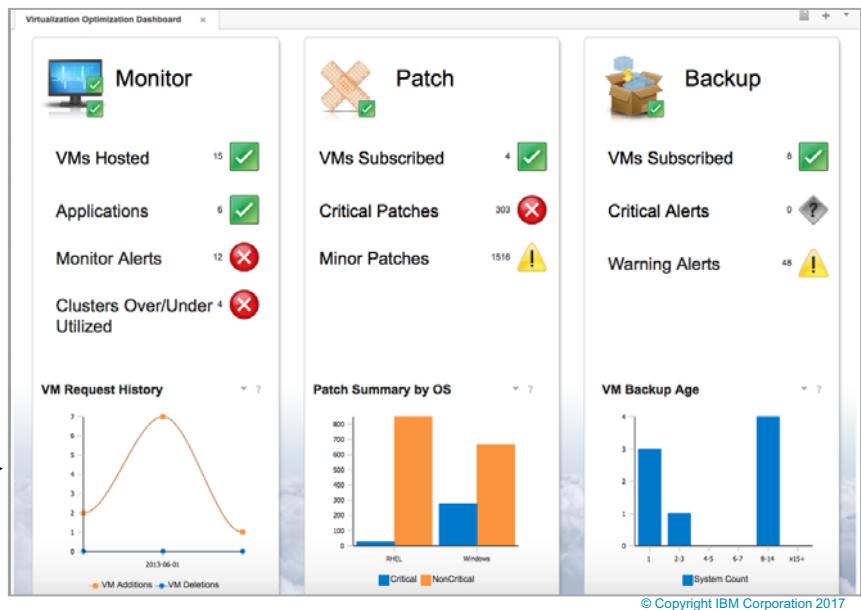


Figure 7-11. Dashboards can use event, topology, and log data

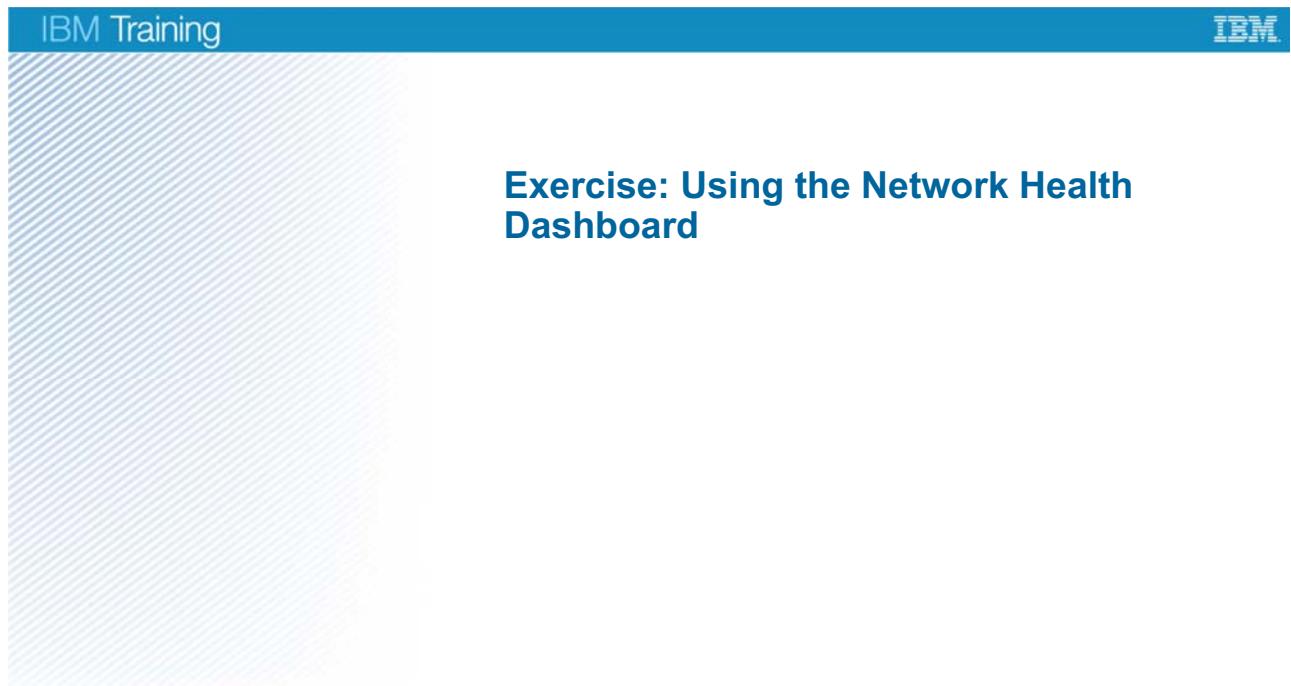


Figure 7-12. Exercise: Using the Network Health Dashboard

Complete the exercises for this unit.

Unit summary

- After you complete this unit, you can use the Network Health Dashboard

[The Network Health Dashboard](#)

© Copyright IBM Corporation 2017

Figure 7-13. Unit summary

Unit 8. Monitoring and polling network devices

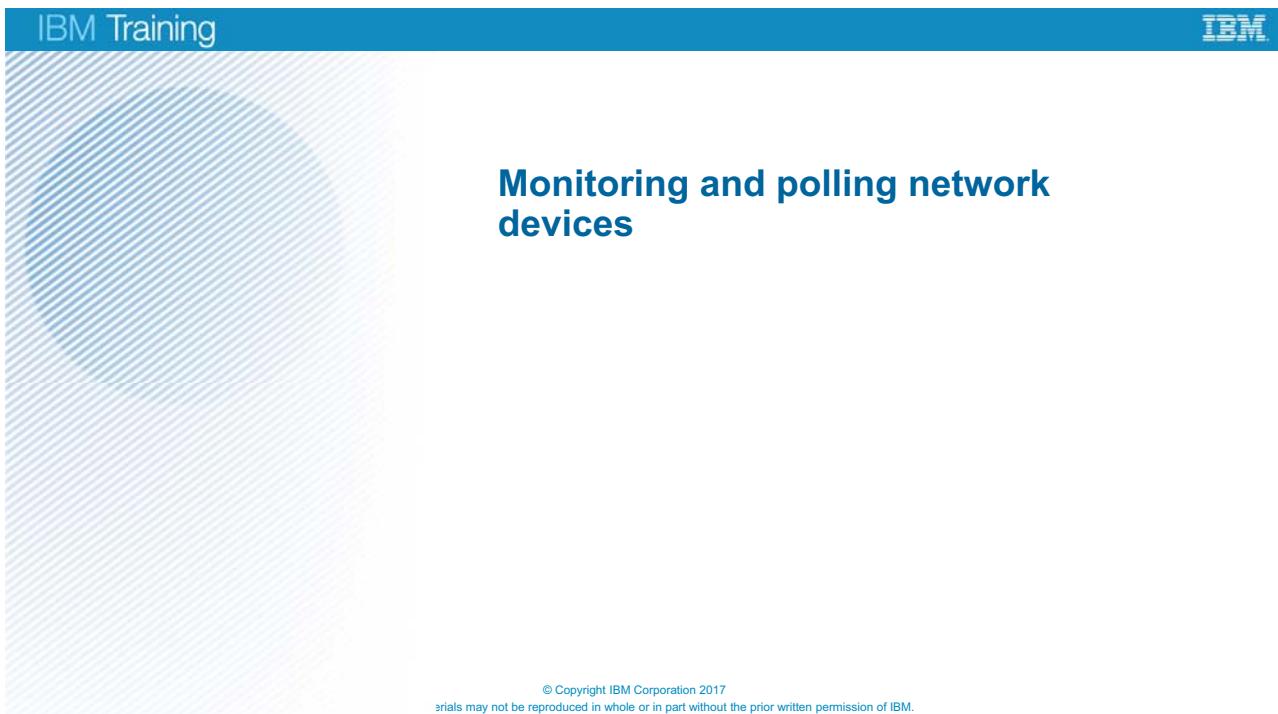


Figure 8-1. Monitoring and polling network devices

Estimated time

01:30

Overview

In this unit, students learn to configure polling of discovered network devices. They also learn to limit the scope of polling, preferably by assigning a polling policy to specific network views.

How you will check your progress

- Complete checkpoint questions
- Complete student exercises

Unit objectives

After you complete this unit, you should be able to do the following tasks:

- Configure normal ping and SNMP link state polling
- Create a custom poll
- Configure adaptive polls
- Create a real-time MIB graph
- Configure distributed polling

Figure 8-2. Unit objectives

8.1. Polling architecture and principles

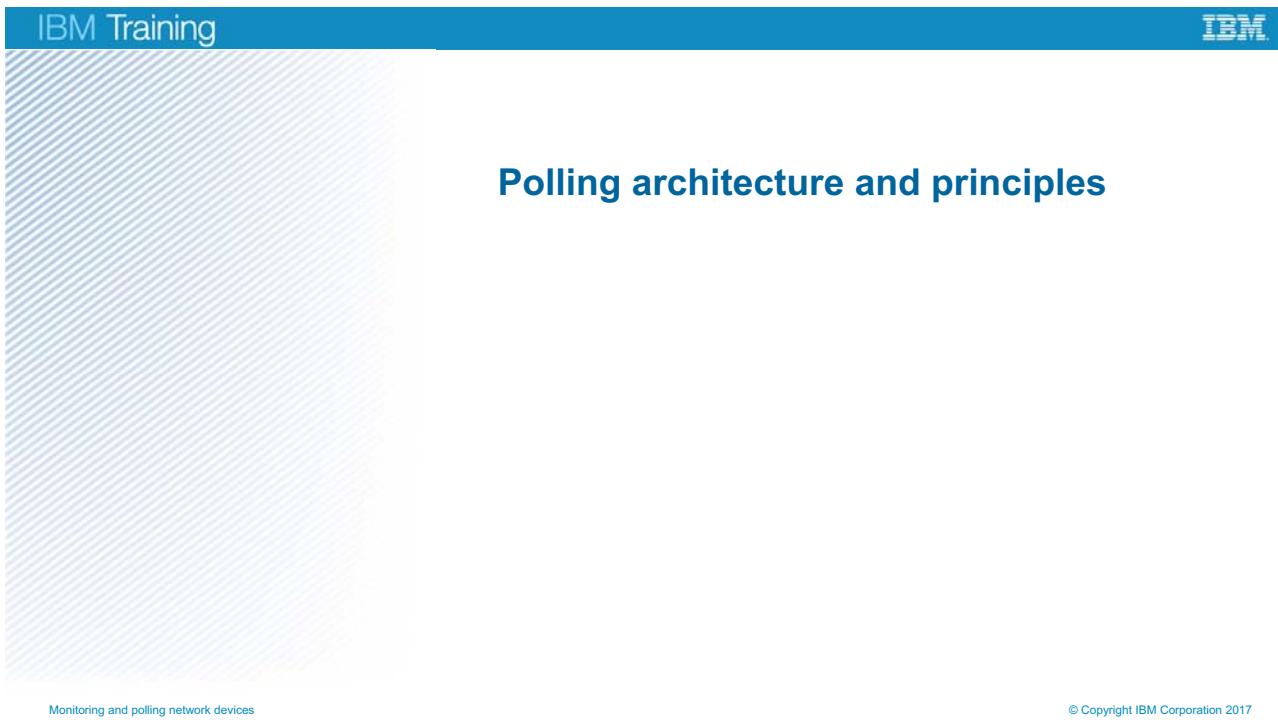
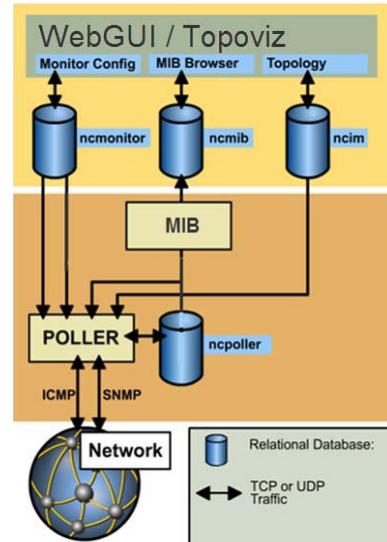


Figure 8-3. Polling architecture and principles

Monitoring architecture

- **NCMONITOR** database contains polling configuration data and is on the same server as the **NCIM** database
- **NCPOLLER** database contains the schedule of actual polls to be sent to network devices
 - The **ncp_poller** engine populates this database and avoids duplicate polls when multiple polling policies and definitions need the same data



[Monitoring and polling network devices](#)

© Copyright IBM Corporation 2017

Figure 8-4. Monitoring architecture

The terms *monitoring* and *polling* are synonymous in this unit. Several databases are used in polling activities:

- The **NCMIB** database contains MIB information that is read from the files in the `$ITNMHOME/mibs` directory.
- The **NCMONITOR** database contains polling definition and polling policy information as configured from the GUI interface.
- The **NCPOLLER** database contains a schedule of the actual polls to be sent to your network devices.

NCMONITOR database

- The installation process creates the **NCMONITOR** database that is used by the GUI and the monitor processes
- The database schema is stored in the following location:
 - `$ITNMHOME/scripts/sql/databasetype/createMonitorConfigDb.sql`
 - The default *databasetype* is **db2**
- The **ncp_poller** process does the following tasks:
 - Reads the policies and definitions from NCMONITOR
 - Schedules the list of polls and timing
 - Writes the values in the NCPOLLER database
- **NCPOLLER** database prevents redundant polls when multiple poll definitions or policies require the same MIB information

Figure 8-5. NCMONITOR database

Previous versions of Tivoli Network Manager supported Informix Dynamic Server (IDS) and MySQL. DB2 is the default database for Tivoli Network Manager 4.2 and later versions. It is more capable of handling large amounts of data. The purpose of reducing the number of supported database platforms is to ensure that more thorough testing is possible before a software release.

How new polls affect the gateway

- When a poller detects a poll threshold violation, an alarm is generated and is sent to the OMNIbus ObjectServer by the **nco_p_ncpmonitor** probe
 - This probe runs on the Tivoli Network Manager server
- Each alarm sent to the ObjectServer has a value set in its **EventId** field
 - The **EventId** is based on the poll definition names
- The Tivoli Network Manager gateway looks at the **EventId** to determine how to process the event
- The gateway enriches the following fields in the ObjectServer's **alerts.status** table:
 - **NmosDomainName**
 - **NmosEntityId**
 - **NmosManagedStatus**

Figure 8-6. How new polls affect the gateway

A Structured Query Language (SQL) script is provided during the installation process to create the new fields for you in an OMNIbus 7.x server if they do not exist.

```
# cd /opt/ibm/netcool/precision/scripts/
# nco_sql -server NCOMS -user root -password '' < ncp_configure_omnibus.sql
```

Newer versions of the OMNIbus ObjectServer already have these fields in the **alerts.status** database.

Polling policies

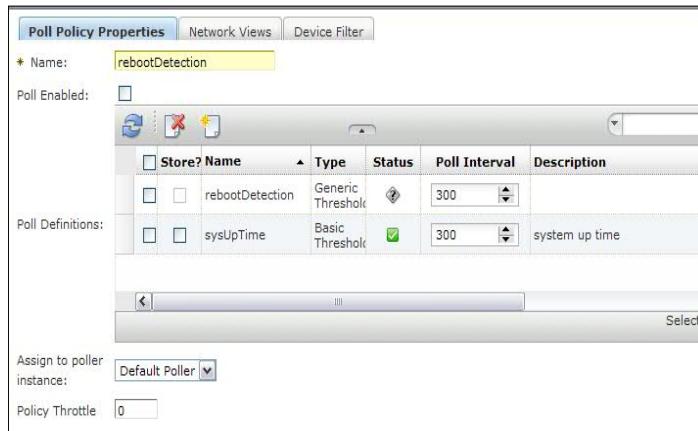
- Poll policies determine:
 - What to poll (**Classes and Scope**)
 - When to poll (**Interval**)
 - How to poll (**Poll Definition**)
 - Where to poll (**Scope**)
 - Information that is collected during the polling
- Poll policies define:
 - The name of the poll policy
 - The poll definition to use
 - The frequency of the poll operation
 - Whether the poll policy is enabled
 - The device classes from which to retrieve information
 - The scope of the poll policy

Figure 8-7. *Polling policies*

One or more policies can be associated with a single poll definition. For example, you might want to poll one set of devices more frequently than another. To do that, you can create two policies with one poll definition. This configuration results in polling different devices with different polling intervals.

Multiple poll definitions per policy (1 of 3)

- Now a policy can be defined with multiple poll definitions that are assigned to them
- Each can have its own polling interval



Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-8. Multiple poll definitions per policy (1 of 3)

You can create polling policies that use multiple polling definitions.

Multiple poll definitions per policy (2 of 3)

The screenshot shows the 'Poll Policy Properties' window. The 'Name' field is set to 'rebootDetection'. Under 'Poll Definitions', there are two entries:

Store?	Name	Type	Status	Poll Interval	Description
<input type="checkbox"/>	rebootDetection	Generic Threshold	<input checked="" type="checkbox"/>	300	
<input type="checkbox"/>	sysUpTime	Basic Threshold	<input checked="" type="checkbox"/>	300	system up time

Below the table, 'Assign to poller instance:' is set to 'Default Poller' and 'Policy Throttle' is set to '0'.

An arrow points from the 'rebootDetection' entry in the main window to the 'rebootDetection' entry in a separate 'Poll Definitions' dialog box.

Poll Definitions

Name	Type
locInCrcErrors	Basic Threshold
memoryPoll	Basic Threshold
rebootDetection	Generic Threshold
SNMP Link State	SNMP Link State
snmpInBandwidth	Basic Threshold
snmpOutBandwidth	Basic Threshold
sysTrafficPoll	Basic Threshold
sysUpTime	Basic Threshold

Selected: 2, Total: 35

Store Poll Data:
Interval(s): ▲

[?](#) OK Apply Cancel

Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-9. Multiple poll definitions per policy (2 of 3)

Multiple poll definitions per policy (3 of 3)

- Many policies can use the same poll definition
- Each poll definition that is specified in a policy can have its own polling interval and storage setting
- Beginning with IBM Tivoli Network Manager 4.2, a polling policy can use several poll definitions
- A poll definition can be assigned only one time to the same policy
- Database table **ncmonitor.poll** does the following tasks:
 - Maps the policies to the poll definitions
 - Includes information about the polling interval and the store data flag

Figure 8-10. Multiple poll definitions per policy (3 of 3)

Polling policy and definition update frequency

- You can use the following properties to configure the frequency that the tables check for updates:

```
monitorconfig.policy.table.refresh.period=60
```

```
monitorconfig.definition.table.refresh.period=60
```

- This period is specified in seconds and represents the minimum time between table refreshes
- If no changes to the status of the table occur, the tables do not refresh
- The administrator has the option of running a manual refresh

- Properties file

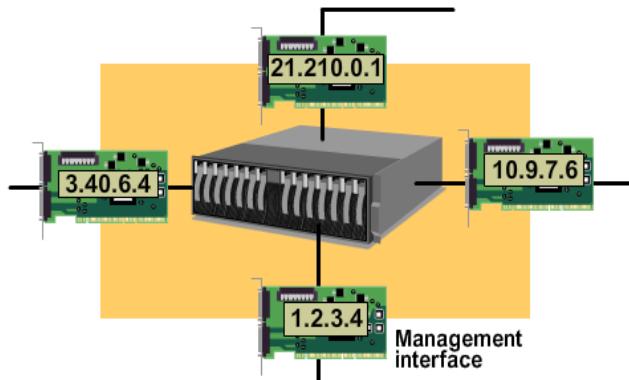
```
$/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/monitorconfig.properties
```

Figure 8-11. Polling policy and definition update frequency

Set the following environment variable to make it quicker to navigate to GUI-related files:

```
GUIHOME=/opt/IBM/netcool/gui/precision_gui
```

Chassis and interface polling



- All chassis polling sends requests to the management interface
- The discovery stitchers designate one of the discovered interfaces as the management interface
- The management IP address is listed in the NCIM **chassis** table
- No interface ping polls are sent to unmanaged interfaces
- No SNMP poll events are raised against unmanaged interfaces

Figure 8-12. Chassis and interface polling

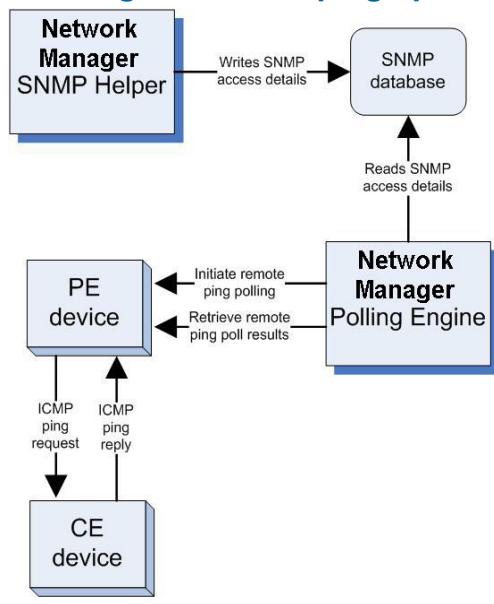
The discovery process designates one interface on the device as the main IP address. Typically, this IP address refers to loopback or the address that is provided in a hosts file or by the DNS server.

Remote ping polling

- Remote ping polling can verify a Multiprotocol Label Switching (MPLS) path between a premise equipment (PE) entity and one or more customer equipment (CE) entities
 - This test occurs over a path that is designated in a virtual routing and forwarding (VRF) table
- Remote ping polling requires SNMP write access to the PE device
- Any device with a VRF table configured is a candidate for remote ping polling
 - Remote ping polling is attempted for each CE device with a connection that is specified in the VRF
 - Currently, supported vendors are Cisco and Juniper

Figure 8-13. *Remote ping polling*

Initiating the remote ping operation



- The SNMP helper writes both read and write access details to the SNMP database
- The poller uses the SNMP write access details to tell the PE device to ping the CE
- The poller uses the SNMP read access details to retrieve the results from the PE device
- If the remote ping polling operation fails, events are raised
 - Indicates a failure in the MPLS path between the PE and the CE across a VRF path

Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-14. Initiating the remote ping operation

Remote ping polling is used to verify connectivity between customer equipment (CE) and premise equipment (PE) in an MPLS network.



Ping enhancements

- Metrics can be collected during ping polling
- Using this option can increase the number of polls that are sent
- You can define the payload size
- These metrics include the following values:
 - Response time
 - Packet loss

Poll Definition Editor - Mozilla Firefox

ibm.com https://mudflap.tivlab.raleigh.ibm.com:16311/ibm/console/hcp_monitor/Template.do?key=3&ac

General	Classes	Interface Filter	Ping
Timeout:	3000	(ms)	
Retries:	2		
Collect Ping Metrics:	<input checked="" type="checkbox"/> Response Time <input type="checkbox"/> Packet Loss		
Payload Size:	<input checked="" type="radio"/> Default(32 Bytes) <input type="radio"/> 32 Bytes		
<small>Payloads of less than 32 Bytes may be dropped by some devices.</small>			

Figure 8-15. Ping enhancements

Topic summary

Having completed this topic, you should be able to do the following tasks:

- List the types of ping polling
- Set a polling policy throttle to limit the number of devices that are polled by a policy

Figure 8-16. Topic summary

8.2. Polling policies

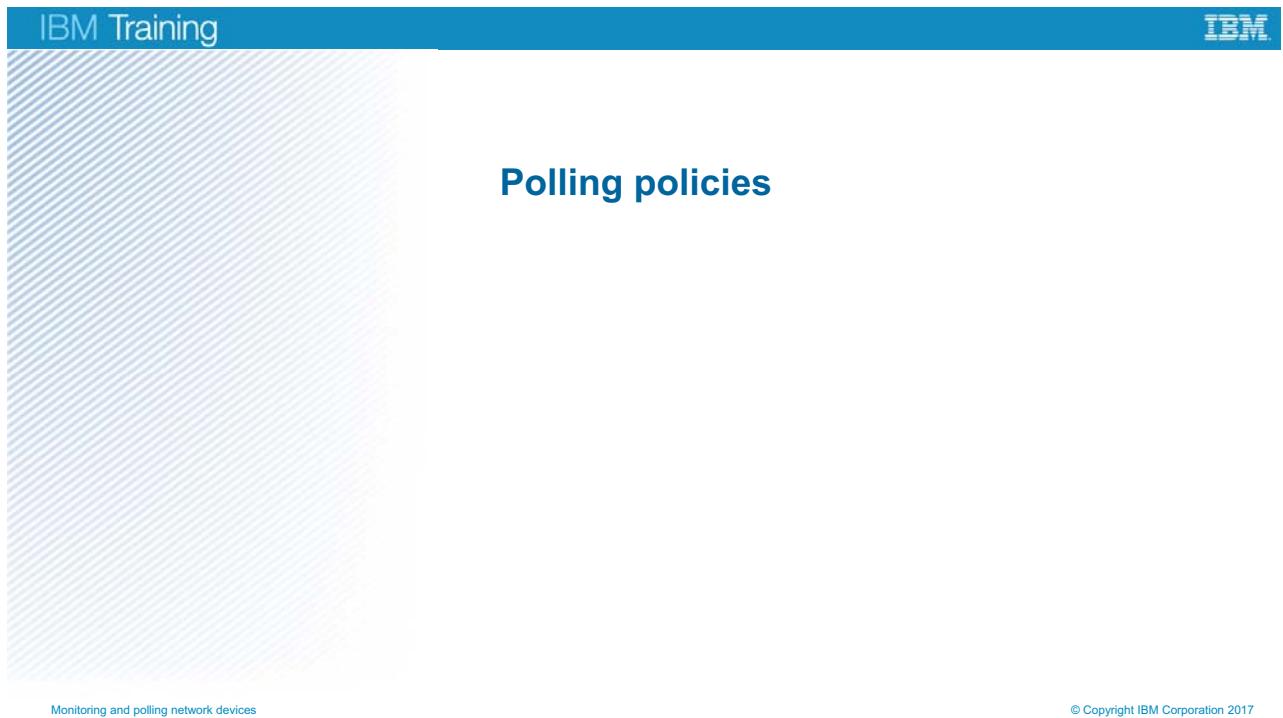


Figure 8-17. Polling policies

Polling GUI (1 of 2)

Green check marks in Enabled and Status columns show poll is working

	Enabled	Status	Name	Poll Definitions	Device Membership
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Default Chassis Ping	Default Chassis Ping	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	bgpPeerState	bgpPeerState	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	bufferPoll	bufferPoll	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	Cisco Remote Ping	Cisco Remote Ping	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	ciscoCPUTotal1min	ciscoCPUTotal1min	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	ciscoCPUTotal5min	ciscoCPUTotal5min	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	ciscoCPUTotal5sec	ciscoCPUTotal5sec	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	ciscoEnvMonFanState	ciscoEnvMonFanState	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	ciscoEnvMonSupplyState	ciscoEnvMonSupplyState	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	ciscoEnvMonTemperatureState	ciscoEnvMonTemperatureState	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	ciscoMemoryPctgUsage	ciscoMemoryPctgUsage	Devices
	<input type="checkbox"/>	<input type="checkbox"/>	ciscoMemoryPool	ciscoMemoryPool	Devices

Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-18. Polling GUI (1 of 2)

In previous versions, the **Network Polling** option was under **Administration**, but in version 4.2 it is under **Network / Administration**.

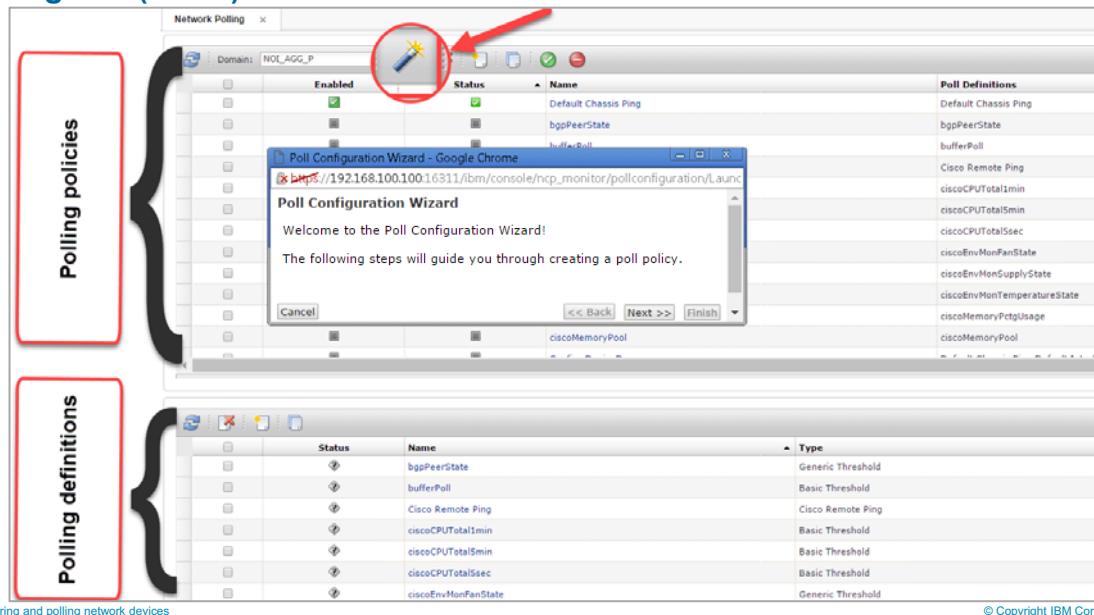
Polling GUI (2 of 2)

Figure 8-19. Polling GUI (2 of 2)

The top portion of the window is to **Configure Poll Policies**. The bottom portion of the window is to **Configure Poll Definitions**.

The polling GUI in Tivoli Network Manager 4.2 does not have labels that distinguish between the polling policies and polling definitions. The polling policies are in the top section, and the polling definitions are in the bottom section.

The **Poll Configuration Wizard** icon creates a poll policy. However, the only way to create a poll definition is to click the new definition icon in the bottom section of the polling GUI.

Tivoli Network Manager does two kinds of polling:

- ICMP or ping polling
- SNMP polling

Default ping polling policies

Chassis poll

- Uses the **Default Chassis Ping** poll definition to do ping operations on all network devices
 - It uses the chassis IP address and is subject to the limitations of any class filters
- This poll policy is the only one enabled by default

Interface poll

- Uses the **Default Interface Ping** poll definition to do ping operations on all interfaces that are within a main node
 - These interfaces must have a valid IP address and match class and scope settings

End Node poll

- Uses the **End Node Ping** poll definition to do ping operations on all end nodes

Figure 8-20. Default ping polling policies

Default SNMP and remote ping polling policies

- **SNMP Link State** poll policy uses the **SNMP Link State** definition to verify the administrative and operational status of network devices
- SNMP remote ping polling
 - **Cisco remote ping** uses the **Cisco remote ping** poll definition to confirm the availability of Multiprotocol Label Switching (MPLS) paths between Cisco provider edge (PE) and customer edge (CE) devices
 - **Juniper remote ping** uses the **Juniper remote ping** poll definition to confirm the availability of MPLS paths between Juniper PE and CE devices

Figure 8-21. Default SNMP and remote ping polling policies

SNMP link state polling checks two MIB variables to determine the status of network interfaces.

- **ifAdminStatus** has a status of 0 or 1. A zero indicates that the interface is administratively disabled. A one indicates that the interface is administratively enabled.
- **ifOperStatus** has a status of 0 or 1. A zero indicates that the interface is inactive (down). A one indicates that the interface is active (up).

Both of these varbinds factor into when an alarm condition occurs.

- On a normal interface, an alarm occurs when **ifAdminStatus=1** and **ifOperStatus=0**.
- On an Integrated Services Digital Network (ISDN) interface, an alarm occurs when **ifAdminStatus=1** and **ifOperStatus=1**. When an ISDN interface becomes active, it indicates that a network outage is occurring.

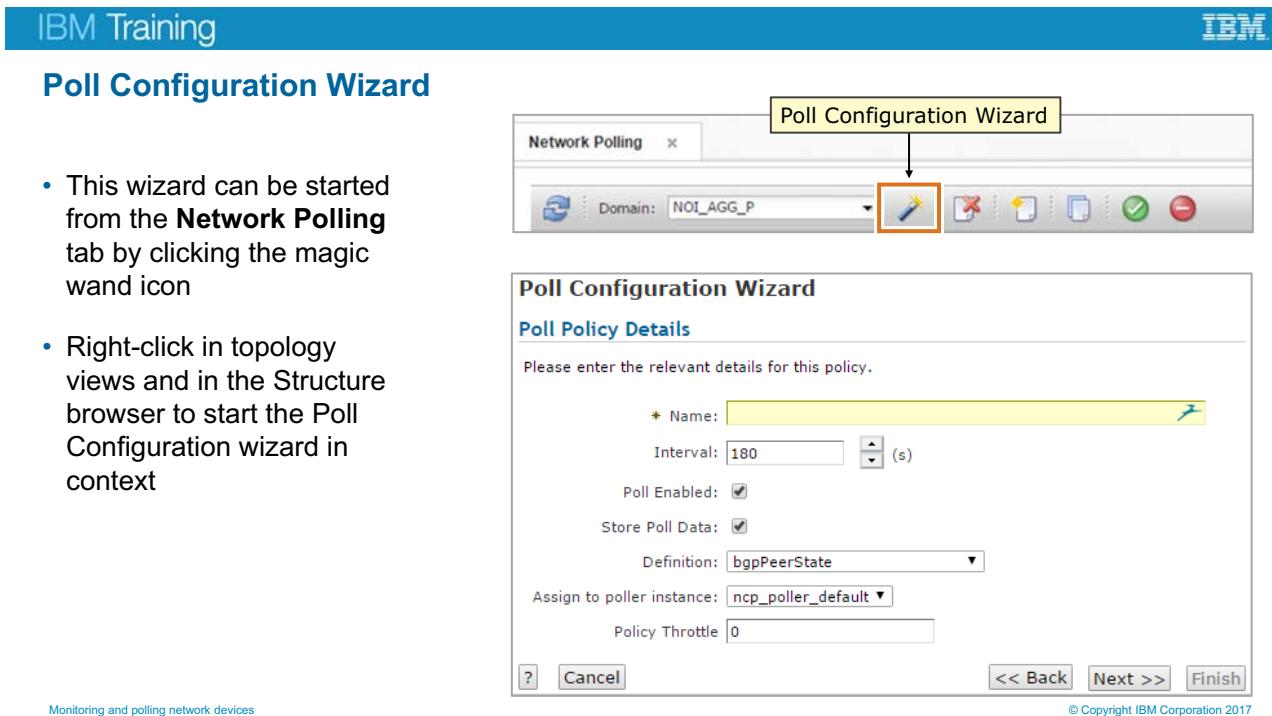
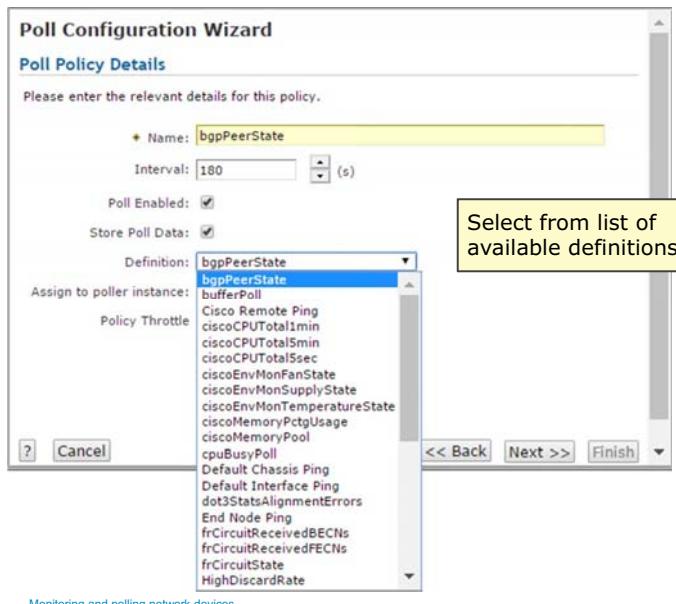


Figure 8-22. Poll Configuration Wizard

The wizard guides you through the process of configuring a poll policy.

Enter poll policy details



1. Enter a policy name (no spaces)
2. Enter polling interval
3. Enable the poll
4. Choose whether to store poll data
5. Select an existing poll definition or create a definition
6. Assign the policy to a poller instance (two poller instances are created by default)
7. Determine whether to set a policy throttle
8. Click **Next**
9. Set scope of polling policy
10. Click **Finish**

© Copyright IBM Corporation 2017

Figure 8-23. Enter poll policy details

Ways to configure scope

- You can limit the scope of a polling policy by one of the following methods:
 - Assign to a network view
 - Limit with a device-based SQL filter
 - Assign to a class of devices (categories are in the network views)
 - For interface-related polls, limit the poll to a certain interface
- Interface filtering
 - Enabled in the poll definition
 - Defined in the polling policy



Figure 8-24. Ways to configure scope

Scope limitations that are based on device filtering can involve complex SQL filters. It is much easier to create a network view and then assign the policy to a network view. Scope tells **ncp_poller** what to poll.

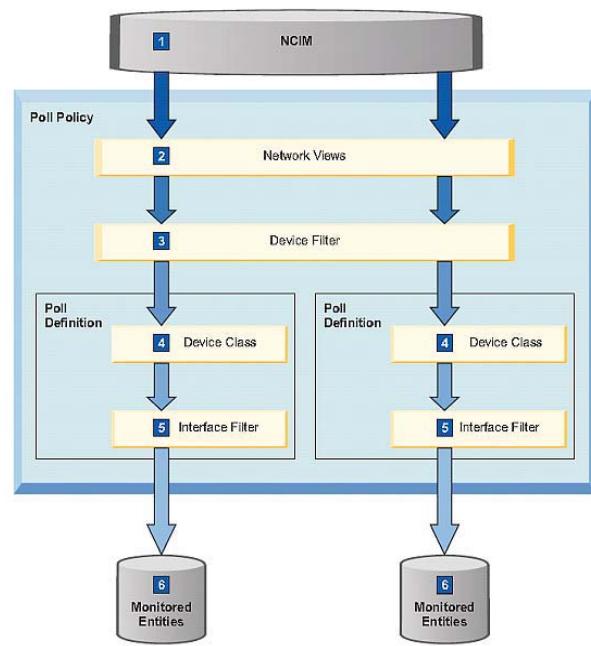
Tivoli Network Manager does two kinds of ping polling:

- Chassis: Must always have a valid value for **accessIPAddress**.
- Interface: Can have a valid value for **accessIPAddress**. However, Network Manager polls only interfaces that have an IP address.

The **accessIPAddress** is the IP address that is used by the polling engine to poll a specific chassis or interface. Some interfaces on a device might not be running the Internet Protocol (IP). Tivoli Network Manager does not poll these interfaces.

Deciding what gets polled

1. Start with all discovered devices
2. Restrict polling to any associated network views
3. Apply device filters
4. Apply class filters
5. Apply valid interface filters
6. Poll entities that match preceding criteria



Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-25. Deciding what gets polled

This diagram illustrates the appropriate order in which to filter polling.

Network Views filter

- Beginning with Tivoli Network Manager 4.2, you can restrict polling policies to defined Network Views
- Using Network Views to restrict the scope of polling policies aligns your polling policies with business requirements
- You can quickly go to the specified Network Views to see the devices that are associated with the polling policies

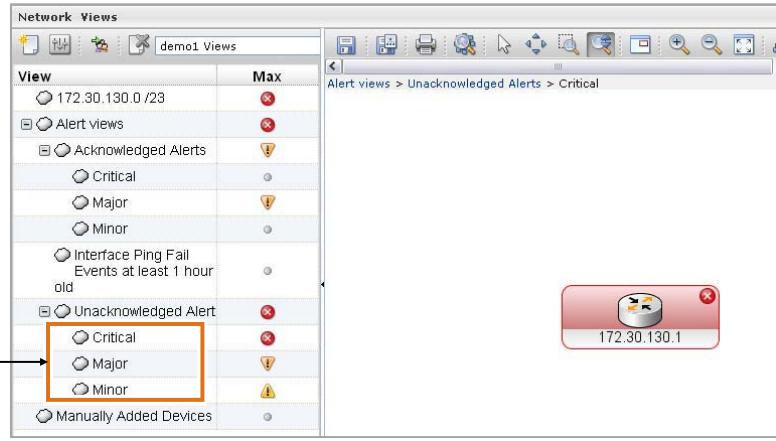


Figure 8-26. Network Views filter

Benefits of applying polling policies to Network Views

- It is easy to see what is being polled by each policy
- It is easy to customize polling to different parts of your network based on geography, business function, device types, or other fields that are used to create Network Views
- It makes it easy to create adaptive polling for a Network View based on event criteria
- It avoids the need for complex SQL filters

By applying a polling policy to one of these groups, you can easily configure more monitoring that is based on the status of a device



Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-27. Benefits of applying polling policies to Network Views

Tivoli Network Manager features simplified polling policy definition and improved feedback.

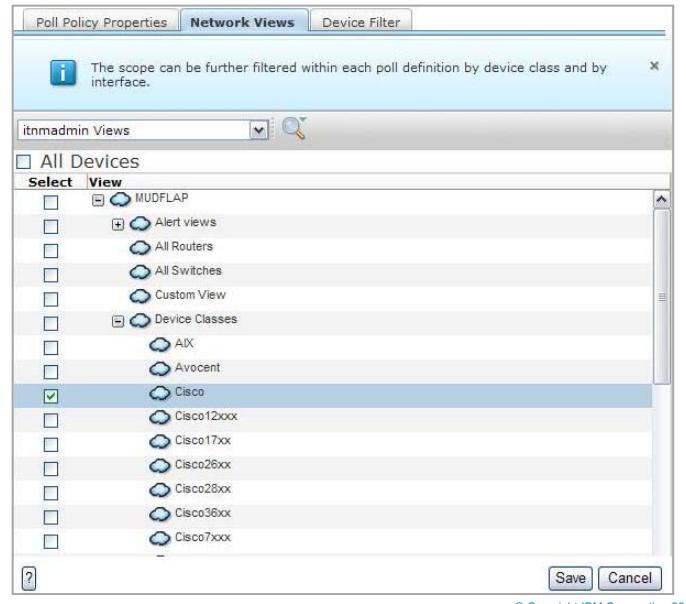
- In previous releases of Tivoli Network Manager, the polling scope for actively monitoring a set of network resources was defined with SQL-like filters. These powerful filters are complicated for some users.
- With Tivoli Network Manager, a polling policy can be assigned to one or more Network Views, with more device-level scoping possible. By assigning polling policies to event-based network views, you can create polls that can respond to specific network conditions.
- In a network view, you can easily see the devices with which a polling policy is associated.
- Reports help the network administrator troubleshoot polling issues.

These polling features enable the network administrator accomplish the following tasks:

- Verify that key entities of the network are being monitored
- Understand why an entity is not polled successfully
- Adapt polling due to conditions on the network

Assigning polling policies to device classes

- The preferred way to assign a scope to polling is to assign a polling policy to a network view
- Tivoli Network Manager automatically creates a Network View for each device class
 - You can create polling policies that are assigned to one or more classes of devices
- If you are using an interface poll, you can further restrict the scope of a poll to certain types of interfaces



[Monitoring and polling network devices](#)

© Copyright IBM Corporation 2017

Figure 8-28. Assigning polling policies to device classes

Assigning a policy to a network view makes it easy to visualize which devices are polled by a policy. Always begin limiting a poll by assigning it to network views before you add other types of filtering. Assigning a polling policy to an event-based network view results in adaptive polling.

Device Filter tab

You can further limit the scope of a polling policy by defining a device filter at the policy level that uses any field in an available discovery table

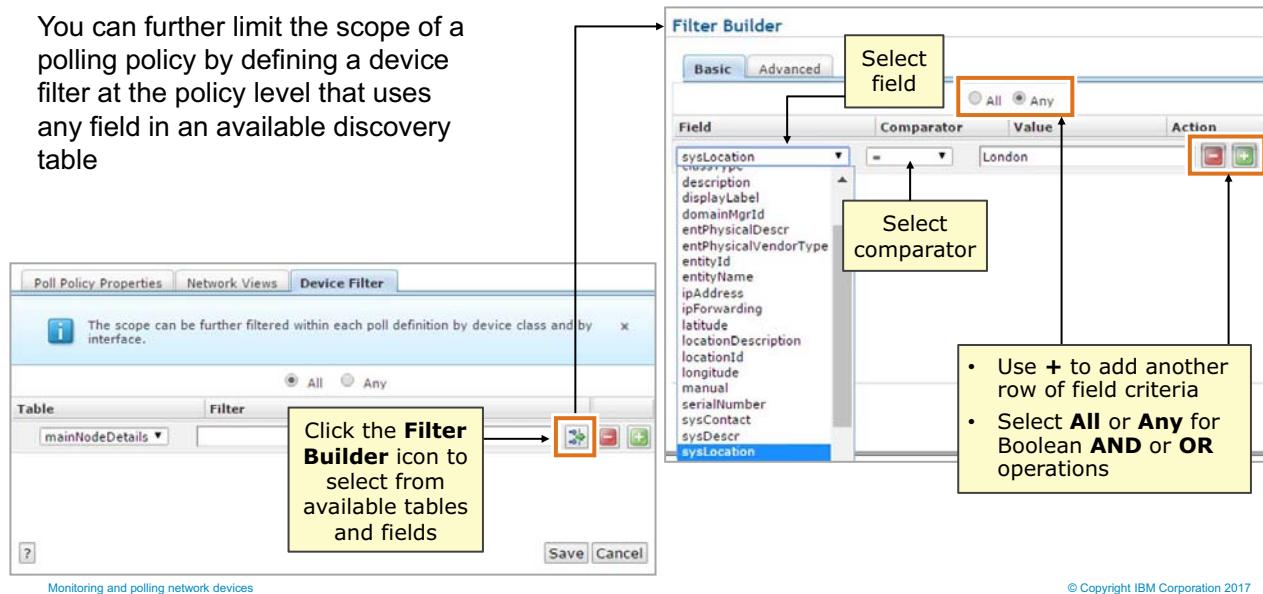


Figure 8-29. Device Filter tab

To create a filter that is based on criteria other than a Network View, do the following steps:

1. On the **Device Filter** tab of the polling policy, select any available discovery table.
2. Click the **Filter Builder** icon and then select fields from the specified table.
3. Select an appropriate **Comparator** and specify the **Value**.
4. If you need to use more fields, click the plus sign (+) to add another row. Select its comparator and value, and then click **All** (Boolean AND) or **Any** (Boolean OR) to define how the rows relate to one another in the **Filter Builder**.



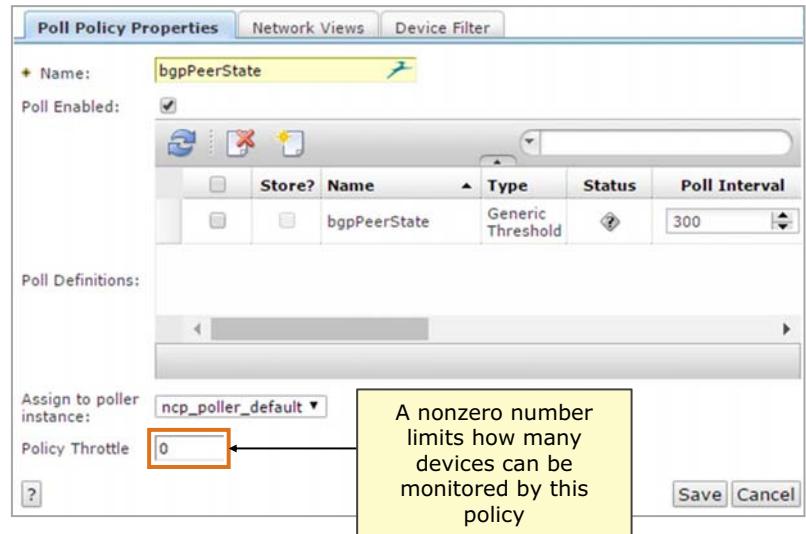
Information

The **Basic** tab of the Filter Builder can have multiple fields, but you must choose **All** or **Any**.

If you need a more complex filter with parenthetical statements and a combination of Boolean AND and OR statements, type the complex expression in the **Advanced** tab.

Policy throttle

- When the number of devices that match the policy criteria exceeds the nonzero **Policy Throttle** value, the poller takes the following actions:
 - An alert is sent to the ObjectServer
 - A message is written to the polling trace file
 - The policy status is set to **throttle limit exceeded**
 - Only the specified number of devices are polled



Monitoring and polling network devices

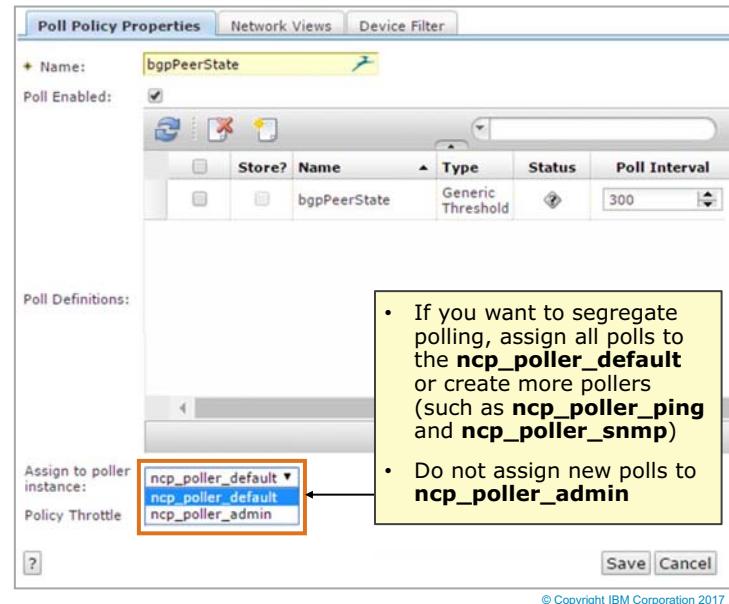
© Copyright IBM Corporation 2017

Figure 8-30. Policy throttle

You can limit the number of devices that are polled by a policy. The purpose of this limitation is to control the network bandwidth that is used by polling activities.

Assign a distributed poller to the policy

- All poller instances must run on the Tivoli Network Manager server
- The **ncp_poller_admin** instance makes cache updates, drops partitions, and supports the SNMP MIB Grapher
 - Do not assign polls here
- The **ncp_poller_default** instance is started with the `-noadmin` option, and is used for SNMP and ping polling
- You can add other polling instances
- Do not remove the entry for the **ncp_poller_admin** instance from `CtrlServices.cfg`
 - There must always be one poller started with the `-admin` option



Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-31. Assign a distributed poller to the policy

Distributed polling does not refer to putting pollers on multiple boxes. Instead, it means having multiple instances of the poller on the same server. The default configuration for Tivoli Network Manager has two poller instances:

1. The **ncp_poller_admin** instance starts with the `-admin` option and handles database operations, record-locking, cache updates, and also supports the real-time SNMP MIB Grapher.
2. The **ncp_poller_default** instance starts with the `-noadmin` option and does ICMP and SNMP polling. If you create more polling instances, they should also have the `-noadmin` option.

Do not change the default configuration for the **ncp_poller_admin** poller. Changes might interfere with its database functions.

However, you can edit the `CtrlServices.cfg` file in the `$NCHOME/etc/precision` directory to remove the **ncp_poller_default** instance and add in other polling instances. For example, you might create two pollers that are called **ncp_poller_ping** and **ncp_poller_snmp** to segregate ICMP and SNMP polling operations to the polling instance that corresponds with each binary.

When creating a new poller, you must first register that poller with the following command:

```
ncp_poller -domain DomainName -register -name polling_instance_name
```

When you delete an existing polling instance, use the following command:

```
ncp_poller -domain DomainName -deregister -name polling_instance_name
```

If no active policies are assigned to a poller, the poller is automatically deregistered. If you want to deregister a poller, you must first assign all active polling policies to another poller.

**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 such as `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 such as `NcPollerSchema.POLLERNAME.DOMAIN.cfg`. A poller instance uses the most specific configuration file available.

Verify the polling policy configuration

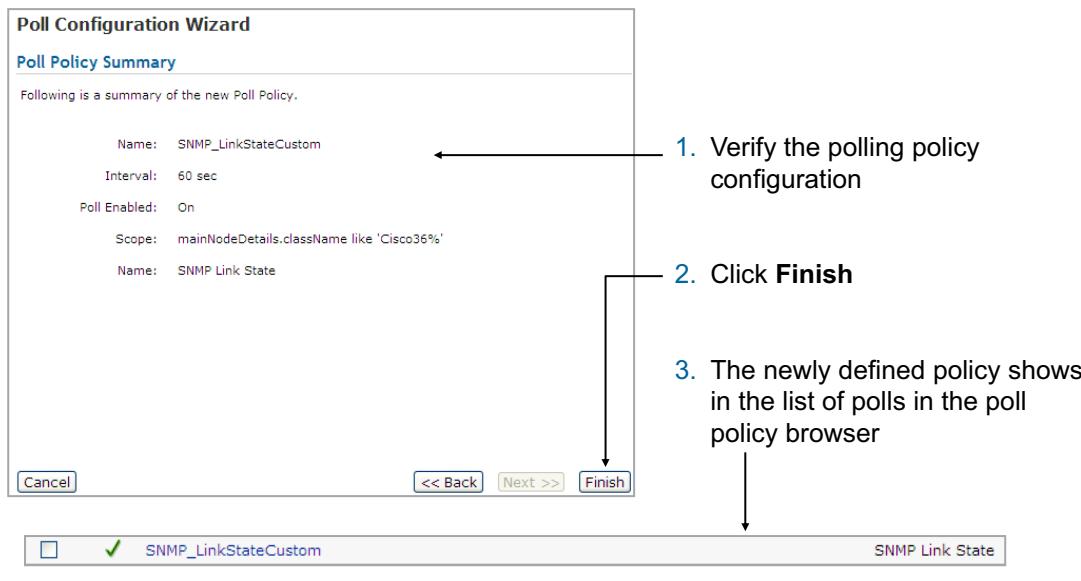


Figure 8-32. Verify the polling policy configuration

By default, the polling helpers **ncp_dh_snmp** and **ncp_poller** do not use SNMP **GetBulk**. They use **GetNext**. In some cases, it can be more efficient to use **GetBulk** to send multiple SNMP queries in one request. You can force Tivoli Network Manager to try to use **GetBulk** when possible.

1. Set **UseGetBulk** to **1** in `config.properties` in `NcPollerSchema.cfg` to enable SNMP v2 GetBulk polling.
2. GetBulk requests are sent by **ncp_dh_snmp** and **ncp_poller** when SNMP v2 or v3 is used.
3. Set **m_UseGetBulk** to **0** in **snmpStack.accessParameters**. This setting filters devices so that they do not use the global setting.

You might need to disable GetBulk for older devices. Go to the **DefaultGetBulkMaxReps** section of the `config.properties` file to adjust the value of **max-repetitions**.

Device membership link

Click the **Devices** link for a polling policy to see a list of devices that are polled by that policy

The screenshot shows the Network Polling interface with the following details:

- Left Panel:** A table of poll definitions. One row is selected, highlighted with a red box around the "Devices" link in the "Device Membership" column.
- Right Panel:** A table of devices. A red box highlights the first few rows (SYD-Core-01, BRU-PE-03, WAS-Core-01, BRU-PE-02, NYC-Core-01, LON-Core-01) with the annotation "These devices are polled according to the selected policy".

Entity Name	IP Address	Device Type
MyManuallyAddedDevice1.example.class.com	10.10.255.47	Chassis
SYD-Core-01	10.10.255.6	Chassis
BRU-PE-03	10.10.255.7	Chassis
WAS-Core-01	10.10.255.8	Chassis
BRU-PE-02	10.10.255.9	Chassis
NYC-Core-01	10.10.255.10	Chassis
LON-Core-01	10.10.255.11	Chassis
BRU-NAS-02	10.10.255.12	Chassis
BRU-CPE-01	10.10.255.13	Chassis
BRU-NAS-01	10.10.255.14	Chassis
MOS-Core-01	10.10.255.7	Chassis
PAR-Core-01	10.10.255.2	Chassis
BRU-ACS-01	10.10.255.11	Chassis
BRU-Core-01	10.10.255.1	Chassis
host1.csite.edu	192.168.100.100	Chassis
BRU-SW-01	10.10.255.10	Chassis

Figure 8-33. Device membership link

Enhanced poll policy status

The status column in the policy manager shows icons that represent six possible values for status

- The display updates automatically
 - Unknown
 - No error
 - Error with at least one poll definition
 - Error with the policy scope, so policy is disabled
 - Policy scope throttle limit disabled
 - Poller processing recent policy update
 - Tooltips display error message

Icon	Description
	The status is unknown because the poll policy has not been run yet.
	No error
	There is an error with at least one associated poll definition. The poll policy is now disabled. The poll definition error must be fixed before the poll policy can be enabled again.
	There is an error with the policy scope for the poll policy. This is probably due to an error in an SQL statement used to define the scope for the poll policy. The poll policy is now disabled. The policy scope error must be fixed before the poll policy can be enabled again.
	The policy scope throttle limit has been exceeded. The throttle limit value limits the membership size for a poll policy to a predefined value. The poll policy is still enabled.
	The policy configuration has been updated but the poller has not yet processed the update.

Figure 8-34. Enhanced poll policy status

Tivoli Network Manager periodically checks for changes to poll policies and definitions. If any of these policies or definitions change, their status is evaluated and the status column in these tables is updated. You can configure the period at which Tivoli Network Manager scans for changes.

Topic summary

Having completed this topic, you should be able to do the following tasks:

- List the types of ping polling
- Explain how remote ping polling works
- Set a polling policy throttle to limit the number of devices that are polled by a policy

Figure 8-35. Topic summary

8.3. Polling definitions

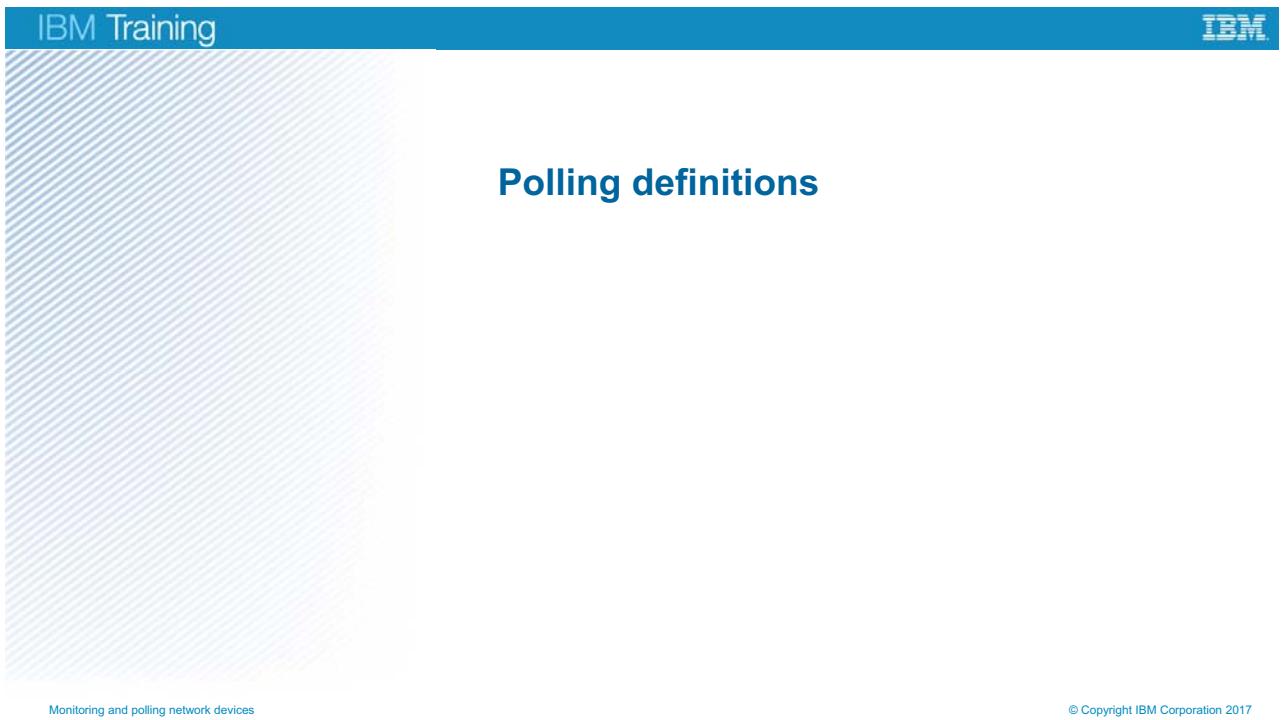


Figure 8-36. Polling definitions

Polling definition

For each poll policy, you must select one or more poll definitions for the policy to use

From the lower portion of the **Network Polling** GUI, you can do these tasks:

- Show a list of existing definitions and their definition type
- Add, delete, or copy definitions
- Edit or specify the following poll definition properties:
 - The name of the poll definition
 - The type of the poll definition
 - The description for the event
 - The event severity level
 - For SNMP threshold poll definitions, the threshold settings for both generating and clearing an alert



	Status	Name	Type
		frCircuitReceivedFECNs	Basic Threshold
		frCircuitReceivedBECNs	Basic Threshold
		End Node Ping	Chassis Ping

Figure 8-37. *Polling definition*

Poll definition types

- Ping polling mechanism
 - **Chassis Ping** pings main node devices
 - **Interface Ping** pings interfaces within devices
- SNMP polling mechanism
 - **Basic Threshold** polls a single MIB variable
 - **Generic Threshold** polls multiple MIB variables and compares values
 - **SNMP Link state** checks the administrative and operational status of links
 - **Cisco Remote Ping** checks the availability of devices with Cisco-specific MIBs
 - **Juniper Remote Ping** checks the availability of devices with Juniper-specific MIBs

Figure 8-38. Poll definition types

Select poll definition type

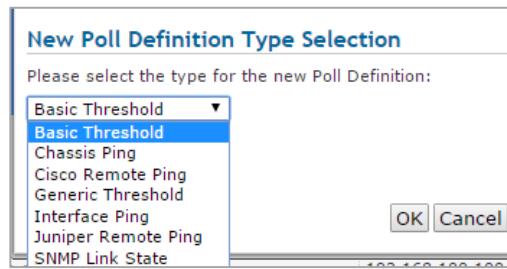


Figure 8-39. Select poll definition type



Set general polling properties

- Data labels are used to gather data for reporting
- Data units include the following values:
 - Counts (#)
 - Percentage (%)
 - Specific units of measure such as bytes or packets
- **Event severity** is the severity that is assigned to an event when a trigger threshold occurs

Figure 8-40. Set general polling properties

Data labels group polling results together for reporting purposes. Some reports that use labels include:

- **Router Health Summary**
 - memoryPercentageUsage
 - cpuBusy
- **Device Ingress Traffic Summary**
 - snmpInBandwidth
 - ifInDiscards
 - ifInErrors
- **Device Outgress Traffic Summary**
 - snmpOutBandwidth
 - ifOutDiscards
 - ifOutErrors
- **Device Availability**
 - sysUptime

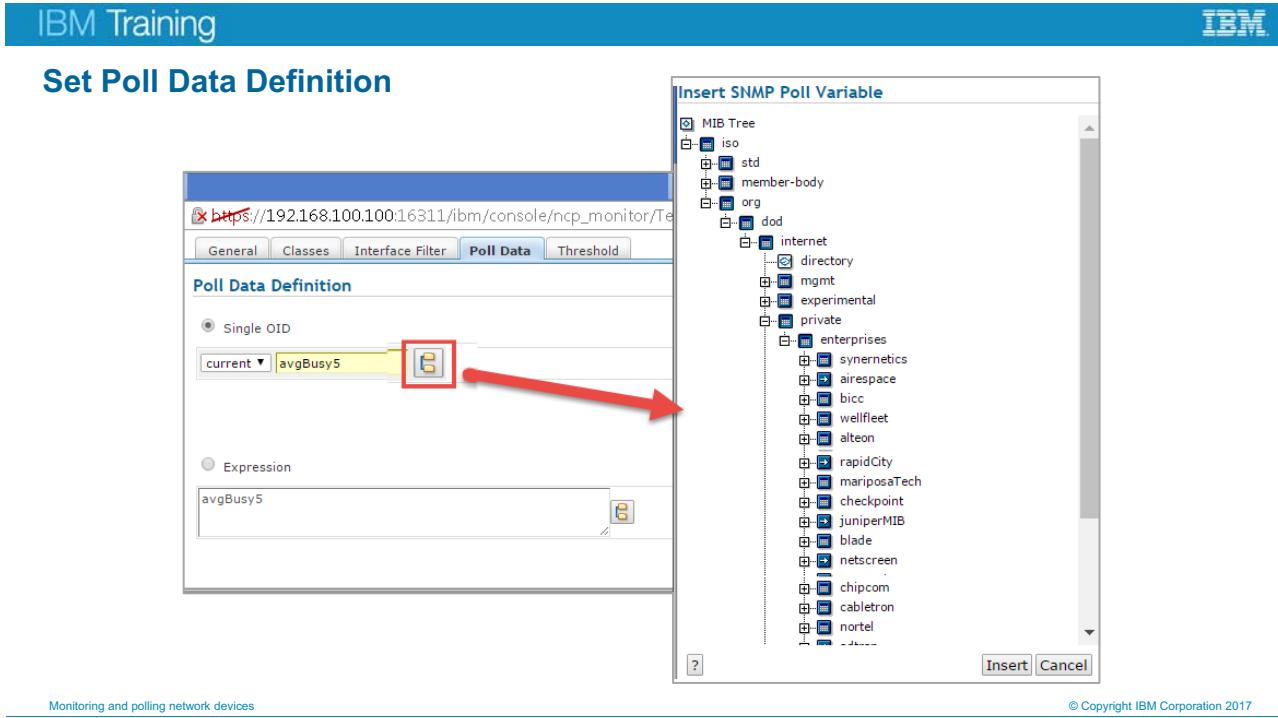


Figure 8-41. Set Poll Data Definition

Use the MIB browser icon to select varbind name or type value into the box.

Set polling thresholds and event messages

Trigger Threshold:

1 current avgBusy5 >= 2 80 3

Trigger threshold determines what triggers a problem event.

Description

4

Clear Threshold:

1 current avgBusy5 < 2 80 3

Clear threshold determines when a resolution event is created.

Description

4

?

Save Cancel

Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-42. Set polling thresholds and event messages

For each trigger, you must set two thresholds.

- The **Trigger Threshold** determines the polling thresholds that trigger an event to be sent to the ObjectServer.
- The **Clear Threshold** defines the conditions under which a resolution (**Type=2**) event is sent to the ObjectServer. After the ObjectServer receives the clearing or resolution event, the **Generic Clear** trigger on the ObjectServer clears the previous problem (**Type=1**) from the ObjectServer.

The text and values that are specified in the **Description** of each section go into the **Summary** field of the event in the ObjectServer.

Interface filtering in poll definition properties

- In the poll definition properties, you can set class and interface filters
- This example definition limits the snmpInBandwidth poll to only polling interfaces that have a defined speed

The figure consists of two side-by-side screenshots of a web-based configuration interface for a poll definition.

Left Screenshot (General Properties):

- Name: snmpInBandwidth
- Type: Basic Threshold
- Event ID: inbandwidth
- Event Severity: 3
- Description: (empty)
- Data Label: snmpInBandwidth

Right Screenshot (Interface Filter):

Shows the "Interface Level Filter" configuration:

- Filter Type: All (radio button selected)
- Table: interfaces
- Filter: ifSpeed <> 0
- Table: interfaces
- Filter: ifSpeed <> 4294967295

Figure 8-43. Interface filtering in poll definition properties

Poll definitions table

- Poll definitions status has three possible values:

Icon	Description
	The status is unknown because the poll definition has not been run yet.
	No error. Poll definition has been run without error.
	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 definition description can show or hide long descriptions:

Status	Name	Type	Description
	snmpOutBandwidth	Basic Threshold	
	snmpInBandwidth	Basic Threshold	Measures interface bandwidth as percentage ...less

Figure 8-44. Poll definitions table

Cross-check to see impact of definition deletion

When you delete a poll definition, Tivoli Network Manager checks to see which poll policies this deletion affects

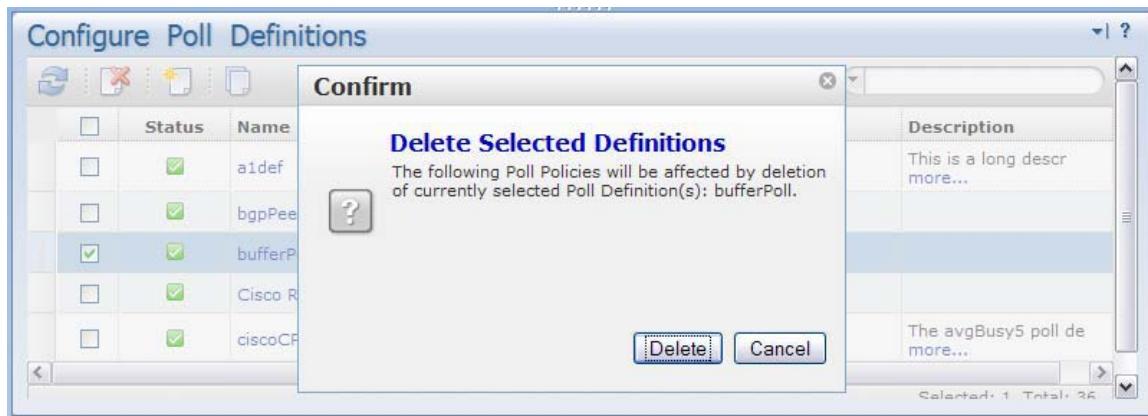


Figure 8-45. Cross-check to see impact of definition deletion

When you delete a poll definition, Tivoli Network Manager shows you a list of all poll policies that use that definition. The definition is deleted only after you confirm the deletion.

Adding new MIBs

- Copy vendor-specific MIB files and any MIBs that they require to **\$ITNMHOME/mibs**
 - MIB files must have a ***.mib** extension
 - Restart the **ncp_mib** process
 - **pkill ncp_mib** (equivalent to a **kill -15**)
 - The **ncp_ctrl** process automatically restarts **ncp_mib** process
 - View **ncp_mib.domainName.log** for errors
 - Test the functions of the SNMP MIB browser

Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-46. Adding new MIBs

Example: UCD MIB set

Topic summary

Having completed this topic, you should be able to do the following tasks:

- Create a poll definition with the poll definition wizard
- Interpret the status icons for polling policies and definitions

Figure 8-47. Topic summary

8.4. Manage and unmanage devices and interfaces



Figure 8-48. Manage and unmanage devices and interfaces

NmosManagedStatus values

- The managed status of devices is in the **ncimCache.managedStatus** table
- Values for **NmosManagedStatus**
 - 0 – Managed
 - 1 – Unmanaged from the GUI
 - 2 – Unmanaged by a discovery stitcher (the `PopulateDNCIM_ManagedStatus.stch` file in the `$ITNMHOME/disco/stitchers/DNCIM` directory)
 - 3 – Unmanaged by discovery because entity is out of the discovery scope
- Not every entity in the containment hierarchy of a device must have an explicit entry in the **managedStatus** table
 - If you unmanage a chassis from the GUI, all subcomponents are considered unmanaged by default
 - If a chassis is managed, all subcomponents are considered managed unless they are explicitly unmanaged
- Tivoli Network Manager automatically propagates managed status 1 – 3 from the chassis to the subcomponents

Figure 8-49. NmosManagedStatus values

If a device is unmanaged from the GUI, the authorized user can manage the device again from the GUI. Discovery cannot change the NmosManagedStatus to 0 (managed) if a user unmanaged the device from the GUI.

If discovery flagged the device as unmanaged, the GUI cannot set a device to unmanaged if the discovery flagged the device as unmanaged.

A subsequent discovery can set the device to managed if it was unmanaged by a previous discovery. For example, the last discovery could set an interface entity to have a **NmosManagedStatus=3** because it was out of scope. However, if the discovery scope now includes that interface, the NmosManagedStatus can be set to 0 (managed) by the new discovery.

The event gateway populates the **NmosManagedStatus** field of the **alerts.status** table on the ObjectServer. The RCA plug-in on the gateway has a default behavior to calculate RCA only for those events with the NmosManagedStatus of 0.

To prevent polling of certain types of devices, you can modify the `PopulateDNCIM_ManagedStatus.stch` file to cause devices to be flagged as unmanaged (NmosManagedStatus=2).

Manage and unmanage tools

- Run in context
- When a device or interface is unmanaged:
 - It receives no ping polls
 - It receives no SNMP polls
 - No root cause analysis (RCA) is done on it unless a gateway property is set to force RCA
- Managing or unmanaging devices from the GUI changes the **ncim.managedStatus** table
- The poller checks the **ManagedStatusUpdateInterval** field of the **config.properties** section of the **\$NCHOME/etc/precision/NcPollerSchema.cfg** file every 30 seconds and then resets internal polling scopes

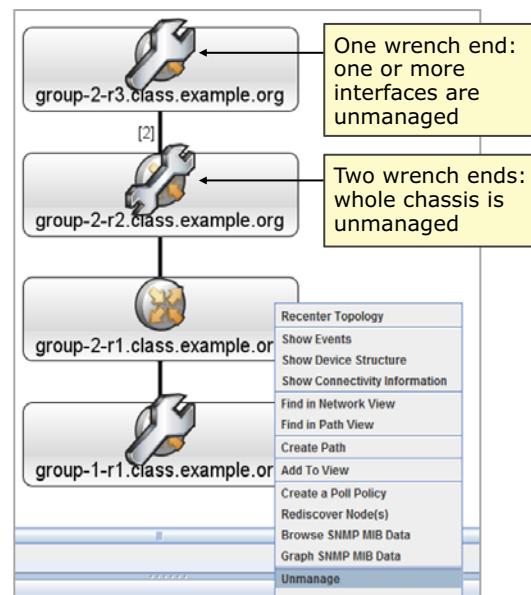


Figure 8-50. Manage and unmanage tools

When you unmanage a device or interface, no polls are sent to it.

- No ping polling is performed against the entity.
- SNMP polling of a managed chassis does not raise events against an unmanaged interface.
- Specific poll policies (such as **isdnLinkUp**) can be created to raise events against an otherwise unmanaged interface. To do so, set the **monitorUnmanaged** flag for that poll.

Raising events on unmanaged devices

- Events can originate on unmanaged devices
 - Sources external to Tivoli Network Manager (such as OMNibus probes) detect these events
 - These event sources are unaware of the managed status of an entity
- Individual Network Manager poll policies can be configured to monitor unmanaged entities
 - An event that is raised against an unmanaged entity has the NmosManagedStatus flag that is set to a nonzero value by the gateway
 - Filters can suppress the display of events from these unmanaged entities
 - No root cause analysis (RCA) is calculated against unmanaged entities

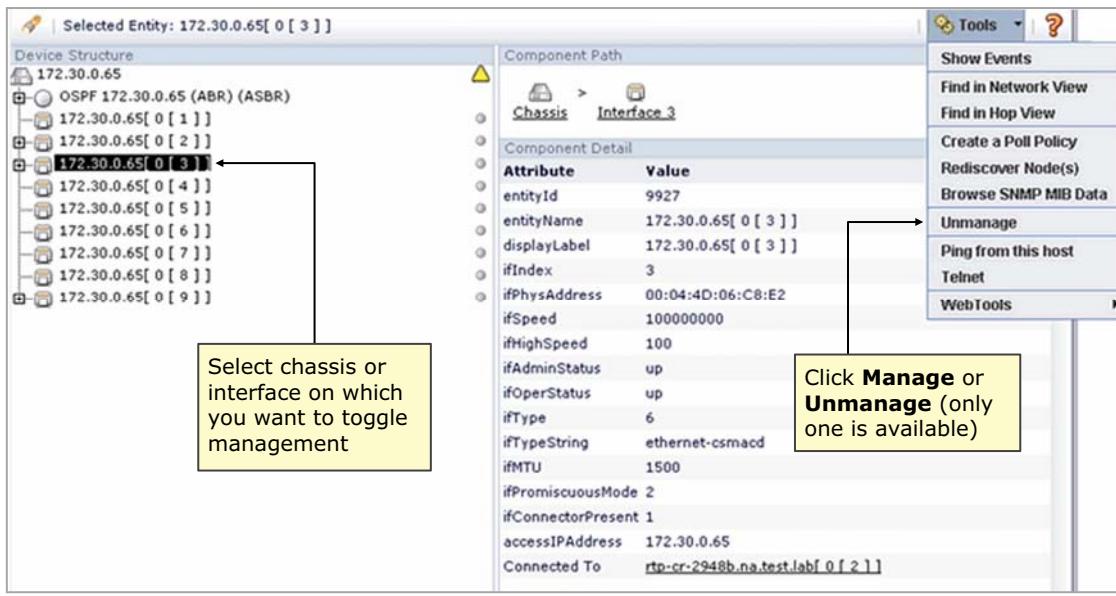
Figure 8-51. Raising events on unmanaged devices



Reminder

You still get events from devices even when Tivoli Network Manager is not polling those devices.

Marking a device as unmanaged from the Device Structure browser



Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-52. Marking a device as unmanaged from the Device Structure browser

Select the chassis or interface on which you want to toggle management.

From the Structure Browser tools menu, select **Manage** or **Unmanage** (only one is available).

When a chassis is marked as unmanaged, all entities that are contained within it, such as interfaces, are implicitly marked as unmanaged. Therefore, the manage and unmanage tools are disabled in these instances.

Identifying unmanaged interfaces

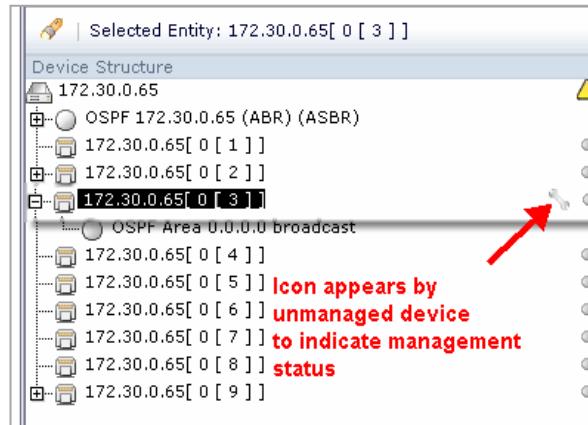


Figure 8-53. Identifying unmanaged interfaces

A wrench icon appears next to unmanaged chassis or interface entities.

Unmanaging devices in PopulateDNCIM_ManagedStatus.stch

- Final phase stitchers query **workingEntities.finalEntity** and DNCIM tables
- PopulateDNCIM_ManagedStatus.stch** updates the entity ManagedStatus in the **dncim.managedStatus** table

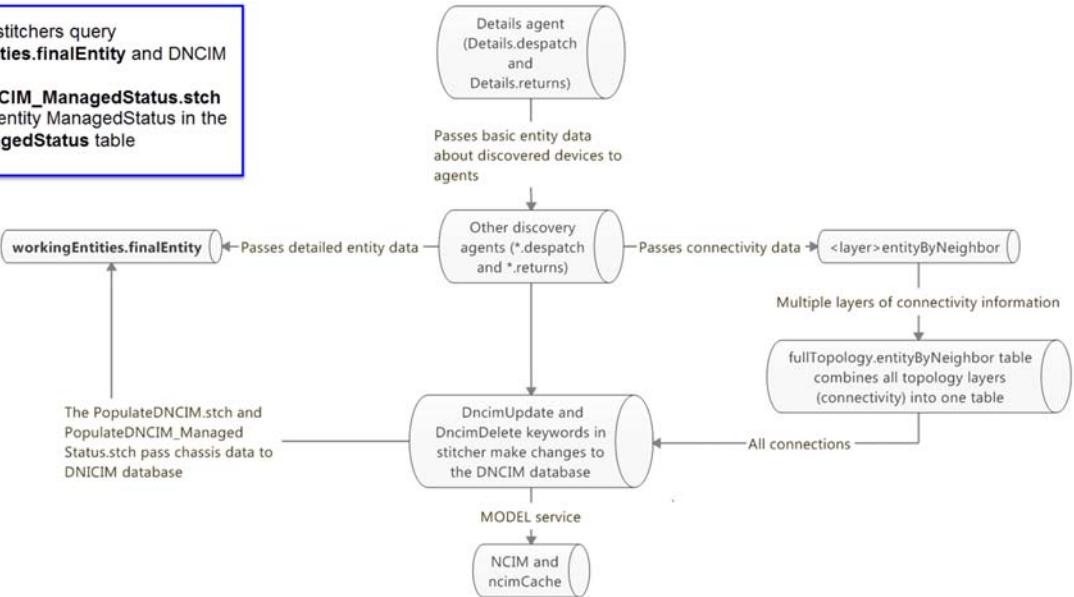


Figure 8-54. Unmanaging devices in PopulateDNCIM_ManagedStatus.stch

© Copyright IBM Corporation 2017

Topic summary

Having completed this topic, you should be able to do the following tasks:

- Manage or unmanage a device chassis
- Manage or unmanage an interface

Figure 8-55. Topic summary

8.5. Configure adaptive polling

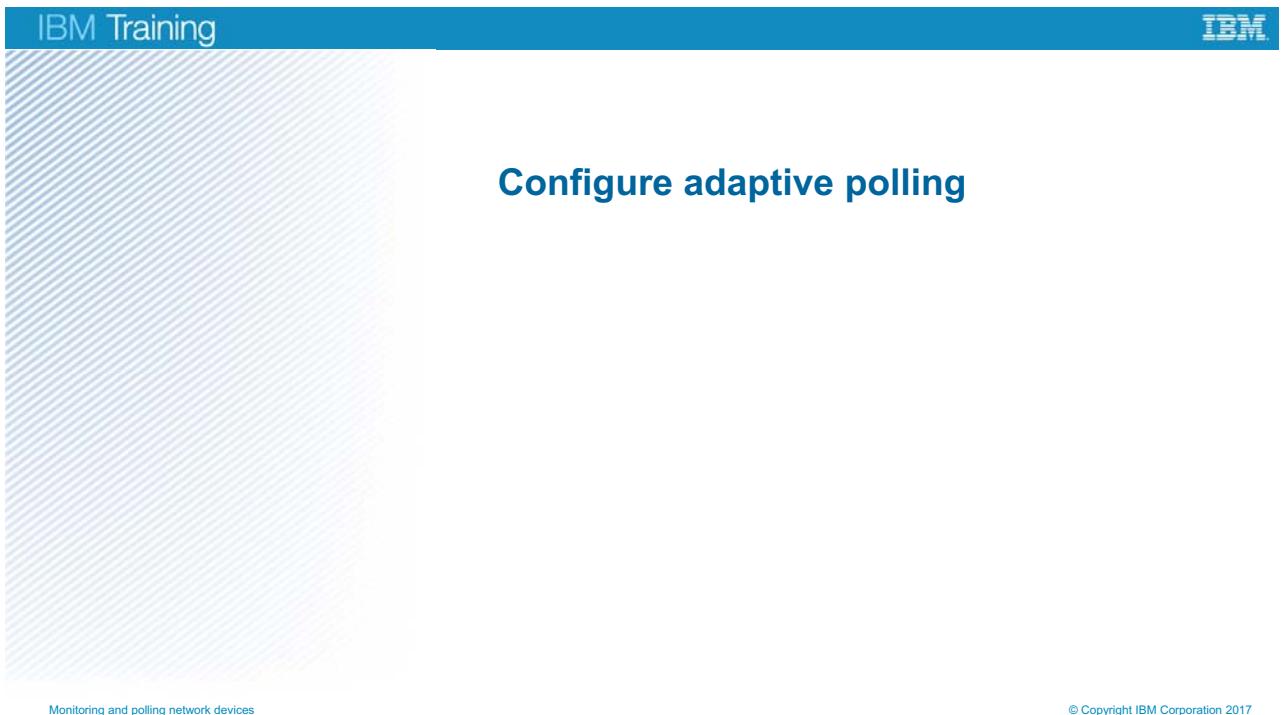


Figure 8-56. Configure adaptive polling

Adaptive polling overview

- You can create network views that are based on specific network and device conditions
- You can then create polling policies that apply to the created network view
- Multiple polling policies can attach to the same network view
- These policies can differ in the following attributes:
 - Polling interval
 - Polled data
 - Type of event that is generated
 - Whether polled data is stored

Figure 8-57. Adaptive polling overview

Tivoli Network Manager 4.2 includes the ability to create monitoring policies that react to dynamic events. The following scenarios are some examples of using adaptive polling policies.

- Scenario 1: Determine as quickly as possible when a device is down. If an interface fails to respond, Tivoli Network Manager generates a Ping Fail event. Tivoli Network Manager then accelerates pinging to clear the event as soon as the device responds. If the device is still not responding after 3 minutes (configurable), confirm to the operator that the device is down.
- Scenario 2: Accelerate polling when a threshold is violated. A standard policy for discard rates on a set of devices is set at a default of 30-minute intervals (configurable). If it detects that an interface exceeds the threshold, the poll generates a **DiscardRate** event. When this condition occurs, accelerate polling to provide more timely information before you react to the situation.

The purpose of these scenarios is to allow relatively slow polling and discard the false positives as quickly as possible. The normal polling uses few system resources, but resource usage increases during the triggered accelerate polling.

Adaptive polling examples

1. Determine as quickly as possible when a device is down
 - Network Manager detects an interface down
 - Accelerate pinging
 - If the device does not respond after 3 minutes, assume that it is down
2. Accelerate polling when a threshold violation occurs
 - A standard policy for discard rates is set at a default of 30-minute intervals
 - If the poller detects that an interface exceeds the threshold, it generates a **DiscardRate** event
 - Accelerate polling to shorter intervals to provide more timely information before you respond to the situation

Figure 8-58. Adaptive polling examples

Adaptive polling example 1

Confirm Device Down:

- Does rapid pinging of devices upon an initial ping fail to elevate critical issues faster
- Create a Network View that is called **Initial Ping Fail Event** based on a filter of **NmosPingFail and Tally < 18**

Policies

- Use as a first policy the **Default Chassis** or **Interface Ping** policy
- Create a second policy called **Confirm Device Down**
 - Limit the policy scope to entities in the **Initial Ping Fail Event** network view
- When the first ping policy detects a ping fail, an **NmosPingFail** event is raised
 - This event causes the device to become a member of **Initial Ping Fail Event** network view
- The **Confirm Device Down** policy sees a device in the **Initial Ping Fail Event** network view and starts to ping the device at a more rapid rate
 - When the tally exceeds 18 or the device responds, the device exits the view and the **Confirm Device Down** policy stops pinging

Figure 8-59. Adaptive polling example 1

Adaptive polling example 2

Confirm High Discard Rate

- Collects and stores discard rate at a shorter interval for those devices that have at least one interface event for **HighDiscardRate**
- Create a Network View that is called **HighDiscardRate** that includes all devices that have at least one interface event for **HighDiscardRate**

Policies

- Use as a first policy the **High Discard Rate** policy with a polling interval of 1800 seconds
- Create a second policy called **Confirm High Discard Rate** with an interval of 300 seconds
 - Store the poll data
 - Assign this policy to work on the Network View called **HighDiscardRate**
 - The first policy detects a threshold violation and issues a **HighDiscardRate** event and the node becomes a member of the High Discard Rate network view
 - The **Confirm High Discard Rate** policy polls the device at the shorter interval
 - When the discard rate falls below the threshold, the device is removed from the view, and polling returns to the longer interval
 - Stored poll data gives you more data points with which to diagnose a problem

Figure 8-60. Adaptive polling example 2

Topic summary

Having completed this topic, you should be able to do the following task:

- Create an adaptive polling policy

Figure 8-61. Topic summary

8.6. Real-time MIB graphing

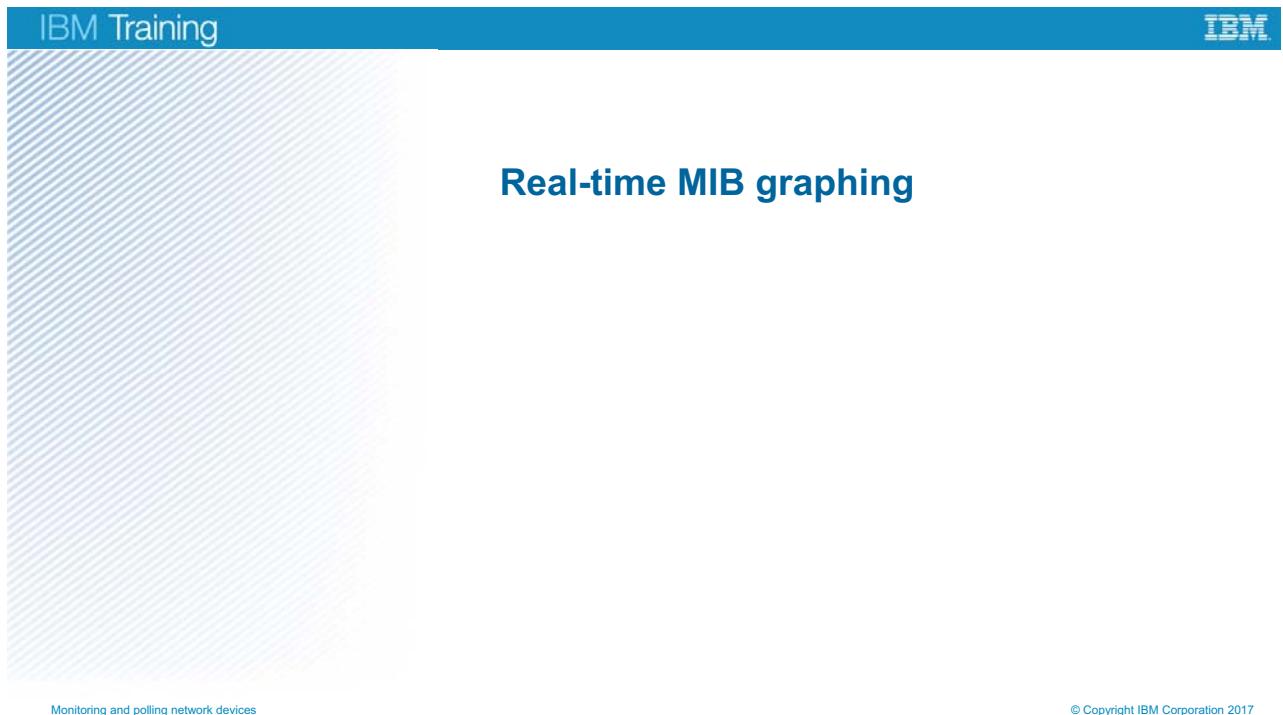


Figure 8-62. Real-time MIB graphing

Starting the SNMP MIB Grapher

You can start the SNMP MIB Grapher from:

1. The DASH menu
2. Right-click a device icon or entity record (tabular view) and select Polling and MIB Info / Graph SNMP MIB Data
3. Right-click event and select Open SNMP MIB Grapher

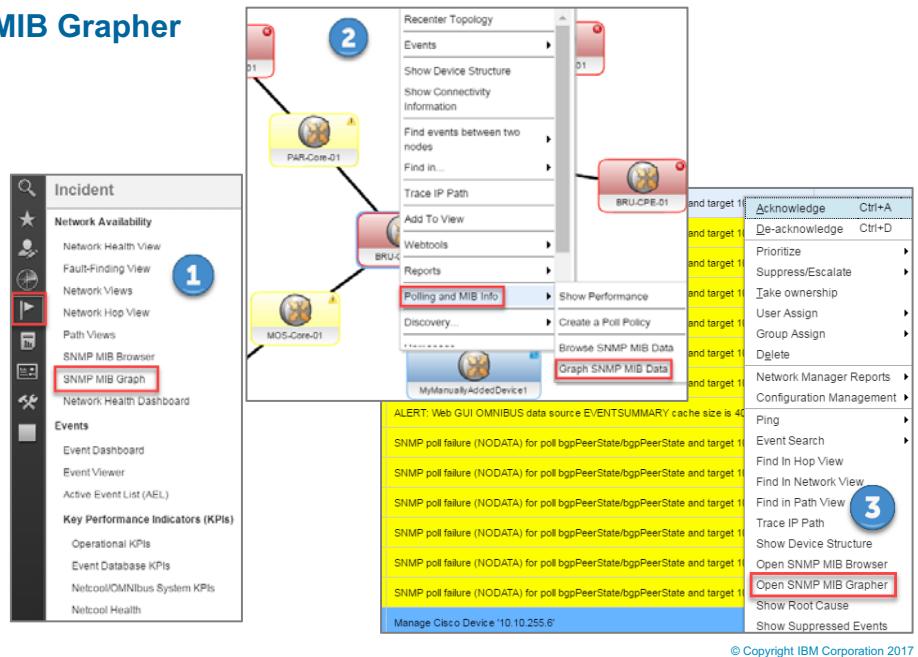


Figure 8-63. Starting the SNMP MIB Grapher

IBM Training



Specify polling parameters

- Specify SNMP varbind names to poll
- Set polling interval
- Set more preferences
- Click **Save**
- Graph can take several moments to start because polls must be scheduled while regular polling continues

Please specify MIB graph properties

*Domain: NOI_AGG_P
*Hostname: 10.10.255.1

* Poll data:

Name	OID/Poll Def
sysUpTime	PollDef
memoryPoll	PollDef

*Polling interval (seconds): 20

Additional Preferences

Title: 10.10.255.1 - sysUpTime:mem
Graph refresh interval (seconds): 30

Default selected rows: Highest Column: Current

Override SNMP Community String (SNMP v1 and v2 only)
Community String: []

Save Cancel

10.10.255.1 - sysUpTime:memoryPoll

No data retrieved from poller at: 18:39:43

Graph will take a few moments to show data

© Copyright IBM Corporation 2017

Figure 8-64. Specify polling parameters

IBM Training

SNMP MIB Grapher

- Real-time MIB graphing of 1 – 2 MIB variables or expressions
- Table format view for sorting and filtering
- Data point value tooltips

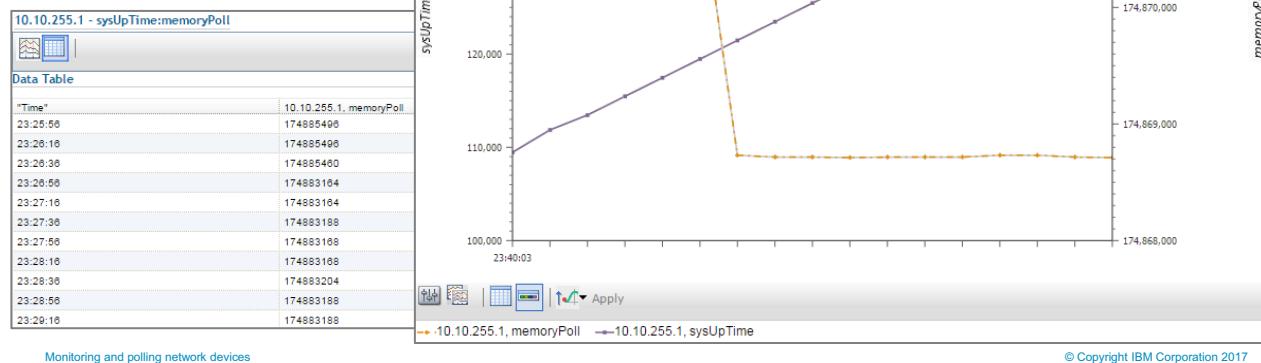
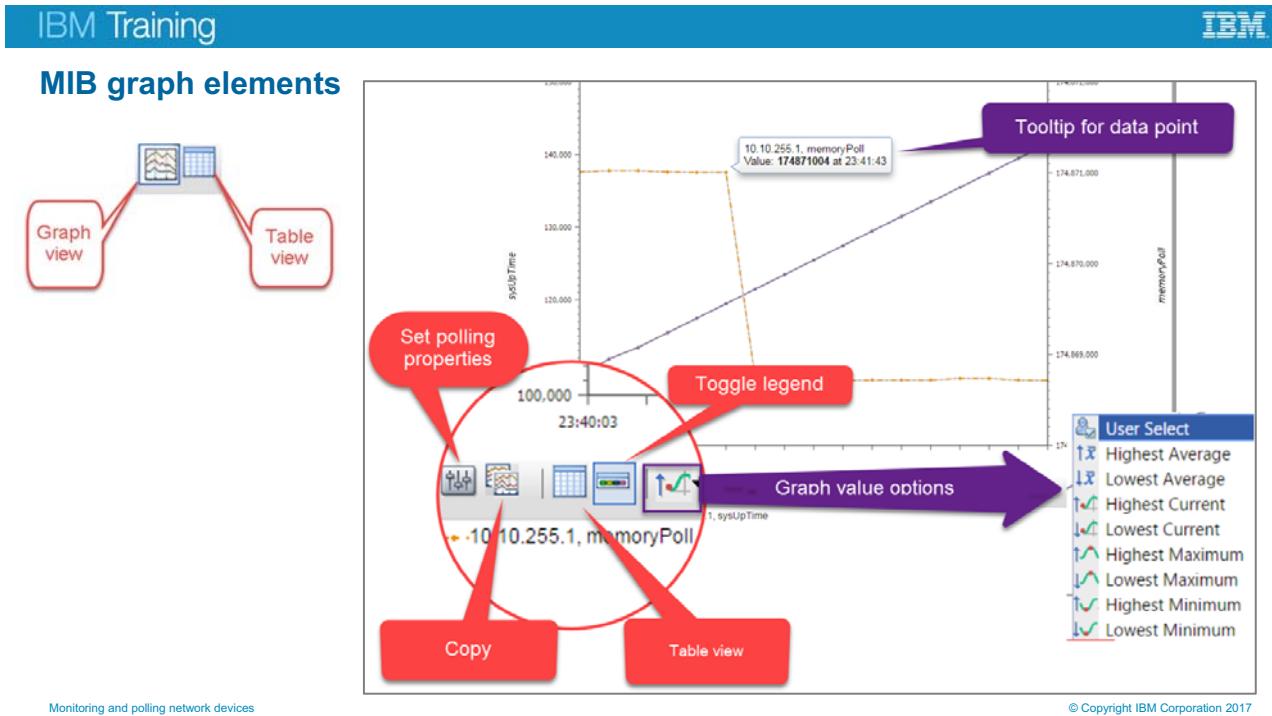


Figure 8-65. SNMP MIB Grapher

MIB graphs support large counter values. Graphs are presented with remembered or default settings. Historical graphs can be presented initially when recent data is available. Real-time graphs are always available.

In previous versions of Tivoli Network Manager, the graph also showed historical data. That feature was deprecated in version 4.2 because the new Network Health Dashboard, reports, and other tools can show historical data.



Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-66. MIB graph elements

The **Graph / Table** toggle button changes between graphical and tabular representations of the selected SNMP MIB variables.

This graph shows the relationship between **sysUpTime** and **memoryPoll**. You can compare two Y-axes with one X axis.



Troubleshooting

Log information for the real-time MIB Grapher can be found in the `/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm/ncp_mib.0.log` file.

Topic summary

Having completed this topic, you should be able to do the following tasks:

- Start the real-time MIB Grapher
- Load new MIB files into Tivoli Network Manager
- Configure multiple pollers on the server
- Specify one poller as the administrative poller

Figure 8-67. Topic summary

8.7. Troubleshooting information

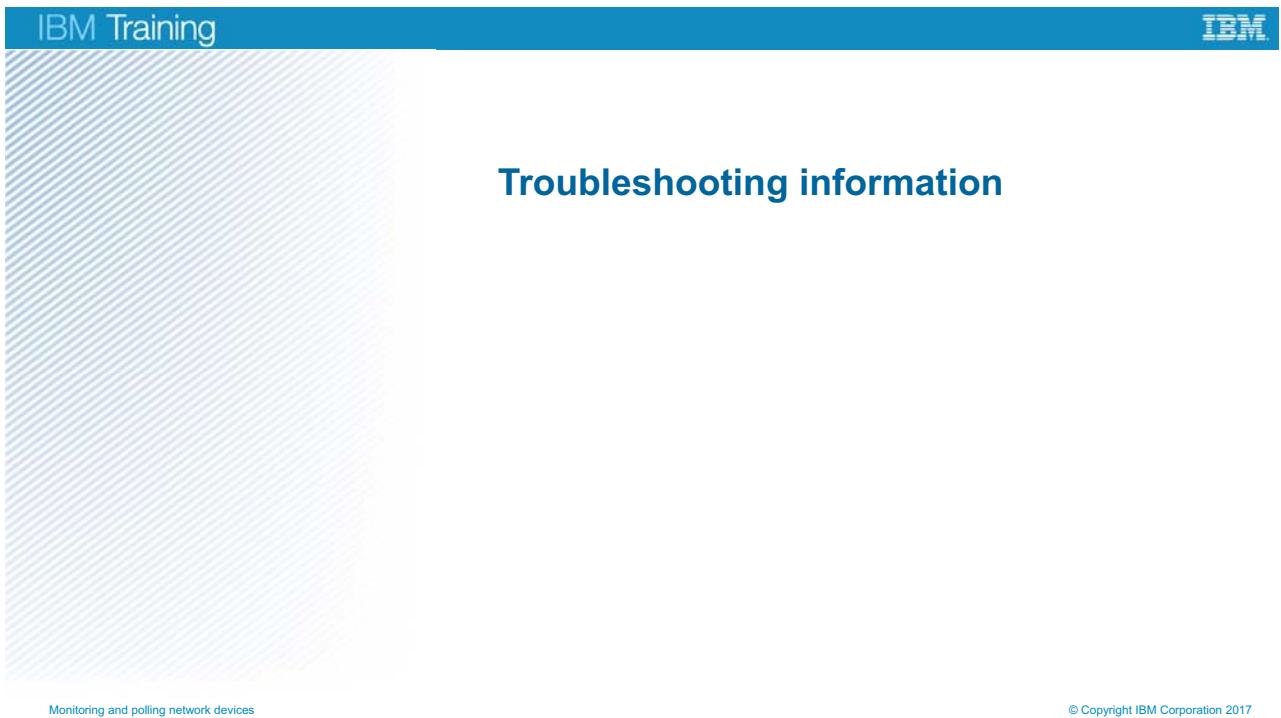


Figure 8-68. Troubleshooting information

Reasons to troubleshoot polling

- Device shows in membership list but is not being polled
- Poll definition filtering (class, interface)
- Throttle value that is reached
- No SNMP access

Figure 8-69. Reasons to troubleshoot polling

Troubleshooting ping polling (1 of 3)

Several factors might prevent the ping polling of IP addresses:

- The device that is specified in the poll policy was not discovered
- The device or interface is not included in any of the ping policy scopes
- Someone used a topology GUI to mark a device as unmanaged
- The interface was marked as unmanaged at discovery time or determined to be unroutable

Figure 8-70. Troubleshooting ping polling (1 of 3)

Troubleshooting ping polling (2 of 3)

1. Add the list of IP addresses whose polling you want to monitor by using the **ncp_upload_expected_ips.pl** script
 - Use the following command:
`$ITNMHOME/bin/ncp_perl
$ITNMHOME/scripts/perl/scripts/ncp_upload_expected_ips.pl -domain
DOMAIN_NAME -file FILENAME -password PASSWORD`
2. Periodically, or when needed for troubleshooting purposes, take a snapshot of the current ping polling status within the domain that is specified in the previous step
 - Use the following command:
`$ITNMHOME/bin/ncp_perl
$ITNMHOME/scripts/perl/scripts/ncp_ping_poller_snapshot.pl -domain
DOMAIN_NAME -password PASSWORD`
 - The results of the operation are stored in the pollLog database table within the NCMONITOR schema

Figure 8-71. Troubleshooting ping polling (2 of 3)

DOMAIN_NAME is the name of domain that contains the IP addresses. The listed IP addresses are compared only with IP addresses in this domain.

FILENAME is a plain text file of IPv4 addresses in standard dot notation with only one address on each line.

Troubleshooting ping polling (3 of 3)

3. Report on the entities that are not being polled

- Run the report of ICMP polling status of entities
- Use the following command:

```
$NCHOME/precision/bin/ncp_perl  
$NCHOME/precision/scripts/perl/scripts/ncp_polling_exceptions.pl  
-domain DOMAIN_NAME [-notpolled] [-format LIST | REPORT]
```
- The **-notpolled** 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
- The **-format LIST | REPORT** parameter specifies whether the output is 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

Figure 8-72. Troubleshooting ping polling (3 of 3)

Log files for troubleshooting SNMP polling

- Review log files for polling problems
 - `ncp_dh_snmp.domainName.log` (usual `stdout` output)
 - `ncp_dh_snmp.manager.domainName.log`
 - `ncp_dh_snmp.NcpMonitor.domainName.log`
 - `ncp_dh_snmp.SnmpPoller.domainName.log`
 - `ncp_dh_snmp.SnmpPoller.domainName.dbg` (enabled by an insert into the poller database)
- Log files for Managed/Unmanaged problems
 - If you manage or unmanage run from a topology view, check
`$NCHOME/log/precision/ncp_topoviz.0.log`
 - If you manage or unmanage from the Structure browser, check
`$NCHOME/log/precision/ncp_structurereview.0.log`

Figure 8-73. Log files for troubleshooting SNMP polling

Troubleshooting polling GUI

- If policies or definitions tables experience problems when you are doing a refresh check, the system gives an error message:
- This message indicates one of the following conditions:
 - Some problem in communication between the client and server
 - Configured refresh period is too short
 - Database is experiencing some difficulties
- Check that the database is running; check relevant log files



Figure 8-74. Troubleshooting polling GUI

Check relevant logs for errors:

- Web server logs
- Monitoring logs
- Poller database logs

Troubleshooting information

- Logs in **\$NCHOME/log/precision**
 - ncp_poller.ncp_poller_admin.NOI_AGG_P.log
 - ncp_poller.ncp_poller_admin.NOI_AGG_P.trace
 - ncp_poller.ncp_poller_default.NOI_AGG_P.log
 - ncp_poller.ncp_poller_default.NOI_AGG_P.trace
 - ncp_poller.SnmpPoller.ncp_poller_admin.NOI_AGG_P.metrics
 - ncp_poller.SnmpPoller.ncp_poller_admin.NOI_AGG_P.trace
 - ncp_poller.SnmpPoller.ncp_poller_default.NOI_AGG_P.metrics
 - ncp_poller.SnmpPoller.ncp_poller_default.NOI_AGG_P.trace
- Logs in **/opt/IBM/netcool/gui/precision_gui/profile/logs/tnm**
 - ncp_topoviz.0.log
 - ncp_topoviz.0.trace
- Check monitoring reports

Figure 8-75. Troubleshooting information

Troubleshooting monitoring status

- Tivoli Network Manager 4.2 includes a set of Perl scripts that help diagnose ping polling problems:
 - **ncp_upload_expected_ips.pl**
Loads in database the list of IP addresses whose polling you want to monitor
 - **ncp_ping_poller_snapshot.pl**
Takes a snapshot of the current ping polling status and stores in the **pollLog** table
 - **ncp_polling_exceptions.pl**
Based on expected IP addresses, the output gives reasons why the IP addresses are not being polled
- Scripts that are found in the **\$ITNMHOME/scripts/perl/scripts** directory

Figure 8-76. Troubleshooting monitoring status

Polling exception report (1 of 2)

```
[root@mdflap scripts]# ncp_perl ncp_polling_exceptions.pl -domain MUDFLAP
IBM Tivoli Network Manager Monitoring Status
=====
SUMMARY
=====
+-----+
| Snapshot Time | Undiscovered IPs | Unmonitored IPs | Unmanaged IPs | Unpolled last 15 Mins | Policies Delayed |
+-----+
| 2011-01-29 14:09:02 | 1 | 1 | 0 | 1 | 0 |
+-----+
UNDISCOVERED
=====
List of IP addresses from the reference list that are not in the management database.
+-----+
| IP Address |
+-----+
| 9.42.16.1 |
+-----+
OUT OF SCOPE
=====
+-----+-----+
| IP Address | Hostname | AOC Class |
+-----+-----+
| 9.42.17.86 | podf105.tivlab.raleigh.ibm.com | NoSNMPAccess |
+-----+
UNMANAGED
=====
List of IP addresses from the reference list that are not being monitored because
they were unmanaged in the GUI (status = 1)
+-----+-----+-----+-----+
| IP Address | Hostname | AOC Class | Entity Status | Device Status |
+-----+-----+-----+-----+
| | | | | |
+-----+
List of IP addresses unmanaged from Discovery (status = 2)
Check ifDescr in TagManagedEntities.stch for the following interfaces:
+-----+-----+-----+
| IP Address | Hostname | ifDescr | Entity Status |
+-----+-----+-----+
| | | | |
+-----+
```

Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-77. Polling exception report (1 of 2)

Polling exception report (2 of 2)

```

SECONDARY or UNPINGABLE
=====
List of IP addresses not polled as they are considered secondary addresses
+-----+-----+-----+
| IP Address | Hostname | ifIndex | Primary IP |
+-----+-----+-----+
|          |          |          |          |
+-----+-----+-----+
NOT POLLED IN LAST 15 MINS
=====
List of IP addresses that have not been polled during the last 15 minutes:
+-----+-----+
| IP Address | Hostname | AOC Class |
+-----+-----+
|          |          |          |
+-----+-----+
FALLING BEHIND
=====
List of IP addresses in policies that are falling behind by more than twice the polling interval
+-----+-----+-----+-----+-----+
| IP Address | Hostname | Policy | Poll Interval | Last Poll Int | Time since Last Poll |
+-----+-----+-----+-----+-----+
|          |          |          |          |          |          |
+-----+-----+-----+-----+
NO SNMP ADDRESS
=====
These devices may have other IP addresses that were not be discovered, but only the management address shown here will be polled (unless unmanaged above).
+-----+-----+
| IP Address | Hostname | Node Managed |
+-----+-----+
| 9.42.17.86 | pdf105.tivlab.raleigh.ibm.com |          |
+-----+-----+

```

Figure 8-78. Polling exception report (2 of 2)

Troubleshooting ping polling with `itnm_poller.pl` (1 of 2)

- Script for enabling, disabling, or checking status of policies
- Shows the monitoring status of an entity
- Syntax options for `itnm_poller.pl`

```
-domain <domainName>
[-enable <Policy Id>]
[-disable <Policy Id>]
[-status <all | static | real-time>] (defaults to static)
[-refresh <Policy Id or all>]
[-chassis <IP Address>]
[-interface <IP Address>]
[-help]
```

- The Policy ID can be obtained from the output of the status command

Figure 8-79. Troubleshooting ping polling with `itnm_poller.pl` (1 of 2)

Troubleshooting ping polling with itnm_poller.pl (2 of 2)

[root@bmudslap scripts]# ncp_perl itnm_poller.pl -domain NUDFLAP -status			
Policy Id	Policy Name	Poll Status	Monitor Status
1	Default Chassis Ping	enabled	no error
2	Default Interface Ping	disabled	
3	End Node Ping	disabled	
4	SNMP Link State	disabled	
5	Cisco Remote Ping	disabled	
6	Juniper Remote Ping	disabled	
7	frCircuitState	disabled	
8	isdnLinkUp	disabled	
9	bgpPeerState	disabled	
10	rebootDetection	disabled	
11	snChanFanOperStatus	disabled	
12	ciscoEnvMonTemperatureState	disabled	
13	snChanPowerSupplyOperStatus	disabled	
14	ciscoEnvMonFanState	disabled	
15	ciscoEnvMonSupplyState	disabled	
16	snChasActualTemperature	disabled	
17	ciscoMemoryPool	disabled	
18	cpnInPktErrors	disabled	
19	do3StateAlignmentErrors	disabled	
20	sysTrafficPoll	disabled	
21	ifInDiscards	disabled	
22	ifInErrors	disabled	
23	ifOutErrors	disabled	
24	loTfInCrcErrors	disabled	
25	bufferPoll	disabled	
26	memoryPoll	disabled	
27	ifOutDiscards	disabled	
28	fxCircuitReceivedFECHs	disabled	
29	fxCircuitReceivedUECHs	disabled	
30	sumpInBandwidth	disabled	
31	sumpOutBandwidth	disabled	
32	ciscoMemoryPctUsage	disabled	
33	ConfirmDeviceDown	disabled	
34	ConfirmHighDiscardRate	disabled	
35	HighDiscardRate	disabled	
36	sysUpTime	enabled	no error
37	ciscoCPUTotal5min	disabled	
38	ababPolicy	enabled	poll def bad

Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-80. Troubleshooting ping polling with itnm_poller.pl (2 of 2)

Topic summary

Having completed this topic, you should be able to do the following tasks:

- View report output to troubleshoot polling problems
- Review polling log files

Figure 8-81. Topic summary

Exercise: Configuring polling

Monitoring and polling network devices

© Copyright IBM Corporation 2017

Figure 8-82. Exercise: Configuring polling

Complete the exercise for this unit.

Review questions

1. Can a polling policy use more than one polling definition?
2. What command-line option must you use to designate the poller that can control polling database record locking?
3. What is the preferred way of limiting the scope of a polling policy?
 - A. Device filtering
 - B. SQL filters on the **mainNodeDetails** table
 - C. Assign the policy to a network view

Figure 8-83. Review questions

Write your answers here:

Unit summary

- Configure normal ping and SNMP link state polling
- Create a custom poll
- Configure adaptive polls
- Create a real-time MIB graph
- Configure distributed polling

Figure 8-84. Unit summary

Review answers

1. A polling policy can use multiple poll definitions.
2. Specify the `-admin` command-line option when you start the first poller in the `CtrlServices.cfg` file to designate the poller that can control polling database record locking.
3. What is the preferred way of limiting the scope of a polling policy?
 - A. Device filtering
 - B. SQL filters on the `mainNodeDetails` table
 - C. Assign the policy to a network view

Figure 8-85. Review answers

Unit 9. Understanding DNCIM

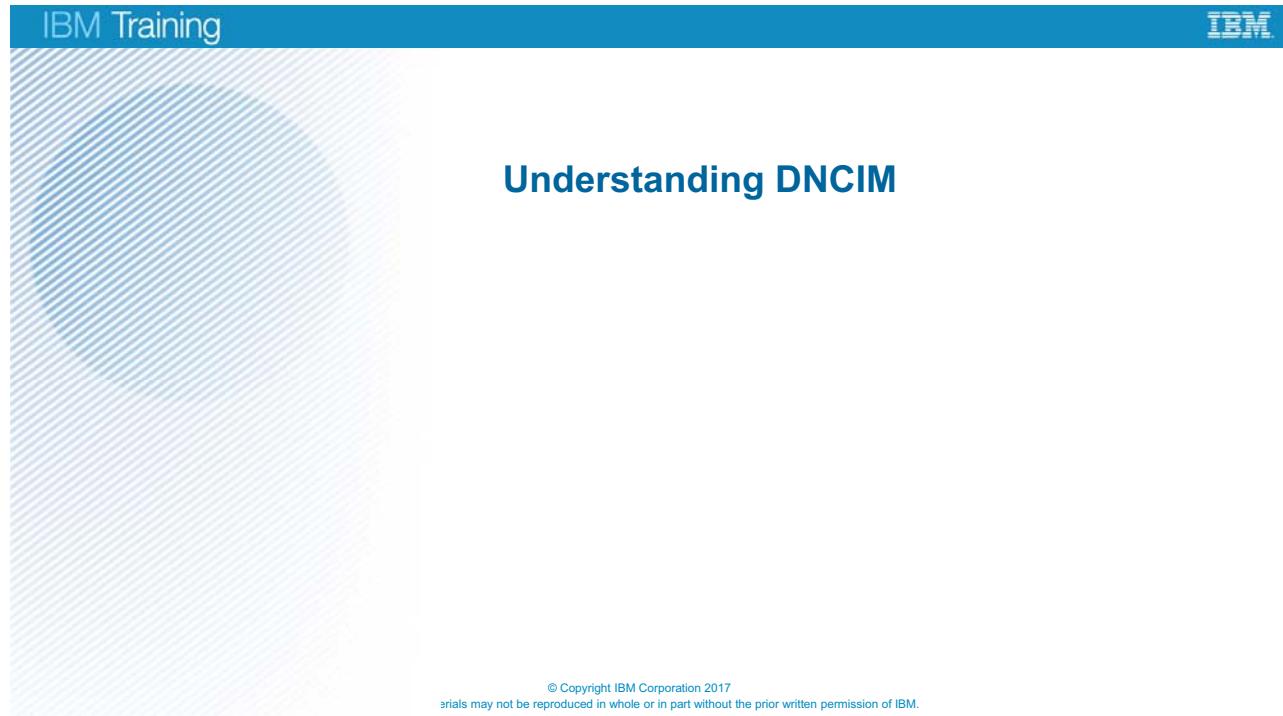


Figure 9-1. Understanding DNCIM

Estimated time

00:45

Overview

This unit describes the use of the DNCIM database during the discovery process.

How you will check your progress

- Run basic queries on the DNCIM database
- Add information to the DNCIM database with a custom stitcher

References

DNCIM documentation

<http://tinyurl.com/q6st49k>

Tivoli Network Manager 4.2 uses a Discovery Network Connectivity and Information Model (DNCIM) database. This database is an embedded SQLite database optimized for performance. It uses language constructs similar to DB2. This new database replaces the functionality of the **scratchTopology.entityByName** table in versions of Tivoli Network Manager before version 4.1.1.

Unit objectives

- Add information to the DNCIM database with a custom discovery stitcher
- Delete the DNCIM database to create a new one
- Query the DNCIM database
- Map data from DNCIM into the MODEL service

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-2. Unit objectives

9.1. DNCIM database basics

This lesson introduces the role of the DNCIM database. Understanding the basics of this database is helpful in understanding how to make customizations to discovery.

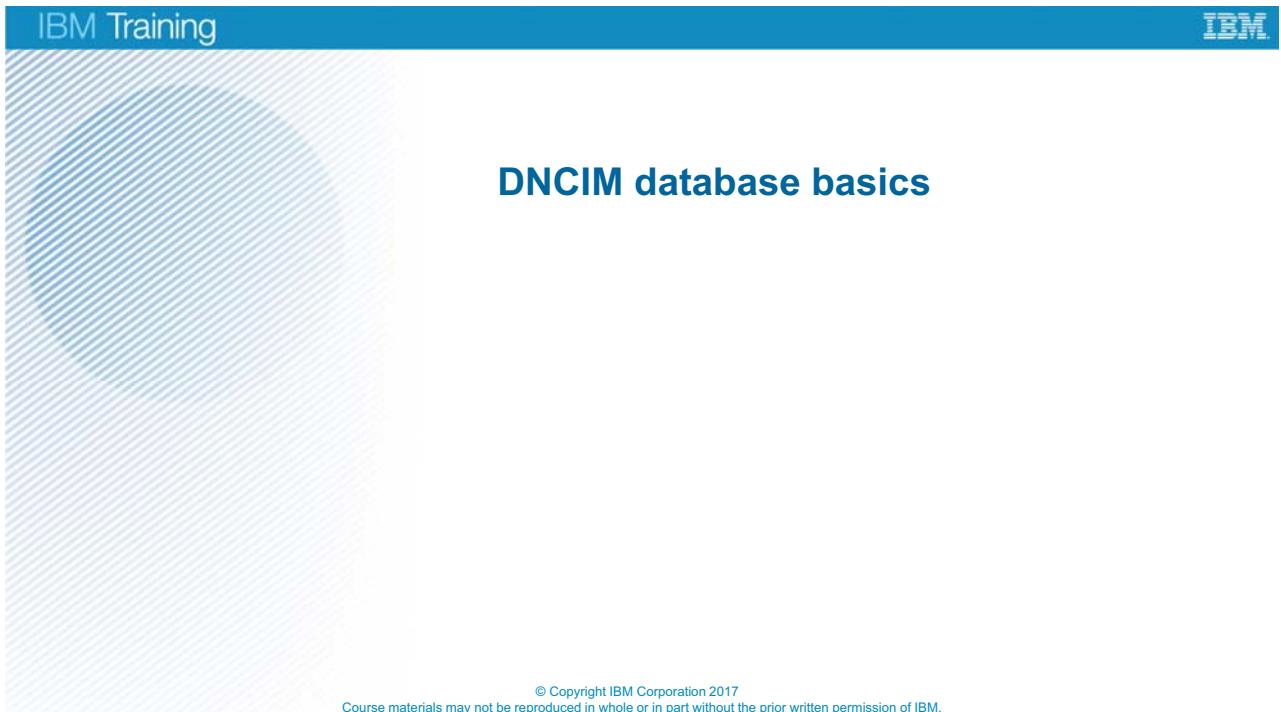


Figure 9-3. DNCIM database basics

DNCIM defined

The Discovery Network Connectivity and Inventory Model (DNCIM) database is a new relational database that replaces two tables that were used in versions of Tivoli Network Manager 3.9 and earlier

- **scratchTopology.entityByName** in the DISCO service
- **master.entityByName** in the MODEL service

Figure 9-4. DNCIM defined

Data flow

- The final phase of discovery runs DNCIM stitchers found in
\$ITNMHOME/disco/stitchers/DNCIM
- These stitchers populate the DNCIM database
- Custom stitchers can be called to work on the data in the DNCIM database to add or modify information
- When this process is complete, the completed topology is transmitted to the MODEL service (**ncp_model**) to populate the cached topology (**ncimCache**)
 - The gateway uses this database for root cause analysis
- The NCIM database is also populated
 - This database provides data for reporting and visualization of network connectivity

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-5. Data flow

Data structure and format

- Data structure is identical between DNCIM and NCIM databases
- DNCIM database is stored in an embedded sqlite database
 - This database is a lightweight, embedded database that is optimized for high-speed data processing
 - It has functions similar to DB2

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-6. Data structure and format

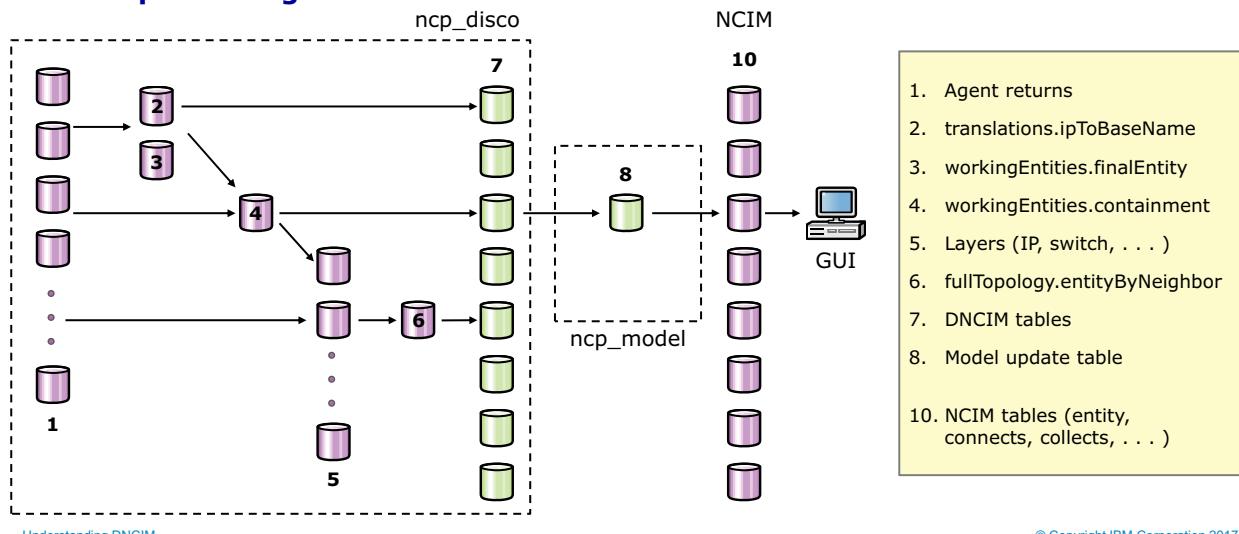
The SQLite database is an in-memory database that is optimized for extreme speed and extreme availability. The sqlite delivers extreme speed as a persistent, relational in-memory database that meets performance and reliability demands of real-time applications.

The `$ITNMHOME/embeddedDb/sqlite/processName.domainName` directory contains these files:

- `dncim.db`
- `main.db`

Version 4.2 discovery data processing

Current processing



Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-7. Version 4.2 discovery data processing

The early phases of discovery populate three tables:

- The **workingEntities.finalEntity** table contains a list of entities.
- The **workingEntities.containment** table specifies containment relationship. A **local neighbor** is an entity inside the same chassis.
- The **fullTopology.entityByNeighbor** table specifies the relationships between remote neighbors. A **remote neighbor** is an entity in another chassis.

The **PopulateDNCIM** stitcher takes the information in these three tables and writes it into the tables of the DNCIM database. The MODEL service processes mapping statements in the `ModelNcimDb.cfg` and `DbEntityDetails.cfg` files to add custom data to the NCIM database. The topology GUI is derived from the NCIM database.

Reasons to use DNCIM

- Discovery performance improvements over previous versions
 - DNCIM database has greater speed, record-locking, and advanced database features
 - Tends to require less memory
- Aids in customization and development efforts
 - DNCIM has a direct 1-to-1 mapping to the NCIM database that facilitates understanding of the data flow
 - Enhanced modularity of discovery information facilitates custom stitcher development
 - Facilitates future software development for increased performance
- Simplifies DISCO and MODEL services by eliminating proprietary flat database tables
 - Eliminates **scratchTopology.entityByName** (DISCO)
 - Eliminates **master.entityByName** (MODEL)
- Deprecates the unused post-discovery (or instantiation) filter

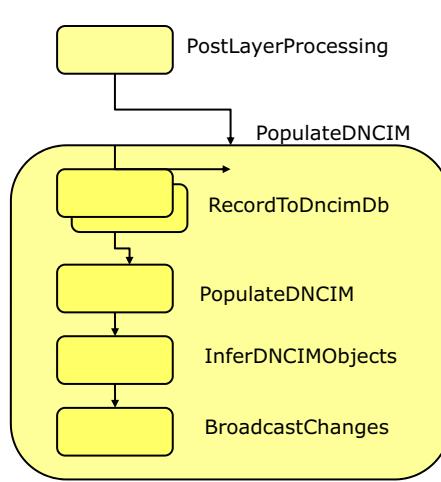
Understanding DNCIM

© Copyright IBM Corporation 2017

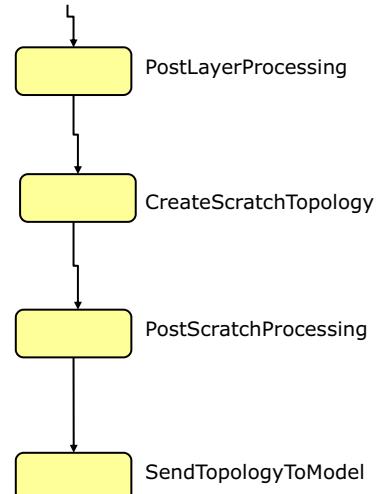
Figure 9-8. Reasons to use DNCIM

Stitching process changes

- Version 4.2



- Versions before 4.1



Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-9. Stitching process changes

The new **PopulateDNCIM** stitcher sends data from tables that contain entity, containments, and connectivity information to populate the DNCIM database. In future versions, discovery agents might write directly to the DNCIM database.

DNCIM configuration (1 of 2)

- The SQLite DNCIM database is started when the discovery process (`ncp_disco`) starts
- Entry in `DbLogins.domainName.cfg` controls DNCIM access
- Port usage
 - If the `DbLogins.domainName.cfg` file has a 0 for port number, the sqlite database uses the default port of 2315
 - If `DbLogins.domainName.cfg` specifies a nonzero value, the database uses the specified port number
- Key files
 - `$NCHOME/precision/embeddedDb/sqlite/appName.domainName`

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-10. DNCIM configuration (1 of 2)

The default ports for databases that are used by Tivoli Network Manager are specified in the `$NCHOME/etc/precision/DbLogins.cfg` file.

The value that is specified for the sqlite database is 0. This zero value means that the system allocates its own port number when you first start the system.

If you want to specify a different port, change the `DbLogins.domainName.cfg` file to specify that port.

DNCIM configuration (2 of 2)

- If no DNCIM database exists, one is created and populated with the DNCIM schema within a few seconds of starting discovery
- Two stitchers control the population and refresh of the database:
 - **PopulateDNCIM.stch** is called in processing phase
 - **RefreshDNCIM.stch** is called at start of next full discovery
 - This stitcher clears the DNCIM database
- On exit from discovery, a persistent copy of the DNCIM database file remains
 - This persistent database enables an administrator to restitch the last discovery at any time
 - For a clean discovery, delete the files in the
`$ITNMHOME/embeddedDb/sqlite/ncp_disco.domainName` directory

Figure 9-11. DNCIM configuration (2 of 2)

DNCIM key tables (1 of 2)

- EntityNameCache
 - EntityId, entityName, domainMgrId
- EntityData
 - EntityId, entityName, mainNodeEntityId, entityType, displayLabel
- EntityType
 - EntityType, typeName, dbTable, metaclass (element, collection, topology, and others)
- Contains
 - ContainingEntityId, containedEntityId
- Connects
 - ConnectionId, aEndEntityId, zEndEntityId

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-12. DNCIM key tables (1 of 2)

The DNCIM database contains the same tables and column names as the NCIM database.

DNCIM key tables (2 of 2)

- TopologyLinks
 - ConnectionId, entityId
- ProtocolEndPoint
 - ImplementingEntityId, endPointEntityId

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-13. DNCIM key tables (2 of 2)

Removing the DNCIM database tables

- To implement new discovery customizations, it can be necessary to delete information in the existing DNCIM database

- To delete the current DNCIM database, delete the directory that contains the information

```
cd $ITNMHOME/embeddedDb/sqlite  
rm -rf ncp_disco.domainName
```

- The database is rebuilt when discovery starts again

- To test a custom stitcher, use the information in the DNCIM database and run the final-phase stitchers again

Figure 9-14. Removing the DNCIM database tables

How to start a new clean discovery without discovery cache files

1. Stop all Network Manager processes by using the **itnm_stop** script
2. Remove all files from the **\$NCHOME/var/precision** directory that belongs to the domain you want to remove (the domain name is part of the file name)
 - For example, a configuration file that belongs to the domain NCMS would be called **file_name.NCMS.cfg**
3. Delete the **\$ITNMHOME/embeddedDb/sqlite/ncp_disco.domainName** directory
4. Archive or remove existing log files in **\$NCHOME/log/precision** to start the next discovery with fresh log files
 - The following log files pertain to the discovery process:
 - **ncp_disco**
 - **ncp_df_***
 - **ncp_agent***
 - **ncp_disco_perl_agent***
5. Restart the Network Manager processes by using the **itnm_start** script
 - New log files are automatically created when the Network Manager processes are restarted by using the **itnm_start** script
6. Run a new network discovery

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-15. How to start a new clean discovery without discovery cache files

9.2. Populating DNCIM database

You can create custom stitchers that add fields to the DNCIM database. These fields are then added to the NCIM database after discovery is complete.

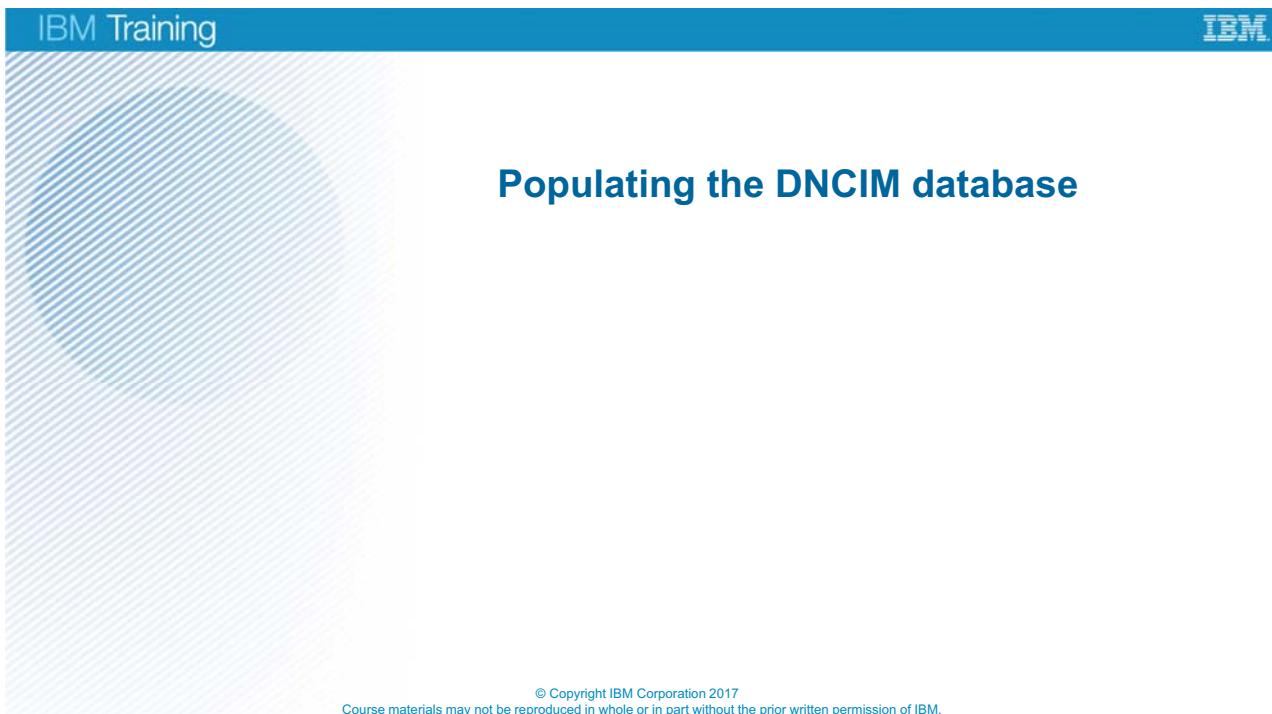


Figure 9-16. Populating the DNCIM database

PopulateDNCIM.stch (1 of 2)

`$ITNMHOME/disco/stitchers/DNCIM/PopulateDNCIM.stch`

- Builds the DNCIM database
- Cleans DNCIM
- Builds topology objects
- Builds objects (by using `RecordToDncimDb` according to `ModelNcimDb.cfg` mappings)
 - Chassis objects
 - Cards, module objects
 - Containers
 - Everything else (such as interfaces and protocol endpoints)
- Builds relationships
 - Containment
 - Connections
 - Collections
 - ProtocolEndPoint
 - HostedServices
 - Dependencies

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-17. *PopulateDNCIM.stch (1 of 2)*

PopulateDNCIM.stch (2 of 2)

- Uses the **RecordToDNCIMDb** action and information in **ModelNcimDb.cfg**
- Takes records from these locations and maps them for insertion into the DNCIM database
 - **workingEntities.finalEntity**
 - **workingEntities.containment**
 - **fullTopology.entityByNeighbor**

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-18. *PopulateDNCIM.stch (2 of 2)*

RecordToDncimDb([optional record])

This function takes a record, and passes it through the mappings in `ModelNcimDb.cfg` and `DbEntityDetails.cfg` to insert information in to the DNCIM database:

```
foreach(entityNames)
{
    entityName = eval(text,'&m_Name');
    RecordList entityRec = RetrieveOQL(
        "select * from workingEntities.finalEntity where
        m_Name = eval(text,'$entityName');");
    foreach(entityRec)
    {
        entityId = RecordToDncimDb();
    }
}
```

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-19. RecordToDncimDb([optional record])

To customize discovery, you assign values in mapping statements to build a temporary record. After the record is built, use the `RecordToDncimDb` keyword to write the record to the DNCIM database.

9.3. Mapping information into the MODEL service

This lesson shows how custom fields that are added to the DNCIM database are mapped into the NCIM database.

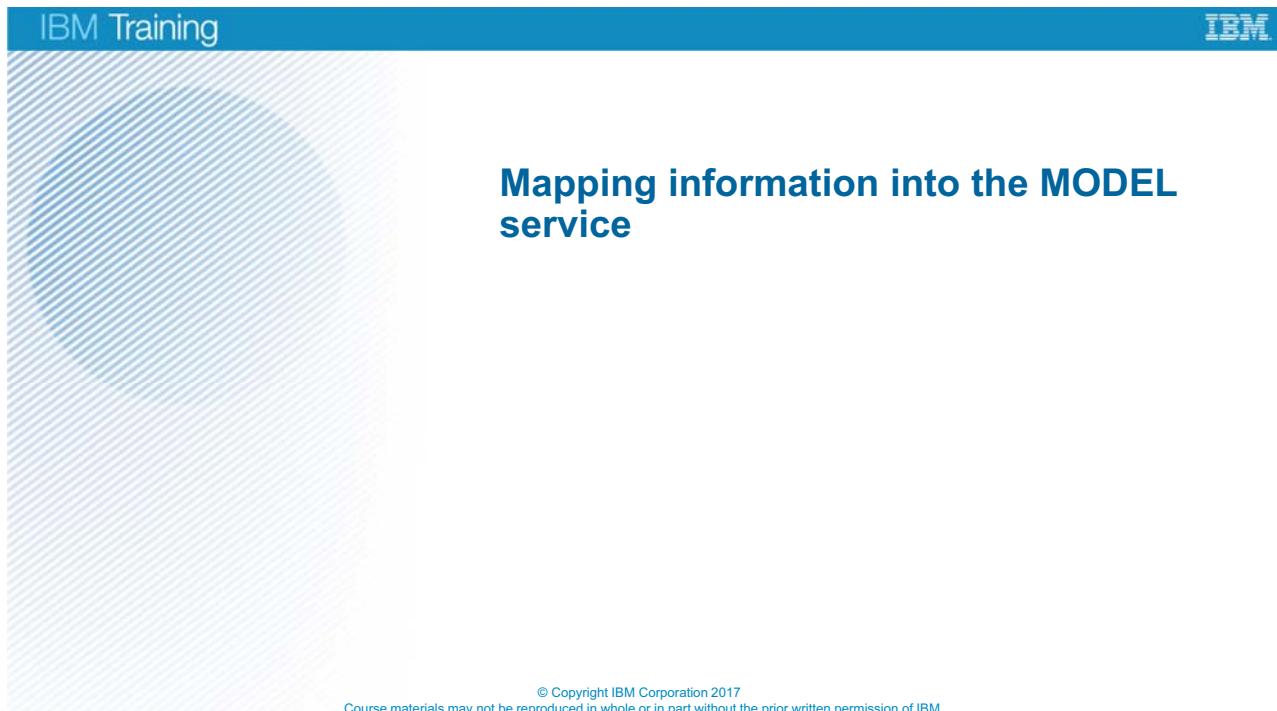


Figure 9-20. Mapping information into the MODEL service

Example mapping in ModelNcimDb.cfg

```
insert into dbModel.entityMap (
    EntityFilter,
    TableName,
    DisplayLabel,
    FieldMap)
values
("m_ExtraInfo->m_CPUData is not null",
 "cpu",
 "eval(text, 'FIRSTVALID(&m_ExtraInfo->m_DisplayLabel, &m_Name)'),"
 {
    entityId      = "eval(int, '&m_EntityId')",
    modelName     = "eval(int, '&m_m_ExtraInfo->m_Model')",
    coresInstalled = "eval(int, '&m_m_ExtraInfo->m_ProcUnits')",
    cpuSpeed      = "eval(int, '&m_m_ExtraInfo->m_Clock')"
}
);

```

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-21. Example mapping in ModelNcimDb.cfg

Tivoli Network Manager 4.2 product documentation contains more examples of how to create new entities or populate field information.

InferDNCIMObjects.stch

- Called by **PopulateDNCIM.stch**
- Creates entries in DNCIM for logical layer information such as these examples:
 - OSPF
 - BGP
 - MPLSTE
 - IGMP
 - IPMRoute
 - PIM
 - VTP
 - VlanCollections
 - MXGroupCollections
 - Network pipe connection speed

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-22. *InferDNCIMObjects.stch*

Avoid putting discovery customizations in the `InferDNCIMObjects.stch` file. Make customizations as early as possible, such as in agent files or in the early phases of stitching.

PopulateDNCIM.stch

The **PopulateDNCIM** stitcher does several important functions:

- It specifies discovery sources, such as SNMP and EMS systems
- The **CleanDNCIMObjects** clears temporary tables after processing discovery information
- It calls other changes:
 - Cross-domain stitchers
 - **PopulateDNCIM_ManagedStatus**
 - **BroadcastChanges** stitcher uses these keywords to alter data in the DNCIM database:
`DncimInsert`, `DncimUpdate`, and `DncimDelete`

Figure 9-23. *PopulateDNCIM.stch*

To customize the topology, call custom stitchers from the `PopulateDNCIM.stch` file.

Unit summary

- Add information to the DNCIM database with a custom discovery stitcher
- Delete the DNCIM database to create a new one
- Query the DNCIM database
- Map data from DNCIM into the MODEL service

Understanding DNCIM

© Copyright IBM Corporation 2017

Figure 9-24. Unit summary

Unit 10. Customizing discovery

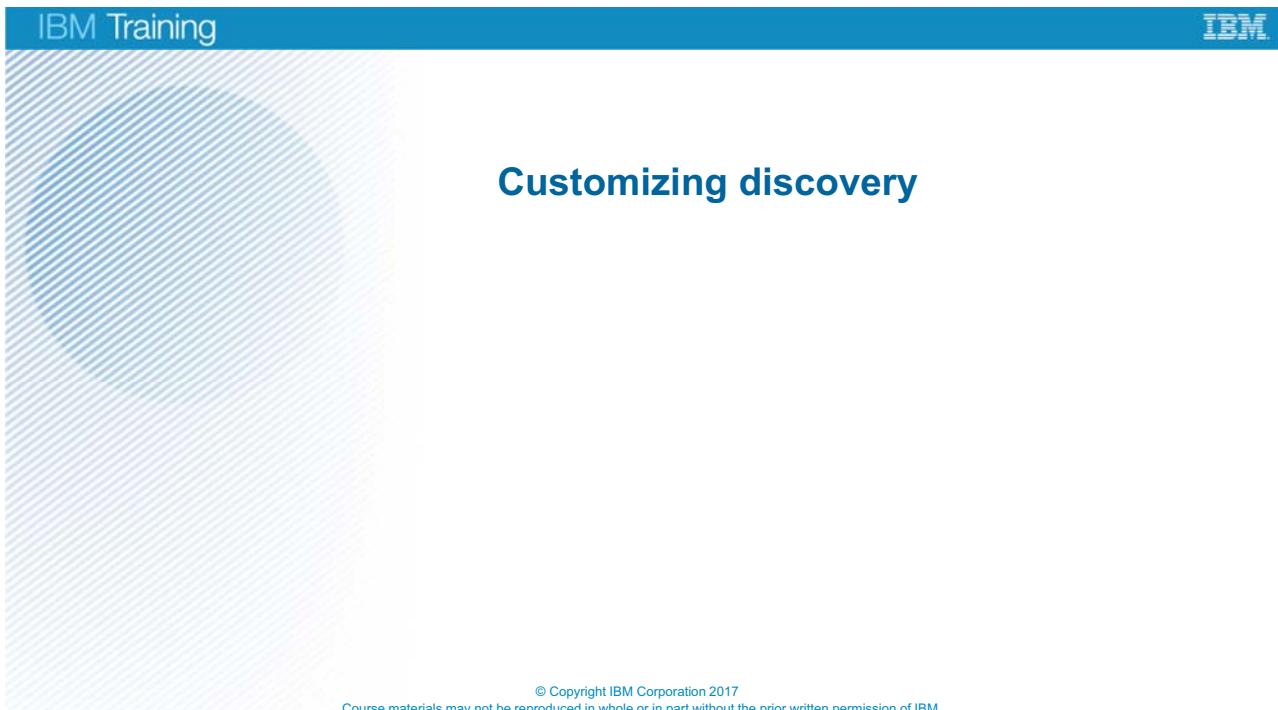


Figure 10-1. Customizing discovery

Estimated time

01:30

Overview

In this unit, students learn to add custom business information to the records for discovered devices.

How you will check your progress

- Complete student exercises
- Answer review questions

Unit objectives

- Explain the final phase of discovery
- Implement a custom stitcher that uses pattern matching on device host names to add business information to discovery
- Create custom tables
- Customize the structure browser view with custom data
- Create network partition views based on custom data
- Configure the MODEL service to populate the new table
- Create and delete custom database tables

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-2. Unit objectives

10.1. Discovery in-depth

Earlier in this course, you learned a high-level description of discovery. Finders find devices. Agents interrogate devices. Stitchers move data into a cohesive network map. This unit explains the final phase stitching process in greater detail. It also shows how to add information to discovery.

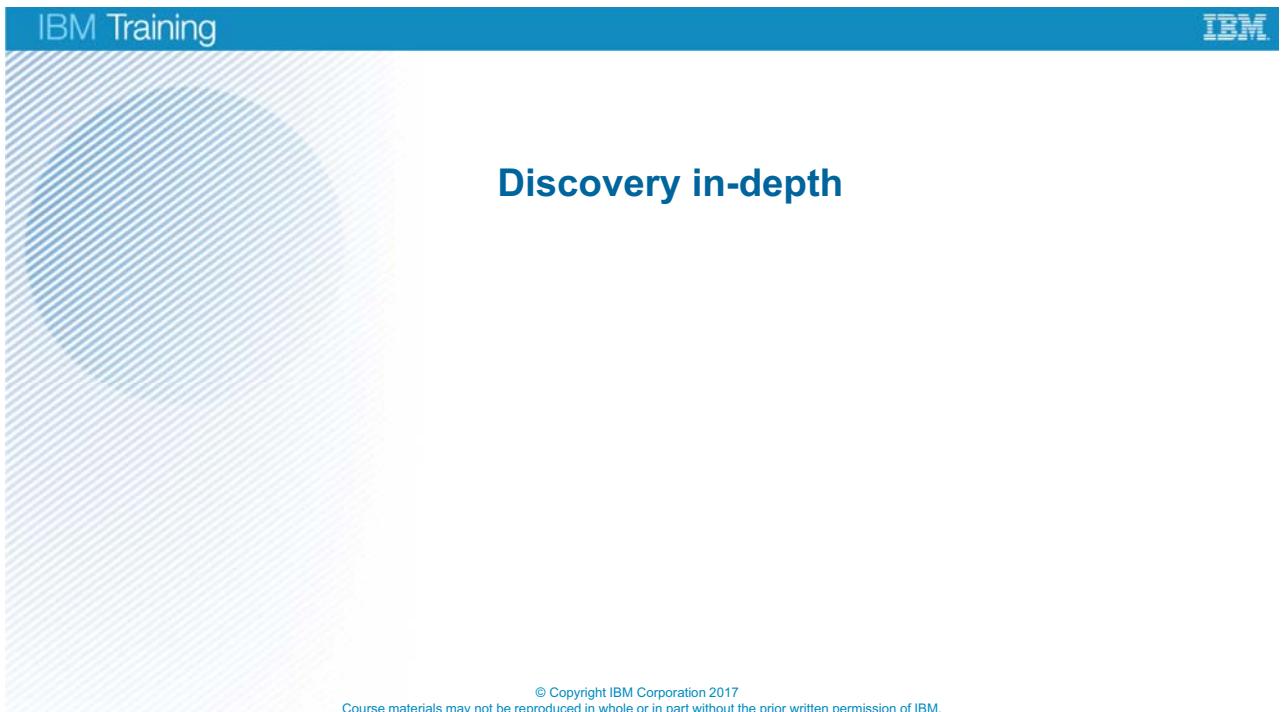
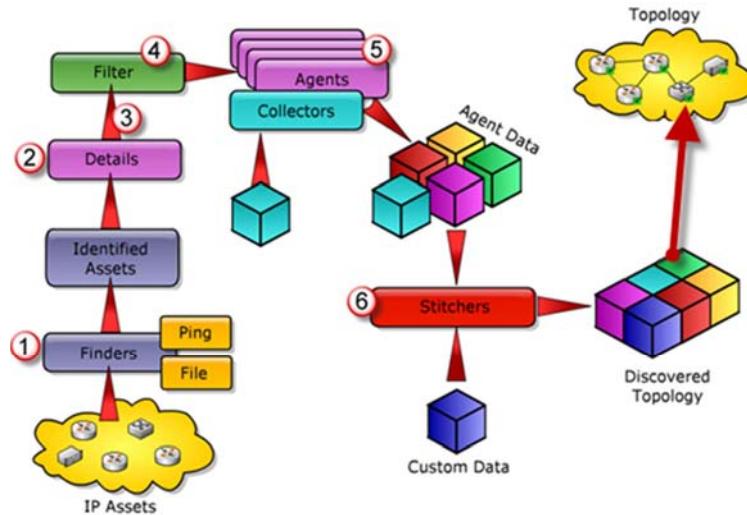


Figure 10-3. Discovery in-depth

Discovery overview



Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-4. Discovery overview

At a high level, discovery proceeds according to these steps:

1. Finders locate IP assets.
2. The **Details** agent identifies the system description and ObjectId (OID) number of a device.
3. The **AssocAddress** agent looks at a table to see whether discovery already found and interrogated the device.
4. If discovery did not already interrogate this device, filters that are found in other agents determine which agents interrogate of the device.
5. Agents interrogate devices and store data in tables.
6. Stitchers add custom data and build a list of entities and the connectivity between those entities.
7. A final topology is built.

Discovery details

After completing this unit, you should be able to do these tasks:

- Explain the details of the final phase of discovery
- Identify what causes a blackout state in discovery and explain what happens to devices that are found during the blackout state
- Identify how many tables each agent has

Enhancing discovery with business information

- Adding business information to topology information reduces mean-time-to-resolve (MTTR) network problems
- During the discovery process, information can be added from external sources:
 - Lookup table
 - Hosts file
- This information can be populated in a structure browser view or used to create network partition views based on business information such as:
 - Location
 - Customer
 - Region
 - Department
 - Country
 - Continent
 - Contact information

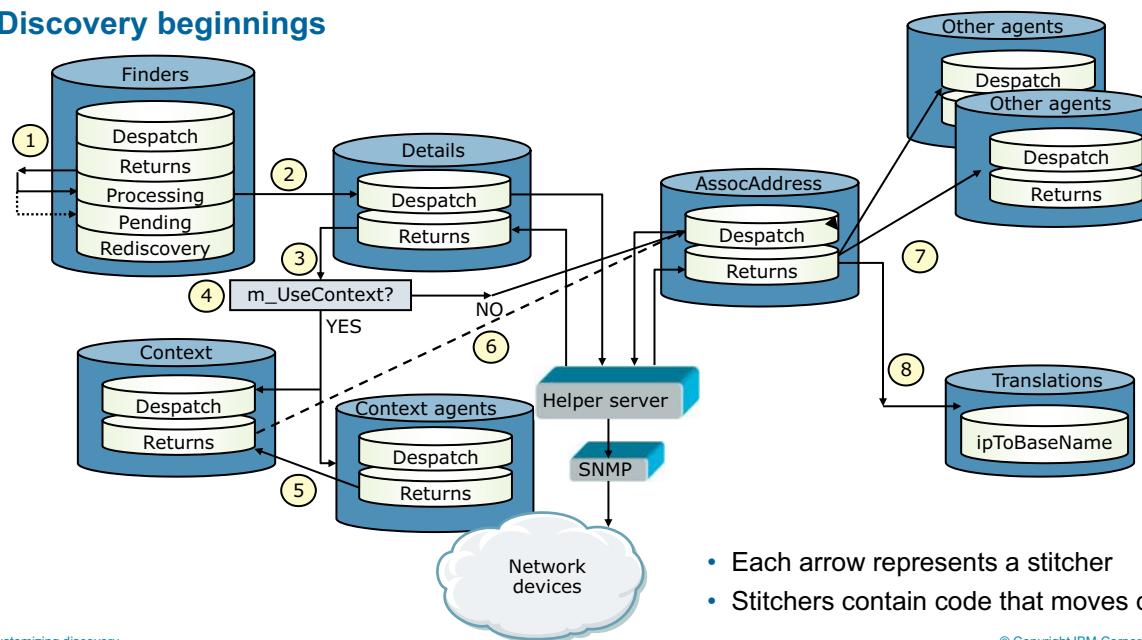
Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-6. Enhancing discovery with business information

By adding business-related information to topology objects, you can reduce mean time to repair (MTTR) a problem. Help desk personnel can more quickly understand the context and impact of the problem without wasting time in data lookup. This unit focuses on the methodology for adding business information during the discovery process. This information can then enrich events and give greater detail in topology views.

Discovery beginnings



Customizing discovery

© Copyright IBM Corporation 2017

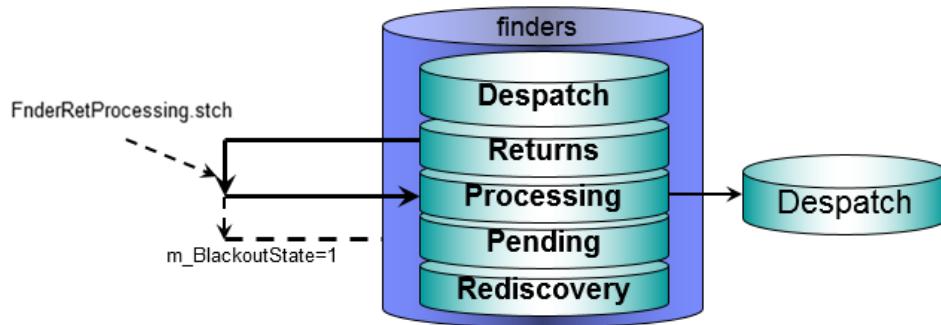
Figure 10-7. Discovery beginnings

1. Found devices are listed in **finders_returns**. If a blackout state exists, device information is stored in **finders_pending**, and no further discovery on them is done until the current stitching operations are completed. If a blackout state does not exist, then entries go into the **finders_processing**.
2. A stitcher takes data from **finders_processing** and puts it into **Details_despatch**. The Details agent then sends a request to the discovery helper server to query the devices for the ObjectID (OID) and system description of the device. Query results are stored in **Details_returns**.
3. A stitcher takes entries in **Details_returns** and evaluates them to see whether the device contains virtual routers inside it (such as a RedBack or Unisphere router).
4. If the router contains virtual devices inside it, then device information is forwarded to the appropriate context agents for device querying. If not, the device information populates the despatch table of the Associated Address (**AssocAddress**) agent.
5. Context agents complete processing.
6. Context agent results (if any) are sent to the despatch table of the **AssocAddress** agent.
7. The **AssocAddress** agent processes each IP address in its **despatch** table. It checks to see whether the IP address exists in its **Translations_ipToBaseName** table. This table contains a list of all IP addresses on each already-discovered device. If the address is not in the table, the new IP address is added and other agents begin interrogation of the device. If the address is in the table, no further processing occurs.

FnderRetProcessing.stch

This stitcher reads entries from **finders.returns**

- IP addresses populate the **finders.processing** table when the **disco.status** has **m_BlackoutState <> 1**
- If the **disco.status** has **m_BlackoutState = 1**, this stitcher populates the **finders.pending** table



Customizing discovery

© Copyright IBM Corporation 2017

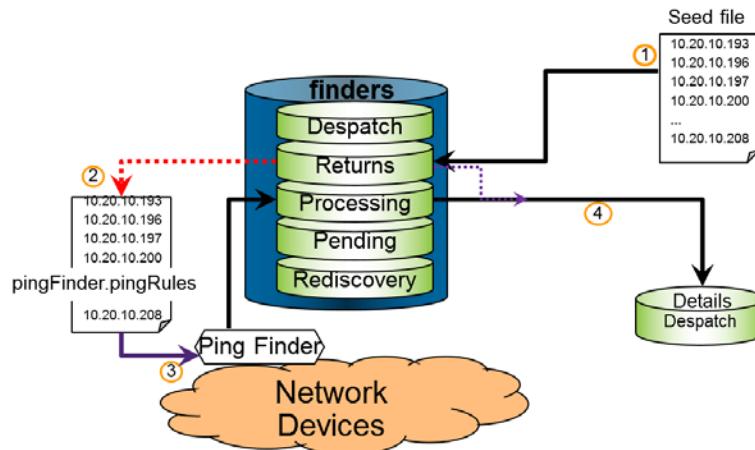
Figure 10-8. *FnderRetProcessing.stch*

When the discovery process fails to find any new devices for 90 seconds, the discovery process goes into a blackout state. During this state, the IP addresses of any devices that are found are placed into the **finders.pending** table. Agents complete interrogation, and final-phase stitching begins. After discovery is complete, any devices in the **finders.pending** table are processed in a partial discovery, and their information is added to the topology.

The timeout value is customizable.

Record data flow

- If file finder verification is enabled, a stitcher inserts the IP addresses in `finders.returns` into `pingFinder.pingRules`
- The ping finder verifies that the devices are reachable before it inserts IP addresses into the `finders.processing` table



Customizing discovery

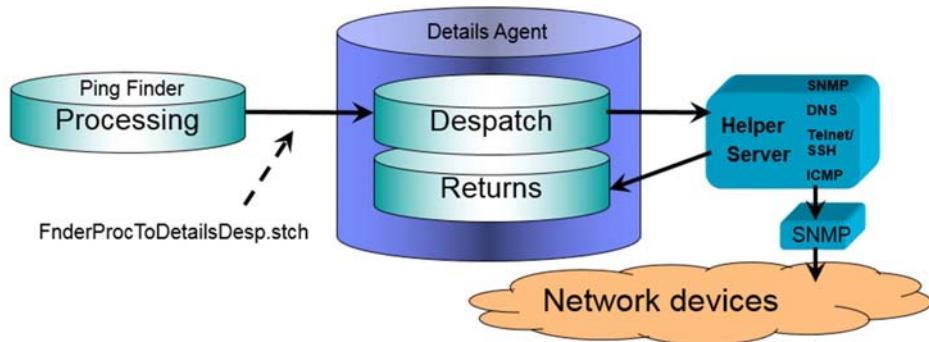
© Copyright IBM Corporation 2017

Figure 10-9. Record data flow

- Entity Record data flow begins with the file finder. It reads the seedlist file as defined within the `DiscoFileFinderSeeds.domainName.cfg` file. The IP address is inserted into the `finders.returns` table and flagged as **m_Creator=FileFinder**. The **FnderRetProcessing** stitcher reads this entry, and if discovery is in Phase 0, the record is inserted into `finders.processing`.
- When the **m_CheckFileFinderReturns** flag is set in the `DiscoSchema.domainName.cfg` file, the **FnderRetProcessing** stitcher causes the record to be inserted into the `pingFinder.pingRules`. It is then deleted from the `finders.returns` table so as not to be seen as a duplicate address.
- The ping finder processes this record. It tests the IP address to see whether it exists within the network. If the device is found to exist, it is inserted back into the `finders.returns` table and flagged with **m_Creator=PingFinder**.
- The **FnderRetProcessing** stitcher continues to check each record. If discovery is not in a blackout state, the record is sent directly to `finders.processing` and then to `Details.despatch`.

FnderProcToDetailsDesp.stch

- This stitcher populates the **Details.despatch** table
- Details agent starts and makes a request to the Helper Server for data
- Helper Server retrieves data and puts it into the **Details.returns** table



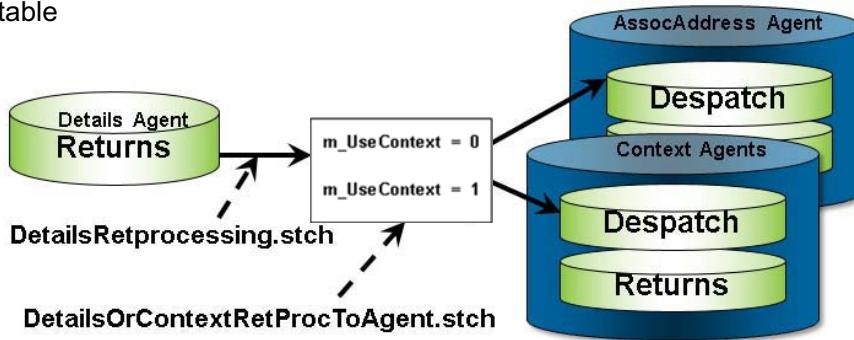
Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-10. FnderProcToDetailsDesp.stch

DetailsRetProcessing.stch

- **DetailsRetProcessing.stch**
 - Runs the **DetailsOrContextRetProcToAgent.stch**
 - Checks for existence in **AssocAddress.returns**
- Context agents look for virtual routers before they are inserted into the **AssocAddress** agent **despatch** table



Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-11. DetailsRetProcessing.stch

The **DetailsRetProcessing** stitcher checks to see whether the discovery service is set to run context agents.

Context agents are agents that gather information about containment relationships in a device when virtual devices are present. Unisphere or RedBack boxes might have virtual routers present inside a chassis, and gathering this information is necessary to build a correct topology.

Run a context-sensitive discovery when you have devices that you need to discover, such as the following types of devices:

- SMS devices
- MPLS Edge devices
- Other devices with virtual routers

Context-sensitive discovery ensures correct representation of virtual routers. Always check that your particular device type is supported for discovery.

The **m_UseContext** value is set in the `DiscoSchema.domainName.cfg` file.

The **DetailsRetProcessing** stitcher runs the **DetailsOrContextRetProcToAgent.stch** and passes the appropriate arguments to ensure that the records move the correct way.

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 operate only on devices with OIDs that match their filter. If the device is not of a type that supports virtual routers, it does not need context-sensitive processing. In that case, it is passed directly to the **AssocAddress** agent.

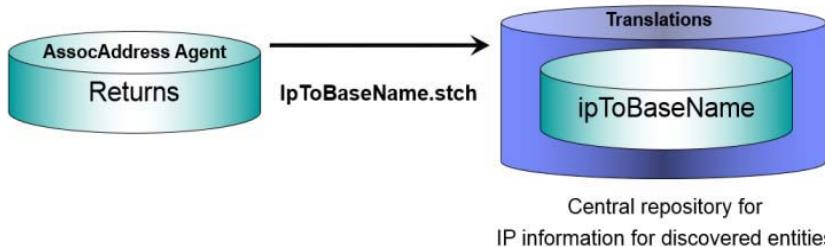
If a context-sensitive discovery is not necessary, then the **DetailsRetProcessing** stitchers move data from the Details agent **Details.returns** table to the **AssocAddress** agent. This movement

happens inside the **DetailsOrContextRetProcToAgent** stitcher. In this way, IP information can be stored in a central repository, ensuring that discovery does not reinterrogate discovered devices.

IpToBaseName.stch

This stitcher checks the **translations.ipToBaseName** table to determine whether the IP address of the current record exists this table

- If the address exists, the device was already interrogated
 - This stitcher prevents reinterrogating the device
- If the address does not exist in the table, the IP address is added to the table and then the device information is sent to other agents for further interrogation



Customizing discovery

© Copyright IBM Corporation 2017

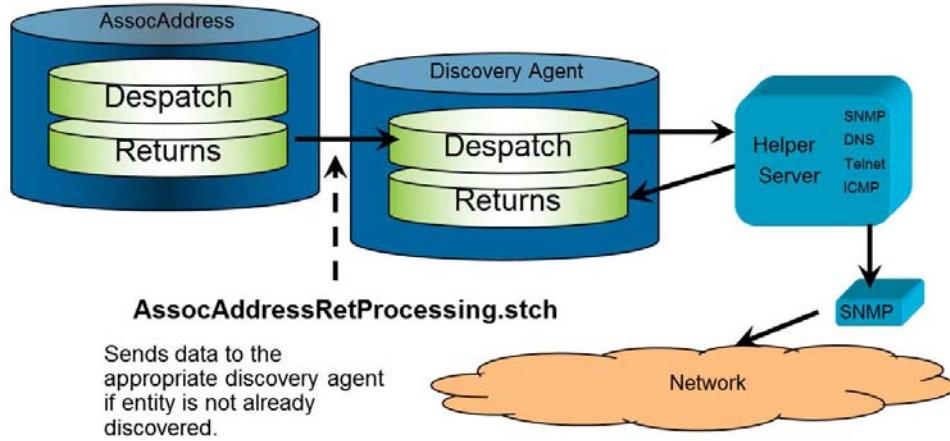
Figure 10-12. *IpToBaseName.stch*

The **translations.ipToBaseName** database table is the central repository for the IP address information that comes from the **AssocAddress** agent.

- The **AssocAddress** agent checks to see whether the addresses of a device are already in the **translations.ipToBaseName** table. If an address is already in this table, discovery does no further processing of this device.
- If a device IP address is not in the table, the AssocAddress agent gets a list of all IP addresses on the device and puts it into the **AssocAddress.returns** table.
- The **IpToBaseName.stch** stitcher takes the address in the **AssocAddress.returns** table and moves it into the **translations.ipToBaseName** table. This table contains all the IP addresses in the discovered network.

AssocAddressRetProcessing.stch

This stitcher sends addresses that are not found in the **translations.ipToBaseName** table to other discovery agents



Customizing discovery

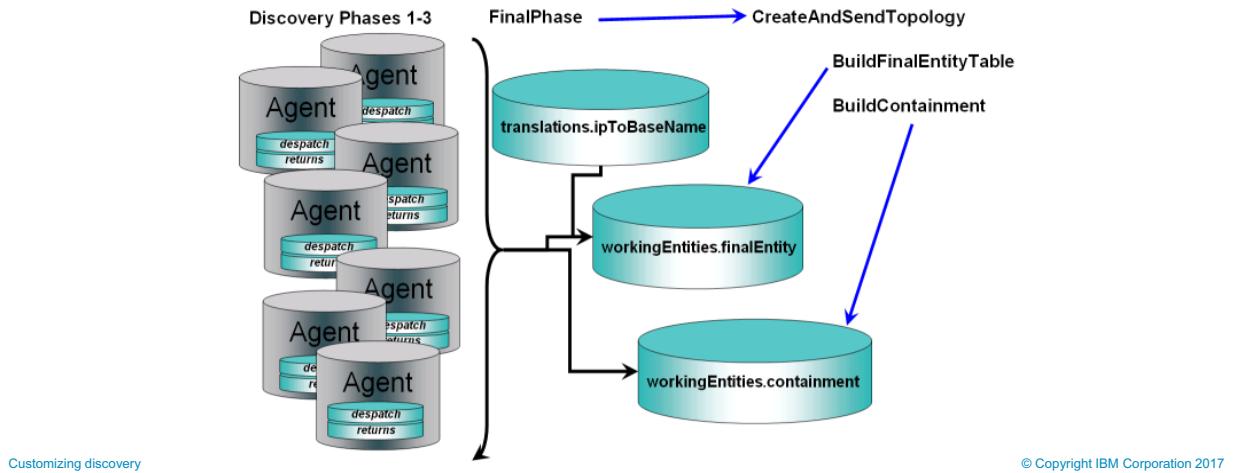
© Copyright IBM Corporation 2017

Figure 10-13. AssocAddressRetProcessing.stch

After the **AssocAddress** agent verifies whether the discovery process already found an entity, the **AssocAddressRetProcessing** stitcher sends addresses of devices that require further interrogation to the appropriate discovery agents.

Final-phase stitchers define entities and containment

- Final-phase stitching begins by building tables of entities and the interfaces that those entities contain
- A subsequent step builds the **fullTopology.entityByNeighbor** table that contains connection information



Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-14. Final-phase stitchers define entities and containment

After the discovery finishes Phase 3, it moves into the topology deduction phase. This phase is also Phase 4 or the final phase.

During this phase, a series of final-phase stitchers manipulates the discovered device information. The **DiscoRequiresLastPhase()** trigger mechanism starts these stitchers.

Three tables are built:

- **workingEntities.finalEntity** (contains entity information)
- **workingEntities.containment** (contains containment information)
- **fullTopology.entityByNeighbor** (contains connectivity information)

After this stitcher establishes the state of discovery, it runs the **PopulateDNCIM** stitcher that populates the DNCIM database. The **PopulateDNCIM** stitcher then calls other stitchers that send the topology data to the NCIM database and the MODEL service.

Passing entity data to the DNCIM database

- After the DISCO process builds a list of entities and their containment, data is sent to the DNCIM database
- Stitchers in the `$ITNMHOME/disco/stitchers/DNCIM` directory process data and populate DNCIM tables
- At the final phase of processing DNCIM data, you can make discovery customizations:
 - Use `$ITNMHOME/scripts/sql/sqlite/createCustomization.sql` to create custom tables that go into the NCIM database or that are used only in the DNCIM database as lookup tables
 - Create custom stitchers (such as `CustomEnrichmentNew.stch`) in `$ITNMHOME/disco/stitchers/DNCIM`
 - Edit `$ITNMHOME/disco/stitchers/PopulateDNCIM.stch` to call your new custom stitcher
- If you create a new custom table (such as `customData`) that goes into your NCIM database, you must define that table in the `$NCHOME/etc/precision/ModelNcimDb.cfg` file

Figure 10-15. Passing entity data to the DNCIM database

Topic summary

After you complete this section, you can do the following tasks:

- Explain the details of the final phase of discovery
- Identify what causes a blackout state in discovery and explain what happens to devices that are found during the blackout state
- Identify how many tables each agent has
- List the steps to add custom data to discovery

Figure 10-16. Topic summary

10.2. Stitcher language constructs



Figure 10-17. *Stitcher language constructs*

Stitcher language constructs

After you complete this section, you can do these tasks:

- List four ways to start a stitcher
- List common database operations that you can do in a stitcher
- List the default values for initializing variables
- Identify the basic parts of a stitcher rules file

Figure 10-18. *Stitcher language constructs*

Types of discovery stitchers

- **CompiledStitcher{ }**

Declares the stitcher as a compiled stitcher and introduces its trigger conditions

- **UserDefinedStitcher{ }**

Declares the stitcher as a user-defined stitcher and introduces its trigger conditions and stitcher rules

Figure 10-19. Types of discovery stitchers

The compiled stitchers are identified with the **CompiledStitcher{}** statement, which encloses the trigger conditions.

Text-based discovery stitchers are identified as text-based with the **UserDefinedStitcher{}** statement. This statement encloses the trigger conditions and the stitcher rules, which do the actual stitcher processing.

Discovery stitchers

- Stitchers move data between discovery tables
- Text-based or compiled
- Perl-like functions and OQL commands
 - **SELECT**
 - **INSERT**
 - **DELETE**
 - **UPDATE**

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-20. Discovery stitchers

Discovery stitchers are text-based processes, and they help the manipulation of data that is retrieved during discovery.

Stitchers can be text-based and provide intuitive rules for data manipulation. Compiled stitchers protect functions that cannot be altered without damaging the software.

Stitchers are in the `ITNMHOME/disco/stitchers` directory.

The stitcher language is like Perl but contains Object Query Language (OQL) statements to select, input, delete, and update data records.

Stitcher data types

- **int:** Integer values
- **text:** Stores string values
- **Record:** Stores a single returned record
- **RecordList:** Stores lists that the **RetrieveOQL** function creates

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-21. Stitcher data types

Stitcher assignment operators

- For all data types the equal sign (=) assigns a value
- For the integer data type, you can use the following operators:
 - + Addition
 - Subtraction
 - $+=$ Incremental addition
 - $=-$ Incremental subtraction
 - $++$ Increase by one
 - $--$ Decrease by one

Figure 10-22. Stitcher assignment operators

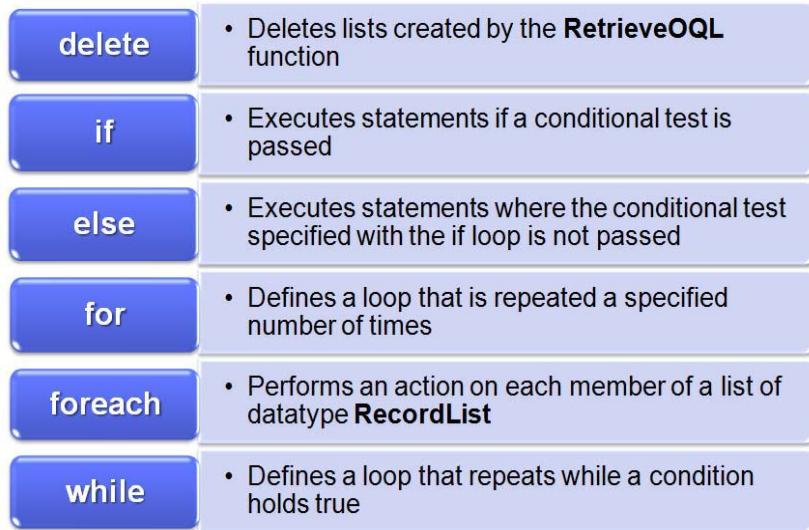
Stitcher relational operators

- The following list describes the relational operators:

== test for equality
!= test for inequality
< test for less than
> test for greater than
<= test for less than or equal to
>= test for greater than or equal to

Figure 10-23. Stitcher relational operators

Stitcher language constructs



Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-24. Stitcher language constructs

Stitcher language constructs are illustrated as follows:

- **delete** (*name of variable of type Record or RecordList*);
- **if** (*some_condition*)
 - {*list of stitcher rules to execute if the condition is true*}
- **else** (*list of stitcher rules to run if the condition is false*)
- **for** (*variable assignment, conditional test, change to variable*);
 - {*list of Stitcher rules to execute*}
- **foreach** (*variable of type RecordList*)
 - {*list of stitcher rules to execute*}
- **while** (*conditional test*)
 - {*list of stitcher rules to execute*}

Creating a RecordList with RetrieveOQL

- `RecordList listName = RetrieveOQL ("query;");`
- Populate a **RecordList** by retrieving data that is defined in the custom table:

```
RecordList LookupData = RetrieveOQL  
("SELECT * from ncim.lookup;");
```

- Easy to use, but less efficient than **PrepareSQL** for processing large amounts of data
- Can be used only to retrieve data with a select statement

Figure 10-25. Creating a RecordList with RetrieveOQL

Creating a RecordList with SQLData and PrepareSQL

- **SQLData** (*temporaryProcedureName*) = **PrepareSQL** ("querySyntax", "databaseName", evalStatements);

- **Example**

```
SQLData LookupData = PrepareSQL("SELECT * FROM ncim.lookup", "NCIM");
ExecuteSQL(LookupData);
foreach (LookupData) { ... }
```

- **Features**

- Can be used for both select and insert statements
- Creates a temporary stored procedure and prepares to run an optimized query
- More efficient than using RetrieveOQL when you are processing large amounts of data

Figure 10-26. Creating a RecordList with SQLData and PrepareSQL

The following **insert** statement shows the use of the **PrepareSQL** function.

```
PrepareSQL = "INSERT INTO Rules (RuleName, RuleNote, FacID, CreationDate,
CreatedBy) &_ "VALUES('' & rulename & '' ,'' & ruledescrp &'','' & ruleparid &
'', '' & createDate & '' ,'' & createBy & '')"
```

Use **PrepareSQL** to define a query one time that can be used multiple times within a stitcher. The system optimizes the query.

Processing the RecordList

- Loop through the RecordList data with the `foreach` command

```
foreach (LookupData)  
{Object query language statements to process}
```

- As each record in the list is processed, that record is said to be **in scope**

- The in-scope record is referenced in `eval` statements with the `&` symbol

```
set ExtraInfo->m_Department = eval(text, '&m_Location')
```

Figure 10-27. Processing the RecordList

Stitcher structure

Stitchers are text-based processes that do these functions:

- Take information from a database
- Process the information
- Put the modified information in other database tables or databases
- Can send information to other processes

// Stitcher.stch

Type of stitcher

Stitcher Trigger

Stitcher Rule

Figure 10-28. Stitcher structure

Discovery and monitoring processes use stitchers extensively. The discovery stitchers that are used during the network discovery process are in the `$ITNMHOME/disco/stitchers` directory.

Using variables in the stitcher rule

```
<variable data type> <variable name> = <initial value>;
```

```
// Stitcher.stch
Type of stitcher
Stitcher Trigger
Stitcher Rule
Variable declarations
Content
```

- Declare variables by specifying data type and the initial value
 - The initial value can **NULL**
- You can use a declared variable throughout the current stitcher
 - The variable can accept only values of the appropriate data type
- You can assign a value to a variable that results from an **eval** statement or a calculation

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-29. Using variables in the stitcher rule

You can declare variables anywhere within the **StitcherRules{}**. When you declare a variable, assign it an initial value. This value can be **NULL**.

You can use a declared variable throughout the current stitcher. The variable can accept only values of the appropriate data type.

You can update variable values at any time. You can also assign the result of an **eval** statement or a calculation to a variable.

Stitcher triggers

- The stitcher trigger conditions determine when a stitcher runs
- It is possible to enclose multiple stitcher triggers within the **StitcherTrigger{}** section
 - Each trigger statement ends with a semicolon (;)
- You can activate a stitcher in four ways:
 - Timed interval
 - User OQL insert into the **stitchers.actions** table
 - Stitchers can call other stitchers
 - Value that is inserted into a monitored table

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-30. *Stitcher triggers*

The stitcher trigger conditions specify the conditions that must be met in order for the stitcher to be run. The trigger conditions are defined within the **StitcherTrigger{}** statement.

It is possible to enclose multiple stitcher triggers within the **StitcherTrigger{}** section. Each of these triggers must terminate with a semicolon (;).

Generic stitcher triggers

- **ActOnTableInsert();**
 - Runs when data is inserted in the specified database table
- **ActOnTimedTrigger();**
 - Runs on a specified frequency
 - **m_DayOfWeek, m_DayOfMonth, m_TimeOfDay, m_Interval**
- **RequiresStitchers();**
 - Runs when the specified stitcher completes
- **ActOnDemand();**
 - Runs when started by the user or another stitcher

Figure 10-31. Generic stitcher triggers

Discovery stitcher triggers

- **DiscoRequiresAgents () ;**
 - Runs when the specified discovery agent completes
- **DiscoRequiresLastPhase () ;**
 - Can be run only in the last discovery cycle phase
- **DiscoRequiresPhase () ;**
 - Can be run only in the specified discovery cycle phase

Figure 10-32. Discovery stitcher triggers

Generic stitcher rules (1 of 2)

- **Print(argument1, argument2);**
 - Prints the specified variables to the standard output
 - Output goes to the process log file
- **PrintRecord();**
 - Prints the record currently being processed
- **count = MatchPattern();**
 - Returns a 0 if no records are found to match the pattern
 - Returns a 1 if matching records are found

Figure 10-33. Generic stitcher rules (1 of 2)

Generic stitcher rules (2 of 2)

- **RetrieveOQL();**
 - Creates a list of the data type RecordList generated by an OQL statement
- **RetrieveSingleOQL();**
 - Runs an OQL statement and retrieves the first record returned
- **DiscoRefresh();**
 - Instructs the finders to restart their operation when you run a rediscovery
- **IsRecordInFilter();**
 - Determines whether the record passes the detection or instantiation filter

Figure 10-34. Generic stitcher rules (2 of 2)

FnderProcToDetailsDesp.stch

```
//Stitcher takes entities from the finders
//processing table, and sends it to the details
//agent Details.despatch table to
//populate required fields such as sysDescr
UserDefinedStitcher
{StitcherTrigger
    {ActOnTableInsert("finders", "processing");}
    StitcherRules
    {ExecuteOQL ...
    }
}
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-35. FnderProcToDetailsDesp.stch

Using fields from a trigger

```

StitcherRules
{
  ExecuteOQL
    ("insert into Details.despatch
      (m_Name,
       m_UniqueAddress,
       m_PhysAddr,
       m_Protocol,
       m_AddressSpace)
     values
      (eval(text, '&m_Name'),
       eval(text, '&m_UniqueAddress'),
       eval(text, '&m_PhysAddr'),
       eval(int, '&m_Protocol'),
       eval(text, '&m_AddressSpace'));" )
}
  
```

& represents the field values from the in-scope record that triggers this stitcher

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-36. Using fields from a trigger

Stitcher rules are defined to insert each found record into the next database table. This database table is the **Details.despatch**.

The & represents the field values from the record that trigger this stitcher.

Evaluation statement

```
eval (<data type>, <variable or database column to be evaluated>)
```

- The first argument that is supplied to the `eval` statement is the data type into which the second argument is converted
- An ampersand (&) references a column name in the current record that the process is evaluating
`eval(text, '&m_UniqueAddress')`
- A \$ value in a stitcher represents a temporary variable that is defined and used only inside that stitcher

```
$ipAddress
```

Figure 10-37. Evaluation statement

The stitcher text files consist of a series of structural statements that enclose the stitcher rules (the actual processing) and the trigger conditions.

The `eval` statement extracts information from variables to use within the rules.

Other services add fields to NCIM entity records

ObjectId

- Random unique identifier (**NmosObjectInst** in events)

ClassName

- Based on Active Object Class (AOC) definitions and derived from the CLASS service
- Used to categorize discovered entities and determine icon representation

LingerTime

- Defined in `$NCHOME/etc/precision/ModelSchema.cfg` file with a default value of 3
- Each time discovery runs without finding a device, the **LingerTime** decrements by 1
- Devices are deleted when the value decrements to -1
- Change default value to a number that is suitable for your environment
 - How often do you run a full discovery?
 - How dynamic is your network?
 - How reliable is your network?

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-38. Other services add fields to NCIM entity records

How long are you willing to wait after devices disappear from your network before the icons are removed from your topology view? You can delete individual devices from your topology manually. If you remove devices that provide connectivity, run a full or partial rediscovery with settings to rediscover surrounding devices and connections.

Topic summary

After you complete this section, you can do the following tasks:

- List four ways to start a stitcher
- List common database operations that you can do in a stitcher
- List the default values for initializing variables
- Identify the basic parts of a stitcher rules file

Figure 10-39. Topic summary

10.3. Creating database tables

You can use a database table to store business information. You can use this information to enhance discovery information.

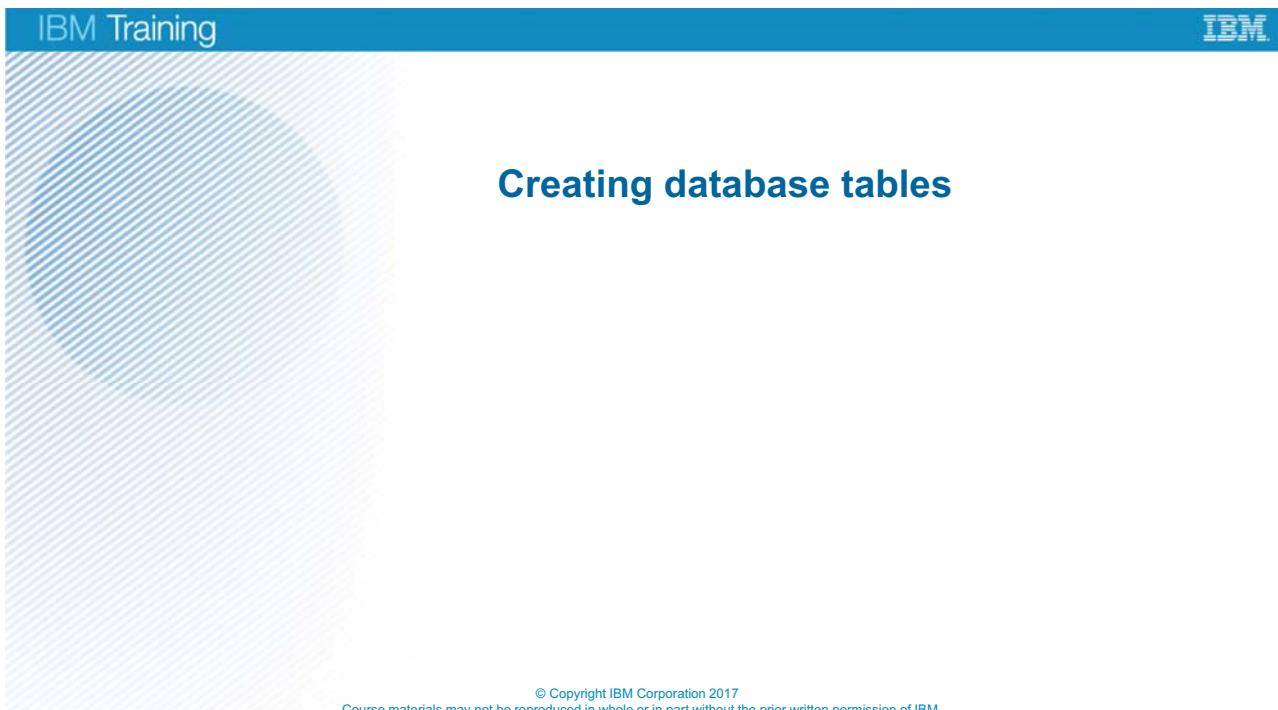


Figure 10-40. Creating database tables

Creating database tables

After you complete this section, you can do the following tasks:

- Create and populate lookup tables with data
- Create a custom table that stores enriched discovery information

Figure 10-41. Creating database tables

Business scenario

- A customer wants to add geographic information to discovered network devices
- The customer wants you to enrich the topology with the following fields: **Continent**, **Country**, and **City**
- You can derive this enrichment information by parsing the host names and assigning field values in custom tables

```
## Primary IPs
10.10.255.1    BRU-Core-01    IBM
10.10.255.15   BRU-PE-01     IBM
10.10.255.10   BRU-SW-01     IBM
10.10.255.12   BRU-CPE-01    IBM
10.10.255.14   BRU-NAS-02    IBM
10.10.255.11   BRU-ACS-01    IBM
10.10.255.16   BRU-PE-02     IBM
10.10.255.13   BRU-NAS-01    IBM
10.10.255.17   BRU-PE-03     IBM
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-42. Business scenario

Create custom tables

- To create new tables in the DNCIM database, edit
\$ITNMHOME/scripts/sql/sqlite/createCustomization.sql
- If you use this same table in the NCIM database, you must create the table in the NCIM database and then define the table in \$NCHOME/etc/precision/ModelNcimDb.cfg

```
CREATE TABLE dncim.customData
(
    entityId      integer not null,
    slaId        VARCHAR(255),
    Continent     varchar(64),
    Country       varchar(64),
    City          varchar(64),
    DeviceType    varchar(16),

    -- Following line specifies the primary key field for the table CONSTRAINT customData_pk
    PRIMARY KEY (entityId),
    -- Following line creates a constraint named customData_entData_fk
    -- The constraint links the entityId field in table to the
    -- field of the same name in the entityData table.
    -- If an entityId record is deleted in the entityData table,
    -- the delete cascades to the dncim.customData table to delete
    -- the corresponding record. This maintains referential integrity. CONSTRAINT customData_entData_fk
    FOREIGN KEY (entityId) REFERENCES entityData(entityId) ON DELETE CASCADE
);
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-43. Create custom tables

```
CREATE TABLE dncim.customData
(
    entityId      integer not null,
    slaId        VARCHAR(255),
    Continent     varchar(64),
    Country       varchar(64),
    City          varchar(64),
    DeviceType    varchar(16),

    -- Following line specifies the primary key field for the table
    CONSTRAINT customData_pk PRIMARY KEY (entityId),
    -- Following line creates a constraint named customData_entData_fk
    -- The constraint links the entityId field in table to the
    -- field of the same name in the entityData table.
    -- If an entityId record is deleted in the entityData table,
    -- the delete cascades to the dncim.customData table to delete
    -- the corresponding record. This maintains referential integrity.
    CONSTRAINT customData_entData_fk FOREIGN KEY (entityId)
    REFERENCES entityData(entityId) ON DELETE CASCADE
);
```

Create lookup tables that are used for stitcher processing (1 of 2)

- Use `$ITNMHOME/scripts/sql/sqlite/createCustomization.sql` to build these tables
- This script is used only when no DNCIM tables exist (follow steps in slide notes if you modify the file later)

```

CREATE TABLE dncim.locationLookup
(
    CityCode      char(3) not null,
    Continent     varchar(64),
    Country       varchar(64),
    City          varchar(64),
    CONSTRAINT locationLookup_pk PRIMARY KEY (CityCode)
    -- No reference to a foreign table is required in this section
    -- since this data is a permanent lookup table.
);
INSERT INTO dncim.locationLookup (CityCode, Continent, Country, City) VALUES
('BRU','Europe','Belgium','Brussels'),
('LON','Europe','United Kingdom','London'),
('MOS','Europe','Russia','Moscow'),
('SYD','Australia','Australia','Sydney'),
('PAR','Europe','France','Paris'),
('NYC','North America','United States','New York City'),
('WAS','North America','United States','Washington DC');

```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-44. Create lookup tables that are used for stitcher processing (1 of 2)

In the previous version of Tivoli Network Manager, you edited the `$ITNMHOME/scripts/sql/sqlite/populate_solid_dncim.sh` script. Now, you put all your SQL statements for creating tables and inserting values into `$ITNMHOME/scripts/sql/sqlite/createCustomization.sql`, and Tivoli Network Manager automatically processes the statements in this file.

This SQL script is run when no DNCIM tables exist. For example, the **locationLookup** table in the preceding example is created when the first discovery is run if no data exists in the `$ITNMHOME/embeddedDb/sqlite` directory. If you modify the `createCustomization.sql` script, do the following steps:

1. Stop Tivoli Network Manager (`itnm_stop ncp`).
2. Delete the contents of the `$ITNMHOME/embeddedDb/sqlite` directory.
`rm - rf $ITNMHOME/embeddedDb/sqlite/ncp_disco.domainName`
3. Restart Tivoli Network Manager (`itnm_start ncp`).

4. Run a full discovery.

```
CREATE TABLE dncim.locationLookup
(
    CityCode      char(3) not null,
    Continent     varchar(64),
    Country       varchar(64),
    City          varchar(64),
    CONSTRAINT locationLookup_pk PRIMARY KEY (CityCode)
);
INSERT INTO dncim.locationLookup (CityCode, Continent, Country, City)
VALUES
('BRU', 'Europe', 'Belgium', 'Brussels'),
('LON', 'Europe', 'United Kingdom', 'London'),
('MOS', 'Europe', 'Russia', 'Moscow'),
('SYD', 'Australia', 'Australia', 'Sydney'),
('PAR', 'Europe', 'France', 'Paris'),
('NYC', 'North America', 'United States', 'New York City'),
('WAS', 'North America', 'United States', 'Washington DC');
```

Create lookup tables that are used for stitcher processing (2 of 2)

```

CREATE TABLE dncim.deviceType
(
    DevCode          varchar(16) not null,
    DeviceType       varchar(64),
    CONSTRAINT deviceType_pk PRIMARY KEY (DevCode)
);

INSERT into dncim.deviceType
(DevCode, DeviceType)
VALUES
('Core','Backbone'),
('PE','Provider Edge'),
('CPE','Customer Premise Equipment'),
('SW','Switch with Routing Blade'),
('NAS','Network Access Server'),
('ACS','Secure Access Control Server');

```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-45. Create lookup tables that are used for stitcher processing (2 of 2)

```

CREATE TABLE dncim.deviceType
(
    DevCode          varchar(16) not null,
    DeviceType       varchar(64),
    CONSTRAINT deviceType_pk PRIMARY KEY (DevCode)
);
INSERT into dncim.deviceType
(DevCode, DeviceType)
VALUES
('Core','Backbone'),
('PE','Provider Edge'),
('CPE','Customer Premise Equipment'),
('SW','Switch with Routing Blade'),
('NAS','Network Access Server'),
('ACS','Secure Access Control Server');

```

10.4. Prepare stitchers to enrich discovery

Stitchers move data from one location to another location. With a custom stitcher, you can add information from external sources to the discovery. The exercise for this unit takes you through the steps of creating a custom stitcher to enhance discovery information.

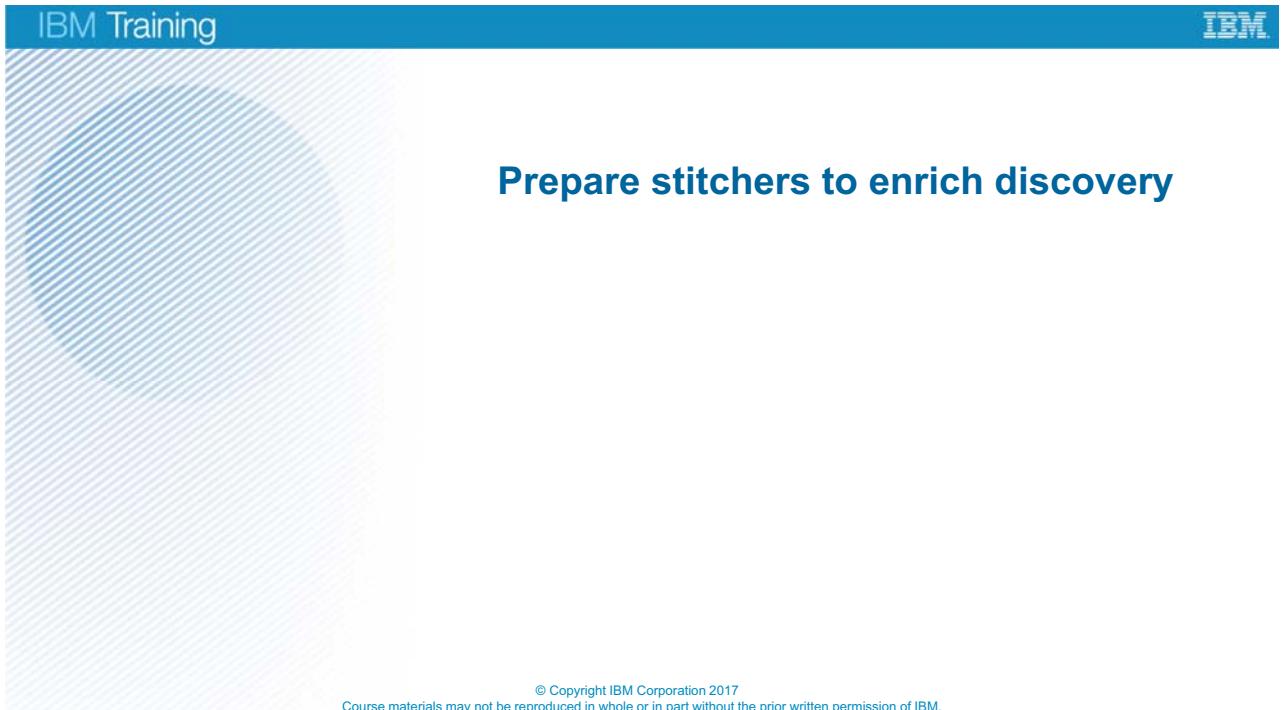


Figure 10-46. Prepare stitchers to enrich discovery

Edit the \$ITNHOME/disco/stitchers/DNCIM/PopulateDNCIM.stch

- This stitcher populates the DNCIM discovery tables and calls your custom stitchers
 - At the end of this stitcher, call your custom discovery stitcher (such as **CustomEnrichmentNew**)
- Use **StitcherTimeCheck** statements to create entries in
\$NCHOME/log/precision/ncp_disco.domainName.trace to show whether the stitcher completed its work successfully

```
ExecuteStitcher('PopulateDNCIM_ManagedStatus', domainId, isRediscovery,
dynamicDiscoNode);
StitcherTimeCheck('DNCIM managed status', 'CustomEnrichmentNew', 97);
ExecuteStitcher('CustomEnrichmentNew');
StitcherTimeCheck('CustomEnrichmentNew', 'SendingDNCIMChangestoModel', 98);
ExecuteStitcher('SendDNCIMChangesToModel');
StitcherTimeCheck('Sending DNCIM to model' , 'Next Discovery', 100);
```

Figure 10-47. Edit the \$ITNHOME/disco/stitchers/DNCIM/PopulateDNCIM.stch

The **StitcherTimeCheck** statement creates entries in the **\$NCHOME/log/precision/ncp_disco.domainName.trace** file that show that the specified stitcher completed its work.

```
ExecuteStitcher('PopulateDNCIM_ManagedStatus', domainId, isRediscovery,
dynamicDiscoNode);
StitcherTimeCheck('DNCIM managed status', 'CustomEnrichmentNew', 97);
ExecuteStitcher('CustomEnrichmentNew');
StitcherTimeCheck('CustomEnrichmentNew', 'SendingDNCIMChangestoModel', 98);
ExecuteStitcher('SendDNCIMChangesToModel');
StitcherTimeCheck('Sending DNCIM to model' , 'Next Discovery', 100);
```

When you run the stitcher, it generates entries in the log file.

```
CustomEnrichmentNew Completed. SendingDNCIMChangestoModel Starting Sat Sep 17
17:56:47 2016
```

Edit RefreshDNCIM.stch

- `$ITNMHOME/disco/stitchers/RefreshDNCIM.stch` truncates DNCIM tables for the current domain when a new full discovery runs
- If you create a stitcher to populate a custom table during discovery, make sure that `RefreshDNCIM.stch` truncates that table before you run a new discovery


```
SQLData customDataTrunc = PrepareSQL("TRUNCATE TABLE customData", "DNCIM");ExecuteSQL(customDataTrunc);
```
- Truncate tables that need to be refreshed each time discovery is run; do not truncate tables with status data
 - Truncate a `customData` table (that contains dynamic data that is altered during discovery)
 - Do not truncate a `locationLookup` table (because the lookup table contains static data that does not change with a new discovery)
- If you do alter a static table by editing `createCustomization.sql`, then you need to stop Tivoli Network Manager, delete all DNCIM data, restart Tivoli Network Manager, and run a full discovery

[Customizing discovery](#)

[© Copyright IBM Corporation 2017](#)

Figure 10-48. Edit RefreshDNCIM.stch

Build custom stitcher (such as CustomEnrichmentNew.stch) (1 of 9)

- Put this stitcher in the `$ITNHOME/disco/stitchers/DNCIM` directory
- Begin by initializing variables

```
UserDefinedStitcher
{ StitcherTrigger{}
  StitcherRules
  { // Initialize Variables
    int chassisId = 0;
    int entityId = 0;
    text chassisName = NULL;
    text domainPattern = NULL;
    text Continent = NULL;
    text Country = NULL;
    text City = NULL;
    text DeviceType = NULL;
    text DevCode = NULL;
    text CityCode = NULL;
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-49. Build custom stitcher (such as CustomEnrichmentNew.stch) (1 of 9)

```
UserDefinedStitcher
{ StitcherTrigger{}
  StitcherRules
  { // Initialize Variables
    int chassisId = 0;
    int entityId = 0;
    text chassisName = NULL;
    text domainPattern = NULL;
    text Continent = NULL;
    text Country = NULL;
    text City = NULL;
    text DeviceType = NULL;
    text DevCode = NULL;
    text CityCode = NULL;
```

Build custom stitcher (2 of 9)

- Only chassis entities appear in network maps
- So the custom stitcher needs to modify only entities where entityType=1

```
// The following statement will be evaluated as soon as it is read since we
// are passing all parameters to the PrepareSQL when we define it.
SQLData getChassis = PrepareSQL(
    "select entityName, entityId from entityData where entityType = 1;",
    "DNCIM"
);
```

Figure 10-50. Build custom stitcher (2 of 9)

The `$NCHOME/etc/precision/ModelNcimDb.cfg` specifies many values for **entityType**. However, the following values are the ones that are most commonly used.

- `entityType=1` is a chassis
- `entityType=2` is an interface
- `entityType=3` is a logical interface

Build custom stitcher (3 of 9)

```
// The following statement will be evaluated as soon as it is read since we
// are passing all parameters to the PrepareSQL when we define it.

SQLData getChassis = PrepareSQL(
    "select entityName, entityId from entityData where entityType = 1;",
    "DNCIM"
);

// In the following select statements, the ? prevents ITNM from evaluating the
// the select statement until we call it later. If we don't use that ?,,
// the statement evaluates now and will not be re-evaluated if we use it
// later in the script.

SQLData getLocation = PrepareSQL(
    "select Continent, Country, City from locationLookup where CityCode = ?",
    "DNCIM",
    eval(text,'$CityCode')
);
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-51. Build custom stitcher (3 of 9)

Using the question mark (?) when you define a stored procedure prevents the query from running until it is called later in the script. Each time the procedure is called, the query reevaluates the data from the current record.

Build custom stitcher (4 of 9)

```
// This section prepares 4 routines for inserting enrichment data into
// the entityDetails table.
// Define one procedure for each keyName-keyValue pair you insert into
// entityDetails table
SQLDATA insertCountry = PrepareSQL(
    "insert into entityDetails
     (      entityId,
            keyName,
            keyValue      )
   values
     (      ?,,
            ?,,
            ?           ) ;",
    "DNCIM",
    eval(text,'$chassisId'),
    "Country",
    eval(text,'$Country')
);
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-52. Build custom stitcher (4 of 9)

```
// This section prepares four routines for inserting enrichment data into
// the entityDetails table.
SQLDATA insertCountry = PrepareSQL(
    "insert into entityDetails
     (      entityId,
            keyName,
            keyValue      )
   values
     (      ?,,
            ?,,
            ?           ) ;",
    "DNCIM",
    eval(text,'$chassisId'),
    "Country",
    eval(text,'$Country')
);
```

Build custom stitcher (5 of 9)

```
ExecuteSQL(getChassis);
int count=0;
foreach(getChassis)
{
// In the stitcher, when you pull data from a database, the column
// names are returned in all uppercase letters. So refer to them
// as uppercase when you use them in statements.
chassisName = eval(text,'&ENTITYNAME');
chassisId = eval(int,'&ENTITYID');
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-53. Build custom stitcher (5 of 9)

When you retrieve records as the result of a query, the field names are returned in uppercase. To refer to the returned values, you must use the uppercase field name or your stitcher does not work.

```
ExecuteSQL(getChassis);
int count=0;
foreach(getChassis)
{
// In the stitcher, when you pull data from a database, the column
// names are returned in all uppercase letters. So refer to them
// as uppercase when you use them in statements.
chassisName = eval(text,'&ENTITYNAME');
chassisId = eval(int,'&ENTITYID');
```

Build custom stitcher (6 of 9)

```
// The following pattern looks for a host name
// with three initial capital letters followed by a
// hyphen and then some more text with mixed
// uppercase and lowercase followed by a hyphen and
// then some irrelevant alphanumeric characters.
// The value inside the first set of parentheses
// is $REGEX1, the second set is $REGEX2, and so on.
// $REGEX0 refers to the entire string.

domainPattern = "^[A-Z][A-Z][A-Z]-([A-Za-z]+)-.*";

// The following statement returns a value of 1 if
// the in-scope record matches the defined pattern.
// Since a foreach statement begins this section,
// the statement evaluates only one chassisName at a time.
count = MatchPattern(chassisName, domainPattern);
```

Figure 10-54. Build custom stitcher (6 of 9)

Build custom stitcher (7 of 9)

```

if(count > 0)
{
    CityCode = eval(text, '$REGEX1');
    DevCode = eval(text, '$REGEX2');
    Print ("City code: ", CityCode);
    Print ("DevCode: ", DevCode);
    ExecuteSQL(getLocation);
    ExecuteSQL(getDeviceType);

    // Retrieve the appropriate data for the entity.
    // Begin building the record to be inserted into
    // DNCIM. Only those lines that begin with
    // @customData are written into the customData
    // table. You first "turn on the recorder" to build
    // up the insert that goes into the customData
    // table.
    Record customData;
    @customData.entityId = eval(int, '$chassisId');
}

```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-55. Build custom stitcher (7 of 9)

The keyword **Record** starts a recorder with the specified name. All data assignments that follow the **Record** command begin with the **@** symbol to designate a temporary field assignment. A later command writes the data in the recording to a database table.

Build custom stitcher (8 of 9)

- Populate the temporary field that is represented by the @ symbol
- The `DncimInsert` command sends the temporary fields into the DNCIM database

```
// Enrich the values needed for entityDetails (for structure browser)
entityId = eval(int, '$chassisId');
Country = eval(text, '&COUNTRY');
Continent = eval(text, '&CONTINENT');
City = eval(text, '&CITY');
ExecuteSQL(insertContinent);
ExecuteSQL(insertCountry);
ExecuteSQL(insertCity);

// Add data to customData table (for network views)
@customData.Country = eval(text, '&COUNTRY');
@customData.Continent = eval(text, '&CONTINENT');
@customData.City = eval(text, '&CITY');
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-56. Build custom stitcher (8 of 9)

```
// Enrich the values needed for entityDetails (for structure browser)
entityId = eval(int, '$chassisId');
Country = eval(text, '&COUNTRY');
Continent = eval(text, '&CONTINENT');
City = eval(text, '&CITY');
ExecuteSQL(insertContinent);
ExecuteSQL(insertCountry);
ExecuteSQL(insertCity);

// Add data to customData table (for network views)
@customData.Country = eval(text, '&COUNTRY');
@customData.Continent = eval(text, '&CONTINENT');
@customData.City = eval(text, '&CITY');
```

Build custom stitcher (9 of 9)

```

// The following statement should cause an entry for each statement
// to go into the log file.
    PrintRecord("Custom data record is ", customData);
// Use chassisId as the first parameter. This first parameter is
// intrepreted as text. Following command writes recorder data to DNCIM db.
    DncimInsert( chassisId, "customData", customData );
// We delete the in-scope record after doing the insert. This prevents
// memory leaks and prevents bad data from going into the next iteration.
    delete( customData );
}
else
{
// Use a Print statement to create log file entries to record when data
// does not match pattern that was defined earlier
    // DNS Name does not match naming convention.
    Print("DNS name does not match naming convention", chassisName);
}

```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-57. Build custom stitcher (9 of 9)

The `PrintRecord` command is a helpful diagnostic tool when you test a new stitcher. It can print the value of the current record into the `$NCHOME/log/precision/ncp_disco.domainName.trace` file. Remove this command when the stitcher moves into production so that you do not use the log space.

The `DncimInsert` command in the preceding slide looks for the `chassisId` of the current record. It then writes all of the recorded fields (assigned earlier with the `@` symbol) from the `customData` recorder into the `customData` table of the DNCIM database. The `customData` table can pass to the NCIM database only if two conditions are true:

- The table exists in the NCIM database.
- The `$NCHOME/etc/precision/ModelNcimDb.cfg` file defines the `customData` table.

Always delete the data recorder. In this example, the `Record customData` statement started the recorder earlier in the stitcher. When you are through processing a record, you must delete all the temporary variable assignments in the recorder, or that memory remains allocated when you process the next record. The failure to clear the recorder data results in a memory leak that can negatively affect your system performance over time.

It is helpful to put a `Print` statement to write a message in the trace file when a record does not match your criteria. The message confirms that the stitcher processed that record but the record failed to match the criteria.

```
// The following statement should cause an entry for each statement
// to go into the log file.
    PrintRecord("Custom data record is ", customData);
// Use chassisId as the first parameter. This first parameter is
// intepreted as text. Following command writes recorder data to DNCIM db.
    DncimInsert( chassisId, "customData", customData );
// We delete the in-scope record after doing the insert. This prevents
// memory leaks and prevents bad data from going into the next iteration.
    delete( customData );
}
else
{
    // DNS Name does not match naming convention.
    Print("DNS name does not match naming convention", chassisName);
}
```

10.5. Run the custom scripts and stitchers

As soon as you create a custom stitcher, you need to run and test it. This unit explains how to run the custom stitcher and use log messages for testing the stitcher.

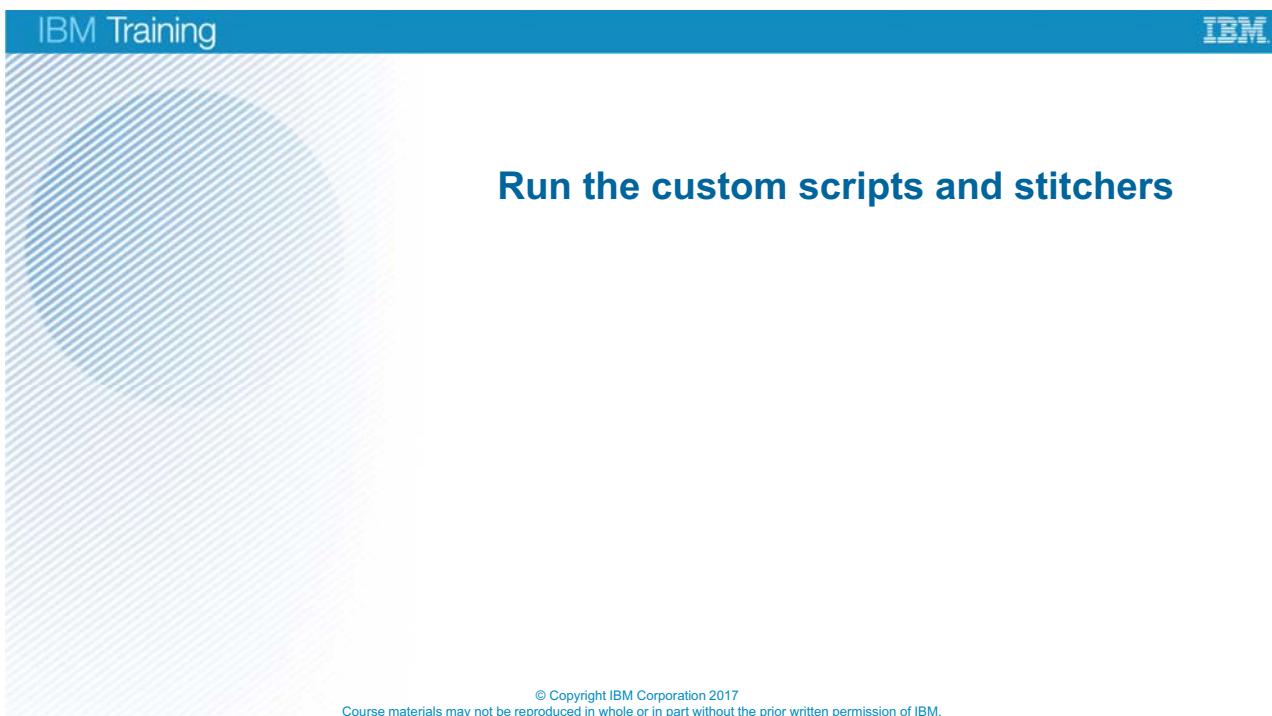


Figure 10-58. Run the custom scripts and stitchers

Build the new DNCIM database

- Stop Tivoli Network Manager core processes

```
itnm_stop ncp
```

- Remove the directory that contains the embedded DNCIM database

```
cd $ITNMHOME/embeddedDb/sqlite
rm -rf ncp_disco.DomainName
```

- Restart Tivoli Network Manager core processes

```
itnm_start ncp
```

- When discovery starts again, it runs the `createCustomization.sql` script to re-create any custom tables since no DNCIM database is found

- The `PopulateDNCIM.sh` script calls any custom stitchers that you added

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-59. Build the new DNCIM database



Hint

When you want to start a fresh discovery, use `itnm_stop ncp` to shut down the Tivoli Network Manager server. This command stops only the core processes. If you use `itnm_stop`, then the Apache Storm server that is used for polling data aggregation must also stop. Shutting down the Apache Storm server adds time to the stopping and restarting of Tivoli Network Manager. In some cases, if you do not allow adequate time for an orderly shutdown of all processes, the Apache Storm server does not restart correctly when you restart Tivoli Network Manager. For this reason, it is best to stop only the essential Tivoli Network Manager processes unless you have a compelling reason to stop everything.

Verify that custom DNCIM tables populated

- Type the following command to log on to the DNCIM SQL database:

```
ncp_oql -domain domainName -service DNCIM -username root  
-password admin -query "select * from locationLookup;"
```

- Run a query to see whether the data is there:

```
select * from locationLookup  
go  
{  
    CITY='Brussels';  
    CITYCODE='BRU';  
    CONTINENT='Europe';  
    COUNTRY='Belgium';  
}  
...
```

Figure 10-60. Verify that custom DNCIM tables populated

Substitute the correct password for your system if the administrator changed it. In previous versions of Network Manager, you used the `-dbId` option. However, now you need to specify the `-service DNCIM` option only to view results from the DNCIM database.

Define custom tables in \$NCHOME/etc/precision/ModelNcimDb.cfg

Custom tables must be defined in **ModelNcimDb.cfg** as part of the **dbModel** database

```
///////////////////////////////
//Create definition for customData table to be used in class exercies
// Note that definitions here use more generic data types compared
// to what is in the sql files used to create the tables in the
// databases.
create table dbModel.customData
(
    entityId      int  not null primary key,
    slaid        int,
    Continent    text,
    Country      text,
    City         text,
    DeviceType   text
);.....
```

Figure 10-61. Define custom tables in \$NCHOME/etc/precision/ModelNcimDb.cfg

Populate custom table with data in ModelNcimDb.cfg

```
///////////////////////////////
// For lab exercise
insert into dbModel.entityMap
( EntityFilter,
  TableName,
  FieldMap,
  StitcherDefined )
values
( "m_EntityType = 1 OR m_EntityType = 8",
  "customData",
///Field names here are not in all UPPERCASE letters because these statements do
///simple data mapping and do not retrieve query results.
{ entityId = "eval(int, '&m_EntityId')",
  Continent = "eval(text, '&Continent')",
  Country = "eval(text, '&Country')",
  City = "eval(text, '&City')",
  DeviceType = "eval(text, '&DeviceType')",
  1
);
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-62. Populate custom table with data in ModelNcimDb.cfg

```
///////////////////////////////
// For lab exercise
insert into dbModel.entityMap
( EntityFilter,
  TableName,
  FieldMap,
  StitcherDefined )
values
( "m_EntityType = 1 OR m_EntityType = 8",
  "customData",
///Field names here are not in all UPPERCASE letters because these statements do
///simple data mapping and do not retrieve query results.
{ entityId = "eval(int, '&m_EntityId')",
  Continent = "eval(text, '&Continent')",
  Country = "eval(text, '&Country')",
  City = "eval(text, '&City')",
  DeviceType = "eval(text, '&DeviceType')",
  1
);
```

Restart the MODEL service

- When you modify the `$NCHOME/etc/precision/ModelNcimDb.cfg` file, restart **ncp_model** to make it read the modified file
 - Always make a backup copy of the original file before you edit it
 - If you forget to make a backup copy of a configuration file in `$NCHOME/etc/precision`, the default files for Tivoli Network Manager are in `$NCHOME/etc/precision/default`
- To restart **ncp_model**, stop the process


```
pkill ncp_model
```

 - The CTRL service automatically restarts **ncp_model**
 - When **ncp_model** restarts, it reads the updated configuration file
 - If `ModelNcimDb.cfg` contains any syntax errors, **ncp_model** does not restart
 - In this event, check the **log** and **trace** files for the **ncp_model** process
 - When domain-specific copies of Tivoli Network Manager configuration or stitcher files exist, they are always used instead of the generic files

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-63. Restart the MODEL service



Hint

It is a good idea to always make a backup copy of any file you are editing. For example, if you are editing `$NCHOME/etc/precision/ModelNcimDb.ITNM42GO.cfg`, you can take these steps:

```
cd $NCHOME/etc/precision
cp ModelNcimDb.ITNM42GO.cfg ModelNcimDb.ITNM42GO.cfg.orig
vi ModelNcimDb.ITNM42GO.cfg
```

If you have several domains that are running on the same server and want all domains to have the same change, edit the generic file `ModelNcimDb.cfg` and delete the domain-specific files. Do this procedure only if you have no existing domain-specific modifications that apply to some domains, but not all domains on the server.

Create Network Views with any **keyName-keyValue** pair in **entityDetails**

- Beginning with Tivoli Network Manager 4.2, any **keyName-keyValue** pair in the **entityDetails** table can be used for creating Network Views
- This feature does not require the modification of the **ncimMetaData.xml** file

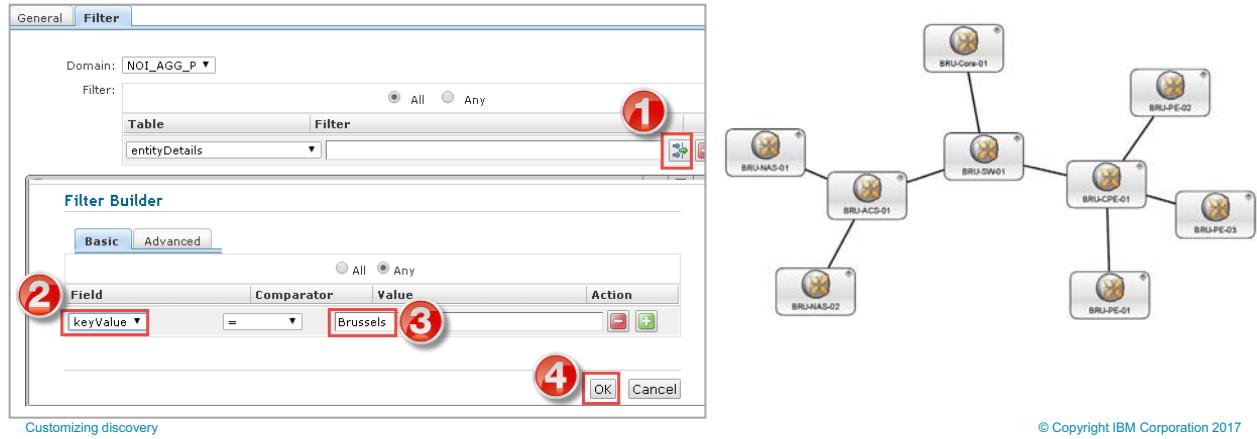


Figure 10-64. Create Network Views with any *keyName-keyValue* pair in *entityDetails*

To make the new fields of the custom table available in the GUI, the fields must be added to the **ncimMetaData.xml** file. Restart Tivoli Integrated Portal to make it read the modified file.

Modify ncimMetaData.xml to use custom table fields

- `/opt/IBM/netcool/gui/precision_gui/profile/etc/tnm/ncimMetaData.xml` controls what fields are available for Network Views
- Add information about your custom tables near the bottom of this file

```
<entityMetaData table="customData" manager="AllManagers" entitySearch="true">
    <dataField dataType="str" column="entityId"/>
    <dataField dataType="str" column="Continent"/>
    <dataField dataType="str" column="Country"/>
    <dataField dataType="str" column="City"/>
    <dataField dataType="str" column="DeviceType"/>
</entityMetaData>
```

- After you restart the Jazz SM server, the **customData** database and its fields are available selections in the GUI to create network views based on the data in that table

Figure 10-65. Modify ncimMetaData.xml to use custom table fields



Hint

To make sure that your angle brackets and punctuation are correct, copy the example lines that begin with `<entityMetaData` and end with `</entityMetaData>`. Then, substitute field names and data types as needed.

Build new views with enriched information

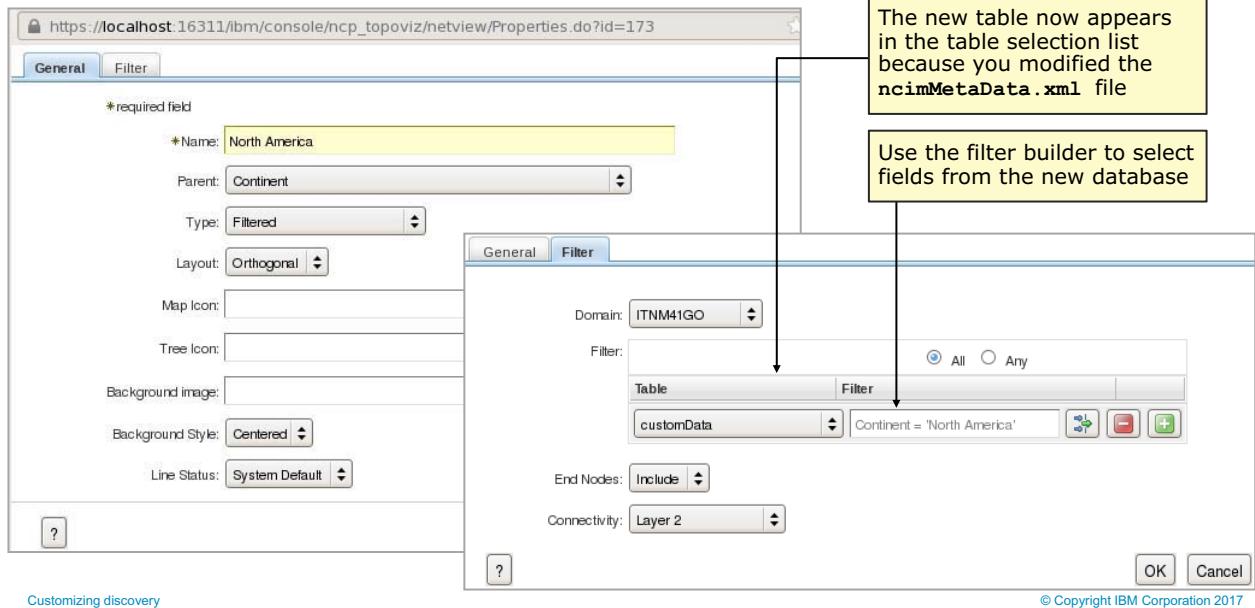
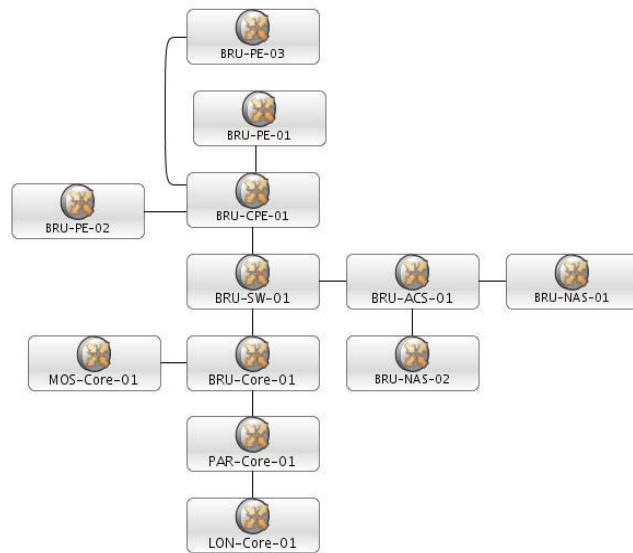


Figure 10-66. Build new views with enriched information

Build useful views



Dynamic views update
as topology changes



Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-67. Build useful views

Topic summary

After you complete this topic, you can do the following tasks:

- Create a custom stitcher
- Create and populate a custom table with information
- Create SQL scripts to populate information into the DNCIM database
- Use a custom stitcher to add business information during a discovery

Figure 10-68. Topic summary

10.6. Using the PresetLayer stitcher

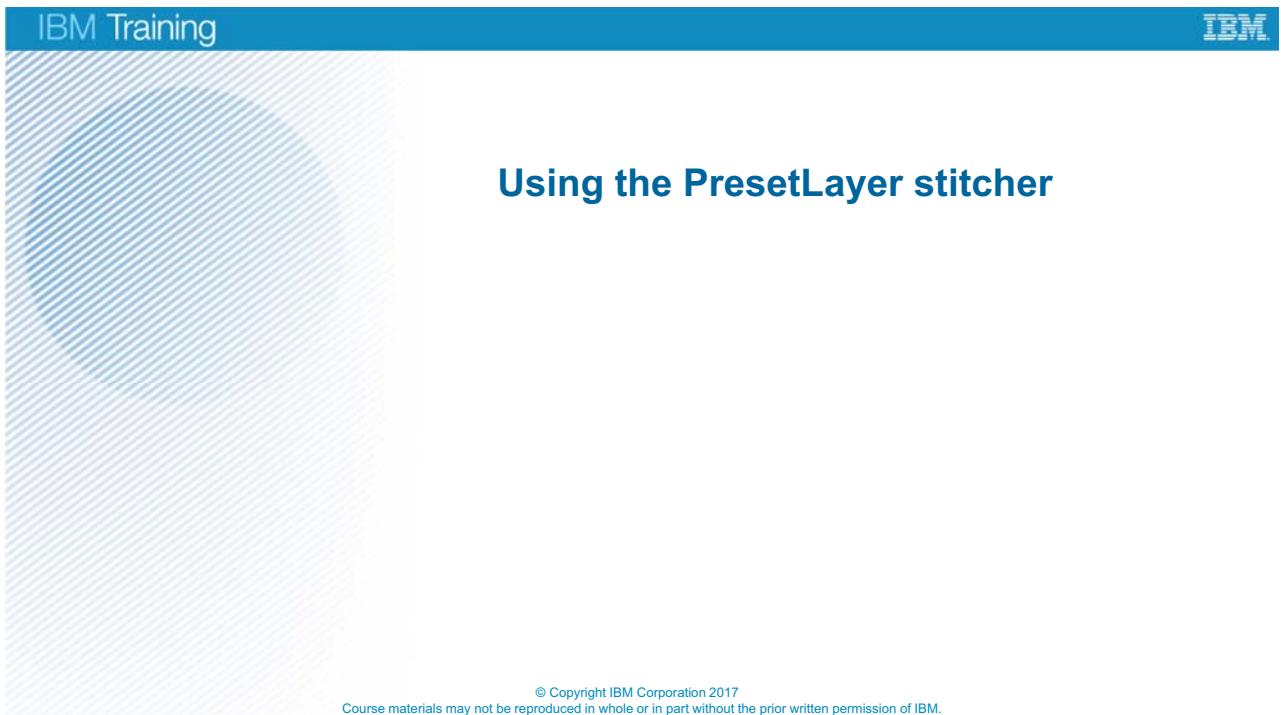


Figure 10-69. Using the PresetLayer stitcher

Using the PresetLayer stitcher

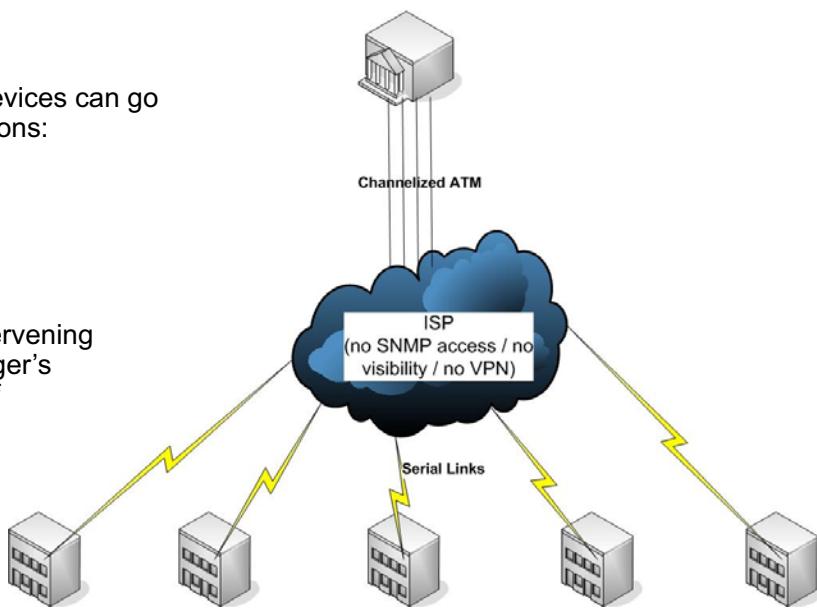
After completing this topic, you should be able to use the **PresetLayer** stitcher to create these topology elements:

- Chassis entities
- Interface entities
- Containment relationships
- Connections between devices

Figure 10-70. Using the PresetLayer stitcher

Islands of connectivity

- Within the network, many devices can go undiscovered for these reasons:
 - No SNMP access
 - Access control list blocking
 - No ICMP response
- With no visibility into the intervening connectivity, Network Manager's map has separate islands of discovery
 - RCA functions only within each island



Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-71. Islands of connectivity

Sometimes companies employ third parties to provide connectivity between their branch sites. Consider this scenario:

- A company has 64 locations.
- From their headquarters (HQ) site, they run an ATM interface to their ISP. The company configures 16 subinterfaces on their ATM interface.
- If the company establishes VPNs between sites, then Network Manager discovers the company network but the intervening third-party equipment is represented as a link. It sees the connection across the tunnel and draws a unified map as though no devices exist between the corporate site and its remote branches.
- But in this case, the company does not have a VPN to each site and the ISP is not providing tunnels. Instead, the ISP is taking the traffic and based on routing information is forwarding the traffic across a serial link to the remote branch.
- Because Network Manager has no visibility into the ISP network and no SNMP, Telnet, or SSH access to those devices, Network Manager draws 64 separate maps that represent isolated islands of connectivity. RCA can occur only within each separate contiguous map.

How can you build a single cohesive map with proper links?

- Sometimes you can eliminate discovery islands by enabling more agents that can get connectivity information not in the original discovery (such as the **TraceRoute** agent).
- To connect isolated portions of the network when you lack information about intervening networks, build connections manually with the Topology Editor or with the **PresetLayer.stch**. This PresetLayer stitcher populates regular discovery tables with information that you configure.

The Topology Editor is well suited for manually creating a few devices or connections. However, when you have multiple devices or connections, the **PresetLayer** stitcher is a better choice. Also,

the Topology Editor is incapable of creating a device, interfaces for that device, and defining containment relationships. For that level of detail, you must use the **PresetLayer** stitcher.

PresetLayer.stch introduction

- Can be used to manually create one or more of the following items:
 - Devices
 - Interfaces
 - Containment relationships
 - Connections between interfaces
- The connections can be between:
 - Discovered interfaces
 - Manually instantiated interfaces
 - Combination of discovered and manually instantiated interfaces
- This stitcher is called from the **BuildLayers** stitcher
 - Must uncomment to activate
- When you need to manually build many devices, interfaces, or links, the **PresetLayer.stch** is the ideal choice

```
//ExecuteStitcher('PresetLayer');
```

Must uncomment this line in **BuildLayers.stch** to
run the **PresetLayer** stitcher

Figure 10-72. PresetLayer.stch introduction

Manually instantiating devices

- You can instantiate devices that you want to represent in a network map but cannot discover such as:
 - Customer device that is connected to your provider edge (PE) device
 - Firewall to which you have no access
- You can create an icon to represent an intervening network to which you do not have access, such as an ISP

```

insert into workingEntities.finalEntity
(
  m_Name,
  m_Creator,
  m_ObjectID,
  m_Description,
  m_UniqueAddress,
  m_HaveAccess,
  m_EntityType,
  m_BaseName
)
values
(
  'deviceA',
  'Preset',
  '1.3.6.1.4.1.2342.23.34',
  'My fake device A',
  '1.1.1.1',
  1,
  eval(int,'$BaseEntity'),
  'deviceA'
);

```

Figure 10-73. Manually instantiating devices

Manually instantiating interfaces

- Create interfaces for devices that have no discovered interfaces because of a lack of SNMP support
- Create interfaces for devices that were manually created with the **PresetLayer** stitcher
- Use the **ObjectId** of a real device or one from an AOC that you want to use for this device
- Include information about the device in the interface description

```
insert into workingEntitiesfinalEntity
(
    m_Name,
    m_Creator,
    m_ObjectId,
    m_Description,
    m_UniqueAddress,
    m_HaveAccess,
    m_EntityType,
    m_BaseName,
    m_LocalNbr
)
values
(
    'deviceA[ 0 [ 1 ] ]',
    'Preset',
    '1.3.6.1.4.1.2342.23.34',
    'My fake device A interface 1',
    '1.1.1.1',
    1,
    eval(int, '$LocalNeighbor'),
    'deviceA',
    { m_IfIndex = 1 }
);
```

© Copyright IBM Corporation 2017

Customizing discovery

Figure 10-74. Manually instantiating interfaces

Manually instantiate containment relationships

- For correct topology representation, create containment relationships
 - Determines what interfaces are contained in what chassis
 - Necessary information to correctly draw links between chassis
- You need only to create containment for those interfaces that were not discovered normally

```
insert into workingEntities.containment
(
  m_Container,
  m_Part
)
values
(
  'deviceA',
  'deviceA[ 0 [ 1 ] ]'
);

insert into workingEntities.containment
(
  m_Container,
  m_Part
)
values
(
  'deviceB',
  'deviceB[ 0 [ 1 ] ]'
);
```

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-75. Manually instantiate containment relationships

Building connections between interfaces

- Build connections between remote neighbors:

```
ExecuteOQL(
    "
        insert into PresetLayer.entityByNeighbor
        ( m_Name, m_NbrName )
        values
        ( 'deviceA[ 0 [ 1 ] ]' , 'deviceB[ 0 [ 1 ] ]' );
    "
);
```

- Interface names must match exactly the **EntityName** for the interfaces in the topology database

```
sbk-pe12-jr2320.core.eu.test.lab[vlan.55 ]
sbk-pe12-jr2320.core.eu.test.lab[lo0 ]
sbk-pe12-jr2320.core.eu.test.lab[tap ]
```

- In **BuildLayers.stch**, uncomment the following line:

```
ExecuteStitcher('PresetLayer');
```

Figure 10-76. Building connections between interfaces

Topic summary

After you complete this topic, you can use the **PresetLayer** stitcher to create:

- Chassis entities
- Interface entities
- Containment relationships
- Connections between devices

Figure 10-77. Topic summary

Exercise: Customizing discovery

Customizing discovery

© Copyright IBM Corporation
2017

Figure 10-78. Exercise: Customizing discovery

Complete the exercise for this unit.

Review questions

1. List four ways to start a stitcher.
2. What discovery table contains a list of all IP addresses on the network?
3. What can you use to add many connections to a network other than the Topology Editor?
4. What stitcher command can you use to define a query that you can use multiple times in an optimized format?

Customizing discovery

© Copyright IBM Corporation 2017

Figure 10-79. Review questions

Write your answers here:

Unit summary

- Explain the final phase of discovery
- Implement a custom stitcher that uses pattern matching on device host names to add business information to discovery
- Create custom tables
- Customize the structure browser view with custom data
- Create network partition views based on custom data
- Configure the MODEL service to populate the new table
- Create and delete custom database tables

[Customizing discovery](#)

© Copyright IBM Corporation 2017

Figure 10-80. Unit summary

Review answers

1. List four ways to start a stitcher.
 - Call from another stitcher or when another stitcher completes
 - Start on a timed interval
 - Start when data is inserted into a table
 - Start when a user inserts a stitcher name into the **stitchers.actions** table
2. The **translations.ipToBaseName** table contains a list of all IP addresses in the discovered network.
3. The **PresetLayer** stitcher can be used to create devices, interfaces, and connections.
4. Use the **PrepareSQL** command to prepare an optimized query that you can call multiple times.

Figure 10-81. Review answers

Unit 11. The gateway and RCA

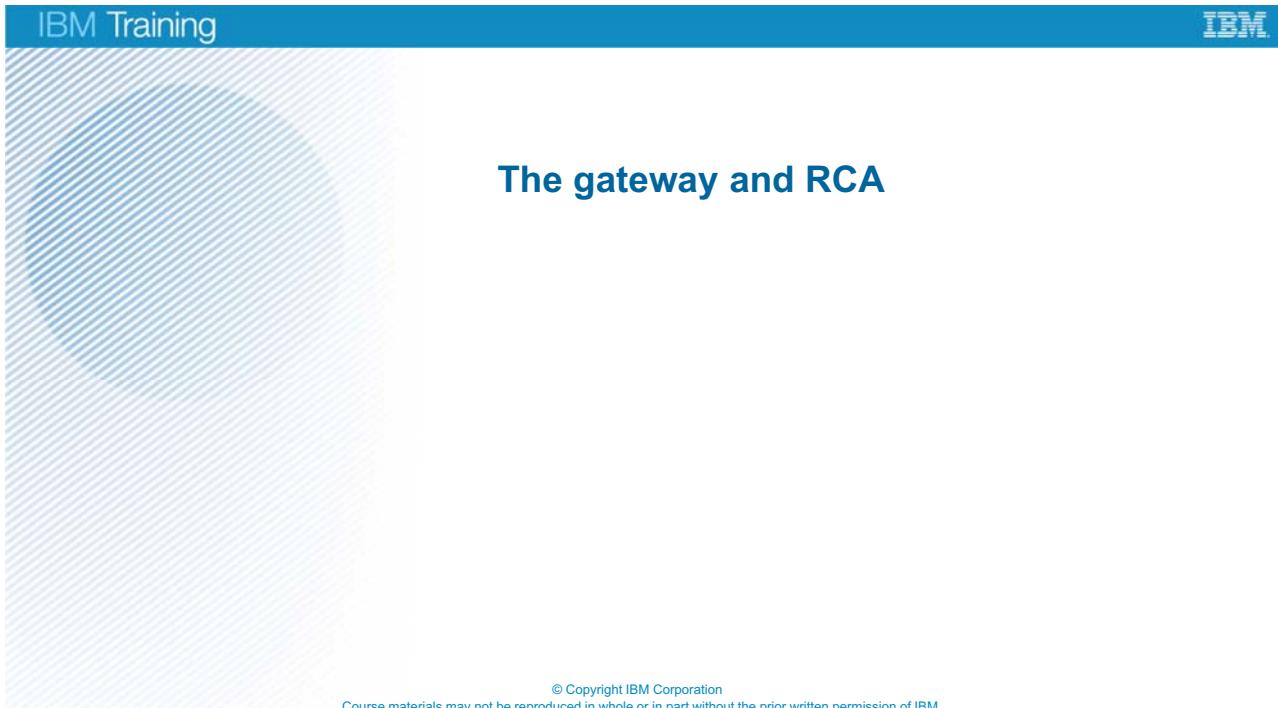


Figure 11-1. The gateway and RCA

Estimated time

01:00

Overview

In this unit, students learn how to enable or disable gateway plug-ins. Students also customize the gateway to enrich event information.

How you will check your progress

- Complete review questions
- Complete lab exercises

Unit objectives

- Disable and enable Tivoli Network Manager gateway plug-ins
- Enrich events with topology-related information

The gateway and RCA

© Copyright IBM Corporation

Figure 11-2. Unit objectives

11.1. Architecture and basic concepts

This unit introduces the architecture of the Tivoli Network Manager gateway and its plug-ins. Understanding of this architecture is crucial if you want to modify gateway or plug-in behavior.

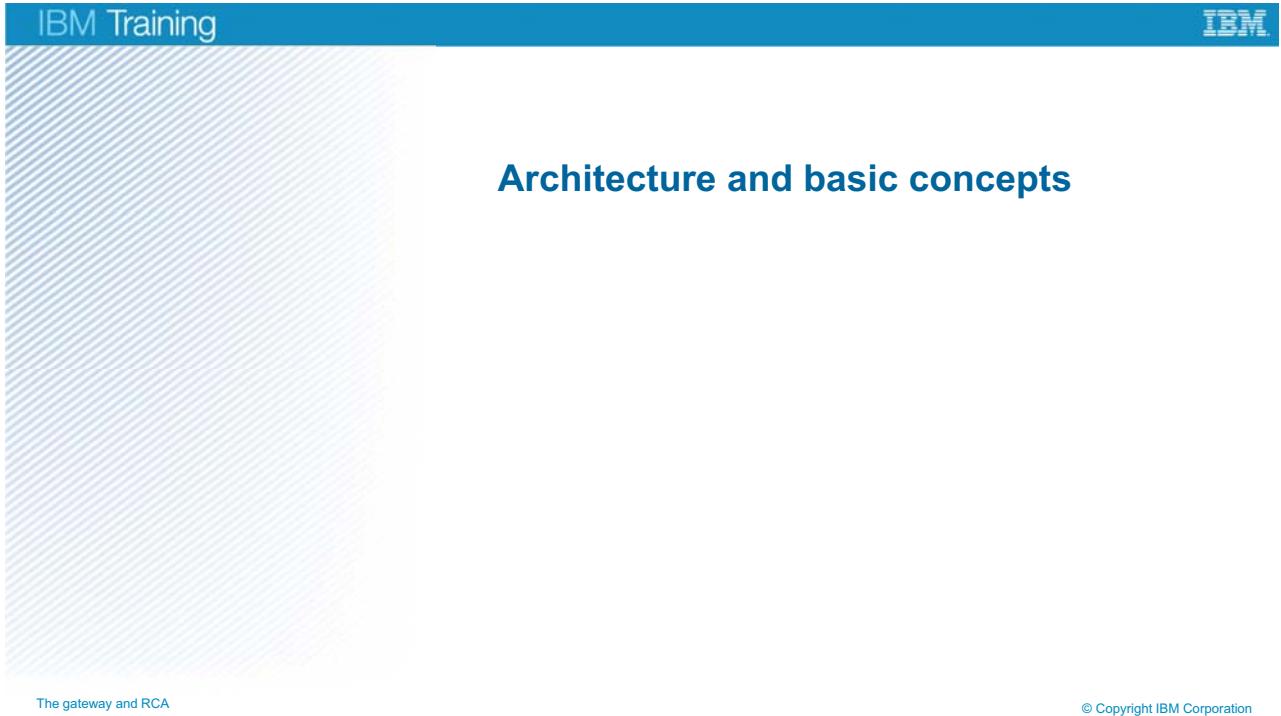


Figure 11-3. Architecture and basic concepts

Basic concepts (1 of 2)

Events in the ObjectServer come from two primary sources

- Tivoli Network Manager poller events already have entity-specific topology information included in the event data
- Events that are detected by OMNIBUS probes do not have topology information in the event data
 - The Tivoli Network Manager gateway must enrich these events

For proper visualization and gateway plug-in analysis, each event needs the following fields set:

- **NmosObjInst** is the ID number for the main node entity
 - Only the Tivoli Network Manager gateway populates this field
- **NmosEntityId** is the ID number for the lowest level entity that an event matches
 - Both the Tivoli Network Manager poller and gateway can populate this field

Figure 11-4. Basic concepts (1 of 2)

If an event is from the main IP address of a device, the **NmosEntityId** and **NmosObjInst** are the same.

If the event is from an interface on a device, the **NmosEntityId** points to the interface record while the **NmosObjInst** points to the chassis record. When you right-click an event and select a tool to show the topology, the tool includes the **NmosObjInst** in the URL so that the appropriate chassis appears in the GUI.

Basic concepts (2 of 2)

The Tivoli Network Manager gateway also enriches the **NmosManagedStatus** of the node in the alarm

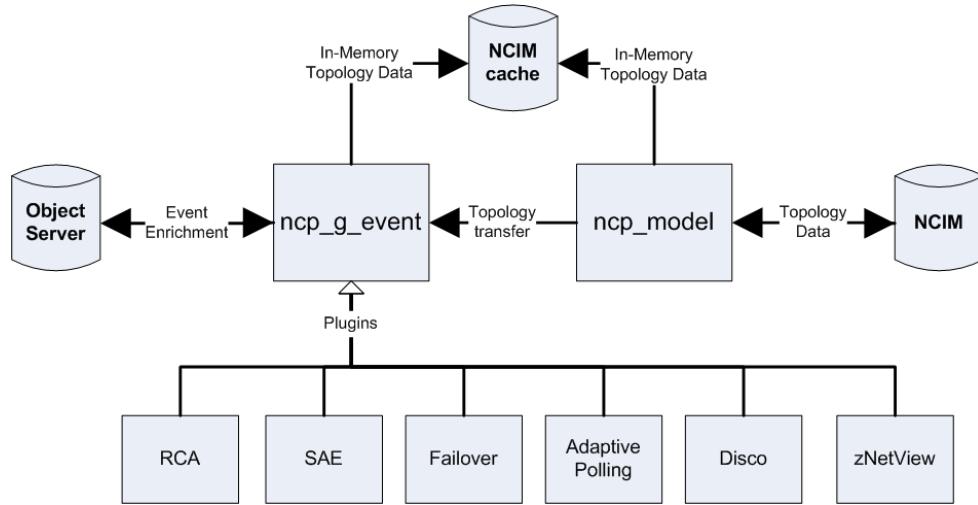
- NOC or help desk personnel can have filters that prevent viewing of devices that are currently in an unmanaged state

The gateway can also add other data from the topology to the event and assign it to an existing ObjectServer field:

- A device **sysLocation** can be put in to the ObjectServer **Location** field
- A device **sysContact** can be put in to a **Contact** field if it exists
- The interface description (**ifDescr**) can be added to the ObjectServer **Summary** field
 - This addition results in more meaningful interface events such as
Interface serial 0/1 down - T1 from Chicago to Little Rock

Figure 11-5. Basic concepts (2 of 2)

Architectural overview



The gateway and RCA

© Copyright IBM Corporation

Figure 11-6. Architectural overview

Architectural rationale

Tivoli Network Manager stores discovery data in two databases

- The **NCIM** database provides data to the topology visualization tool to render network maps
- The **NCIMCache** database contains a subset of **NCIM** data
 - This database has only the information that gateway plug-ins require for event enrichment and root cause analysis

The event gateway matches each event to a discovered topology entity and passes event information on to interested gateway plug-ins

- All events that correspond to a discovered entity are enriched with topology information
 - The stitcher language provides greater flexibility
- Gateway plug-ins use the same syntax as the discovery stitchers but also contain more gateway-specific stitcher rules

[The gateway and RCA](#)

© Copyright IBM Corporation

Figure 11-7. Architectural rationale

Architectural details

- Only two Tivoli Network Manager processes connect directly to the Object Server
 - Event gateway
 - The **nco_p_ncpmonitor** probe
- Gateway stitchers are in **\$ITNMHOME/eventGateway/stitchers/**
- You can enable, disable, or configure plug-ins to have a registered interest in certain types of events

The gateway and RCA

© Copyright IBM Corporation

Figure 11-8. Architectural details

Plug-in summary

- **RCA:** Root cause analysis
- **SAE:** Service-affecting events
- **Adaptive polling:** Populates the **activeEvent** table
- **Failover:** Used in Tivoli Network Manager failover
- **zNetView:** Populates custom **zNetView** alert fields
- **Disco:** Can trigger partial rediscoveries

The gateway and RCA

© Copyright IBM Corporation

Figure 11-9. Plug-in summary

Event processing

The gateway works on events in the ObjectServer **alerts.status** table and applies a filter to them

- The gateway resynchronizes with the ObjectServer at initialization, or after Tivoli Network Manager failover or fallback
- The gateway gets new events, notification of deleted problems, and problem events with updated fields every 5 seconds (default setting)

Each event gets assigned to an event map by a probe rules file

- The event map determines how an event is handled
- In previous versions of Tivoli Network Manager, the **NcoGateInserts.cfg** file assigned these values
- In Tivoli Network Manager 4.2, event maps are assigned by rules files in the Netcool Knowledge Library (NcKL)
 - The **NcoGateInserts.cfg** file is used only if you are using a version of NcKL before version 4.4.3

[The gateway and RCA](#)

© Copyright IBM Corporation

Figure 11-10. Event processing

The gateway works on events in the ObjectServer **alerts.status** table. The gateway applies a filter to those events. The **LocalNodeAlias** field of an event must have an assigned value in order for the gateway to enrich or analyze an event. This default **EventFilter** in **EventGatewaySchema** specifies this requirement. When Tivoli Network Manager failover is enabled, the standby domain uses the **StandByEventFilter**.

The gateway resynchronizes with the ObjectServer, retrieving and filtering all events, when one of the following events happens:

- Gateway is initialized (start)
- Reinitialized (through SIGHUP)
- Tivoli Network Manager failover or fallback

The gateway receives the following events:

- New events and notification of deleted problem events by periodic retrieval from the ObjectServer
- All problem events that had any field that was updated

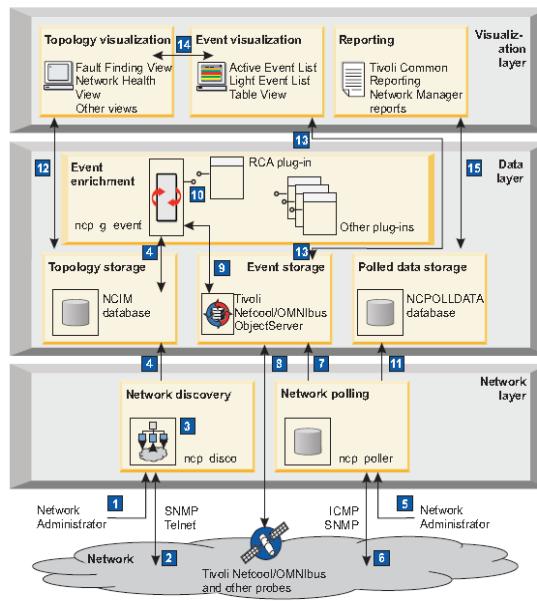
Event maps determine how an event is handled. These maps attempt to match the event to an entity in the topology, and pass the event to interested plug-ins such as RCA.

Event maps are selected in two ways:

- The **config.precedence** entries look at the **EventId** field of the **alerts.status** event.
- With the new **NmosEventMap** field of the **alerts.status** event.

Topology lookups and event enrichment are performed with the stitcher that is named in the **eventMap**. Some events, such as Tivoli Network Manager health check events, do not correspond to topology entities. These events are not enriched.

Tivoli Network Manager data flow



© Copyright IBM Corporation

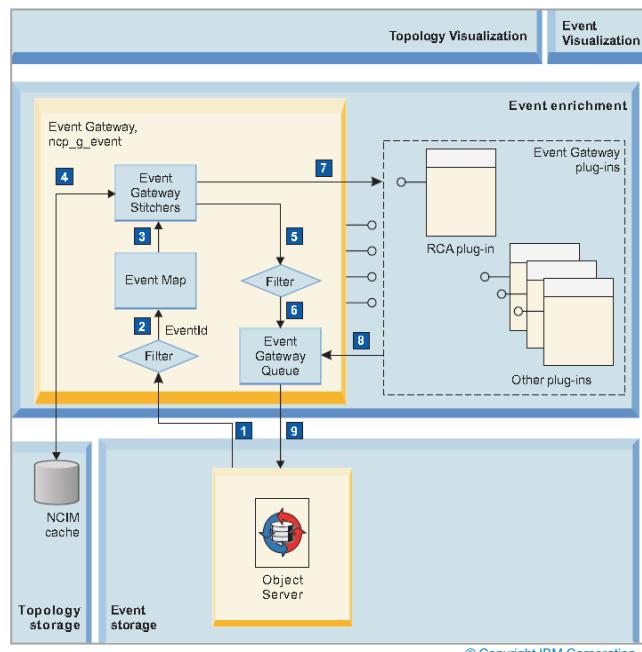
Figure 11-11. Tivoli Network Manager data flow

1. The administrator configures and starts a network discovery.
2. The **ncp_disco** process (or DISCO service) finds and interrogates network devices.
3. Discovery information is sent to the MODEL service (**ncp_model**).
4. The MODEL service pushes data to the NCIM database and to the NCIM cache.
5. The administrator configures network polling.
6. The POLLER service (**ncp_poller**) polls network entities according to poll definitions and policies.
7. Threshold violations from polls are sent to the ObjectServer.
8. Tivoli Netcool/OMNIbus probes also send event data to the ObjectServer.
9. Events are passed to the event gateway, **ncp_g_event**, where they are enriched with topology data. Some events are passed directly back to the ObjectServer. Event gateway plug-ins subscribe to certain types of events. Based on these subscriptions, events are passed to the event gateway plug-ins.
10. Event gateway plug-ins perform various actions that are based on the events that are received from the ObjectServer. For example, the RCA plug-in does further event enrichment. The SAE plug-in generates synthetic service-affecting events (SAE) that are based on events that are received. Other plug-ins take other actions that are based on the occurrence of certain events. For example, the Disco plug-in starts partial discovery on devices that have alarms in which it is interested. The Failover plug-in initiates failover that is based on the occurrence of Network Manager health check events. Plug-ins pass enriched events back to the ObjectServer.

11. Network Manager gathers device performance data on demand. A network administrator can set up polling of specific SNMP and ICMP data on one or more network devices at any time. Network Manager gathers this data and stores it in the **NCPOLLDATA** historical database.
12. Topology visualization software accesses the NCIM database. The topology visualization web application, running within the Tivoli Integrated Portal application, accesses the topology within the NCIM database. Network operators can now log on to the Tivoli Integrated Portal server. They can view network devices and components with the Network Manager topology visualization GUIs. The visualization options include both multi-portlet and single portlet views:
 - Multi-portlet views include the fault-finding view and the network health view.
 - Single-portlet views include the hop view, network views, and the structure browser.
13. Event visualization software accesses the ObjectServer. The Tivoli Netcool/OMNibus Web GUI requests the most recent set of events from the ObjectServer. Changes that operators make to these events with the Web GUI are sent back to the ObjectServer. Network operators can log on to the Tivoli Integrated Portal and view events with the Active Event List (read/write) or an Event Viewer (read-only).
14. Event information is requested. The Topology Visualization web application requests event information from the Tivoli Netcool/OMNibus Web GUI application.
15. Report data for performance reports is retrieved from the NCPOOLDATA historical polled data database. Network operators log on to the Tivoli Integrated Portal, access Tivoli Common Reporting, and run performance data and other reports. The report data for performance reports is retrieved from the **NCPOOLDATA** historical polled data database.

Event enrichment data flow

1. Gateway receives event from ObjectServer
2. Gateway selects event map
3. Event map triggers topology lookup stitcher
4. Gateway looks up topology
5. Gateway applies filter to determine whether to pass the event to the ObjectServer
6. Enriched fields are placed on the event gateway
7. Gateway passes event to interested plug-ins
8. Plug-ins update event in gateway
9. Event is returned to ObjectServer



The gateway and RCA

© Copyright IBM Corporation

Figure 11-12. Event enrichment data flow

Selecting an entity for processing

The gateway matches events to topology entities

- Event maps for interface events run a stitcher that looks up the main node IP of the device, and then it populates the correct **NmosObjInst** integer value for that node
- The **EventGatewaySchema.cfg** file associates event maps with stitchers
 - If a stitcher is specified for a certain type of event, that stitcher immediately processes the event data
- OMNIbus fields that are available to the gateway stitchers are listed in the **nco2ncp FieldFilter** in the **EventGatewaySchema.cfg** file

The gateway checks each event as it arrives to see whether the device in the alarm exists in the discovered topology

- If the device does not exist, no further processing is done on the event
- If the device does exist, the gateway first enriches the event (**NmosObjInst**, **NmosEntityId**, **NmosDomainName**, and **NmosManagedStatus**) and then it runs the specified gateway stitcher for that type of event

Figure 11-13. Selecting an entity for processing

Event enrichment

- The `GwEnrichEvent()` stitcher rule is used to add fields to an event
 - You can aggregate multiple fields into one temporary table and then update all the fields in the ObjectServer with one statement
 - Only the fields that are listed in the `ncp2nco FieldFilter` in the `EventGatewaySchema` file are passed to the Object Server
 - See `StandardEventEnrichment.stch` for examples
- All enriched fields are also passed to subsequent stitchers and plug-ins
- By default, ObjectServer updates occur asynchronously at 5-second intervals
 - You can configure this behavior by changing the `ObjectServerUpdateInterval` in the `EventGatewaySchema.cfg` file (in the `$NCHOME/etc/precision/` directory)

Figure 11-14. Event enrichment

Event processing plug-ins

- Plug-ins run simultaneously but independently of one another
 - Therefore, two plug-ins can operate on the same event data at the same time
- Individual plug-ins are interested only in certain types of events as defined in the event maps
- You can use the **ncp_gwplugins.pl** script to cause a plug-in to have a registered interest in certain types of events and to view the status of each plug-in
- Configure plug-ins with care:
 - Plug-in configuration is stored in SQL
 - The plug-in architecture makes it easier to enhance future releases of the product

Figure 11-15. Event processing plug-ins

Custom plug-in development (1 of 3)

Overview:

- Development of plug-ins is a task for IBM Tivoli development
 - New stitcher development or modification of core stitchers is not undertaken without fully understanding the potential impact
- A typical customer customization would be adding event enrichment data

The core gateway stitchers are intended for top-level topology lookups

- Match an event to an entity and enrich the event with topology information
- Start specific plug-ins to handle certain events

Figure 11-16. Custom plug-in development (1 of 3)

Custom plug-in development (2 of 3)

- Gateway plug-in structure makes it easier for Tivoli development to implement new plug-ins in future development
 - Plug-ins are independent of each other
 - Creating and implementing custom plug-ins requires a full understanding of the potential impact on how events are handled
- **Disco** and **zNetView** plug-in configuration can be used as templates:
 - Entries in the **ncmonitor gwPluginTypes** and **gwPlugins** tables determine which plug-ins are available
 - Entries in **gwPluginStates** and **gwPluginEventMaps** determine which sets of events are passed to a plug-in

Figure 11-17. Custom plug-in development (2 of 3)

Custom plug-in development (3 of 3)

- Stitcher plug-ins currently accept three variables in the **gwPluginConf** table:
 - **StitcherSubDir**: Specifies the subdirectory of `$ITNMHOME/eventGateway` that contains stitchers for this plug-in
 - **SchemaFile**: An optional schema file to read in at start that creates a table local to that plug-in
 - **StartupStitcher**: An optional stitcher to run at Initialization (start, HUP, failover, and fallback)
- Verify the configuration with the `ncp_gwplugins.pl` script

The gateway and RCA

© Copyright IBM Corporation

Figure 11-18. Custom plug-in development (3 of 3)

The **ncp_gwplugins.pl** script enables or disables plug-ins, registers plug-in interest in certain types of events, and provides plug-in status.

Custom plug-in development is a task for IBM Tivoli software developers.

Access to ObjectServer tables (1 of 2)

ObjectServer tables can be accessed from the stitchers:

- Only the Tivoli Network Manager gateway process and the **ncp_oql** binary can access ObjectServer databases
 - The discovery engine cannot access these databases
- When your write stitchers to access the ObjectServer database, use the existing **PrepareSQL()** and **ExecuteSQL()** rules
 - See the example in **RetrieveAlertDetails.stch**

When the gateway updates the ObjectServer, it is a one-way update

- Updated fields go into the ObjectServer
- The normal gateway synchronization to the ObjectServer does not retrieve the fields that the gateway modified

Figure 11-19. Access to ObjectServer tables (1 of 2)

Access to ObjectServer tables (2 of 2)

- If you need to use ObjectServer data for troubleshooting, you can access this capability from the command line on the Tivoli Network Manager server
 - The **ncp_oql** binary can access Object Server data
 - Use **-service ObjectServer** on the command line
 - You can use the **-tabular** command-line argument or use **tabon** or **taboff** on the OQL command line to change the format of the returned data
 - To log on to the ObjectServer, you need to supply a valid user name and password
- ```
ncp_oql -domain NCMS -service ObjectServer -username root
```

Figure 11-20. Access to ObjectServer tables (2 of 2)

## Custom enrichment

In Tivoli Network Manager 4.2, you can customize the enrichment of the ObjectServer event

Edit these two files:

- `$/opt/IBM/tivoli/netcool/precision/eventGateway/stitchers/StandardEventEnrichment.stch`
  - Declare the new field
  - Set a value in the `@enrichedFields` recorder
  - The `GwEnrichEvent` function sends data to the gateway for enrichment according to the fields in `$NCHOME/etc/precision/EventGatewaySchema.cfg`
- **The `$NCHOME/etc/precision/EventGatewaySchema.cfg` file specifies which fields are sent to the ObjectServer in the `ncp2nco` section of the file**

Figure 11-21. Custom enrichment

## Gateway stitcher rule summary

- Enriching events
  - `GwEnrichEvent(serial number, enrichedFields)`
- General entity-related functions
  - `GwIpLookup(IP address or DNS name)`
  - `GwIpLookupUsing(event field name)`
  - `GwMainNodeLookup(identifier)`
  - `GwMainNodeLookupUsing(event field name)`
  - `GwManagedStatus(integer entity ID)`
- Retrieving NCIM cache data directly
  - `GwCollects(entityId or entityName)`
  - `GwConnects(entityId or entityName, [optional topology name])`
  - `GwContains(entityId or entityName)`
  - `GwDependency(entityId or entityName, [optional dependency identifier])`
  - `GwEntityData(entity ID or entity name)`
  - `GwHostedService(entityId or entityName)`
  - `GwPipeComposition(entityId or entityName)`
  - `GwProtocolEndPoint(entityId or entityName)`

The gateway and RCA

© Copyright IBM Corporation

Figure 11-22. Gateway stitcher rule summary

Two of these rules are used frequently when you enrich OMNIbus events.

- **GwEnrichEvent** sends data back to the ObjectServer to enrich the event.
- **GwMainNodeLookup** looks up the main node **entityId** for an interface event. This lookup enables Tivoli Network Manager to correctly color the appropriate icon on the topology views and account for the containment relationships during root cause analysis.

## 11.2. RCA overview

Root cause analysis (RCA) is the common term for network fault isolation. The 3.x versions of Tivoli Network Manager used a proprietary language for RCA code, which required great expertise to be modified. In Tivoli Network Manager 4.2, the RCA plug-in uses the same stitcher language as discovery stitchers. However, several stitcher keywords are unique to the RCA process.



Figure 11-23. RCA overview

## RCA overview

- RCA rules use stitcher language
- RCA stitchers are in `$ITNMHOME/eventGateway/stitchers/RCA` directory
- Configuration file: `$NCHOME/etc/precision/RCASchema.cfg`
  - Creates **mojo.events** table that contains events that plug-ins are processing
  - Contains variable that controls how the RCA plug-in uses the managed status of a device
  - Contains variable that controls whether RCA plug-in uses time correlation

The gateway and RCA

© Copyright IBM Corporation

Figure 11-24. RCA overview

Modify RCA rules only if you fully understand the implications of those changes.

## RCASchema.cfg

- The Tivoli Network Manager 4.2 **mojo.events** table includes these fields:

EntityName, ClassName, EventName, RuleSet, Contact, AssignedTo, Acknowledged, Location, CorrelatedId, EventGroupId, ActionGlyph, Occurred, ActionType, AgentAddress, ExtraInfo, AlertType, RuleName

- The **config.defaults** file has two fields that alter the way root cause analysis works:
  - MaxAgeDifference**
  - HonourManagedStatus**

Figure 11-25. RCASchema.cfg

## MaxAgeDifference

- In previous versions of Tivoli Network Manager, it was possible for recent events to designate events that were several days older as symptom or root cause events
- In IBM Tivoli Network Manager 4.2, you can customize `RCASchema.cfg` to set the `MaxAgeDifference` parameter
  - This parameter controls the maximum number of minutes between two events in order for the RCA engine to allow the suppression of either event

```
insert into config.defaults
(
 MaxAgeDifference,
 HonourManagedStatus
)
values
(
 0,
 1
);
```

- When `MaxAgeDifference` is set to zero, the time values of the events are ignored
- A nonzero value represents the maximum number of minutes between events for RCA engine to potentially suppress either event

[The gateway and RCA](#)

© Copyright IBM Corporation

Figure 11-26. *MaxAgeDifference*

Setting the **MaxAgeDifference** value ensures that Tivoli Network Manager correlates only events that occur in close time proximity to one another. Set the value to the number of minutes between two events in order for the RCA engine to potentially suppress either event. This setting is not the default value.

To ensure that Tivoli Network Manager does root cause analysis only on devices that are actively managed, set **HonourManagedStatus** to **1**. This setting is the default value. To force Tivoli Network Manager to do root cause analysis on all events that are associated with network devices, set **HonourManagedStatus** to **0**.

## HonourManagedStatus

This value is also set in **RCASchema.cfg**

- A value of 0 tells Tivoli Network Manager 4.2 to ignore the **ManagedStatus** of a device and always attempt to do network fault isolation (RCA)
- A value of 1 (the default setting) tells Tivoli Network Manager 4.2 to recognize the **ManagedStatus** of a device and:
  - Perform network fault isolation (RCA) on those devices that are managed
  - Ignore unmanaged devices and interfaces for purposes of RCA

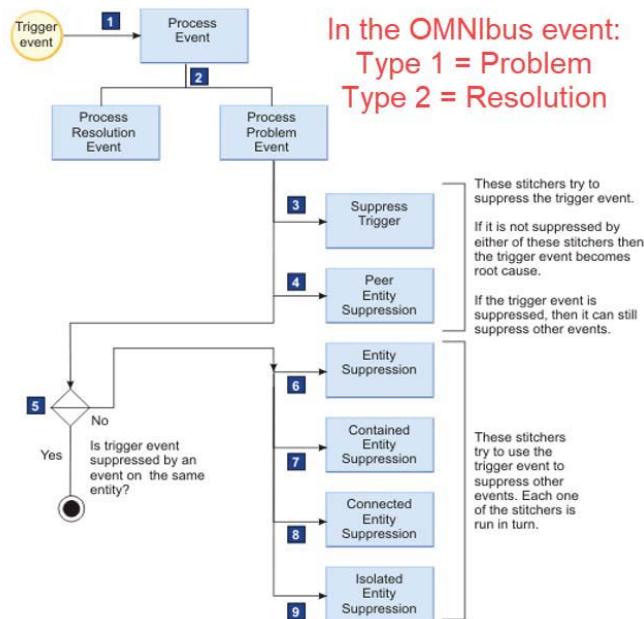
```
insert into config.defaults
(MaxAgeDifference,
 HonourManagedStatus
)
values
(0,
 1
);
```

[The gateway and RCA](#)

© Copyright IBM Corporation

*Figure 11-27. HonourManagedStatus*

## RCA process



The gateway and RCA

© Copyright IBM Corporation

Figure 11-28. RCA process

The RCA process can be described in three broad general steps:

1. The event gateway receives an event. The gateway RCA plug-in checks whether this event matches specific event maps in which the plug-in has a subscribed interest. Events that cause the RCA plug-in to do analysis are called ***trigger events***.
2. The gateway inserts trigger events into the **mojo.events** database. The RCA stitchers process the events in this table.
3. The gateway starts the **ProcessEvent.stch** stitcher (block #1 in the preceding diagram) that begins the root cause analysis on the event.

## Precedence values in NcKL rules files, formerly in NcoGateInserts.cfg file

- The Netcool Knowledge Library (NcKL) 4.4.3 or later rules file for processing SNMP traps assigns a precedence value for certain types of events
- When knowledge of network topology is inadequate to firmly determine root cause, the precedence values assign greater weight to some events than to others in calculating root cause
- Precedence values aid in diagnosis when multiple poll events or traps indicate problems in the network
  - **EventIds** with a precedence of 0 cannot be root cause events; other events can suppress these events
  - **EventIds** with a precedence of 10,000 cannot become suppressed events

The gateway and RCA

© Copyright IBM Corporation

Figure 11-29. Precedence values in NcKL rules files, formerly in NcoGateInserts.cfg file

Tivoli Network Manager uses topology data first to determine root cause. When that data is insufficient to determine root cause, the RCA engine evaluates the precedence value in the `NcoGateInserts.cfg` file. Some events have a precedence value that prevents them from becoming either a root cause event or a suppressed event.

## 11.3. Gateway plug-ins

You can configure the behavior of gateway plug-ins or view their configuration with one command. You can also configure a plug-in to have a registered interest in certain types of events. In this section, you learn how to configure the Disco plug-in to automatically rediscover devices when certain events occur.

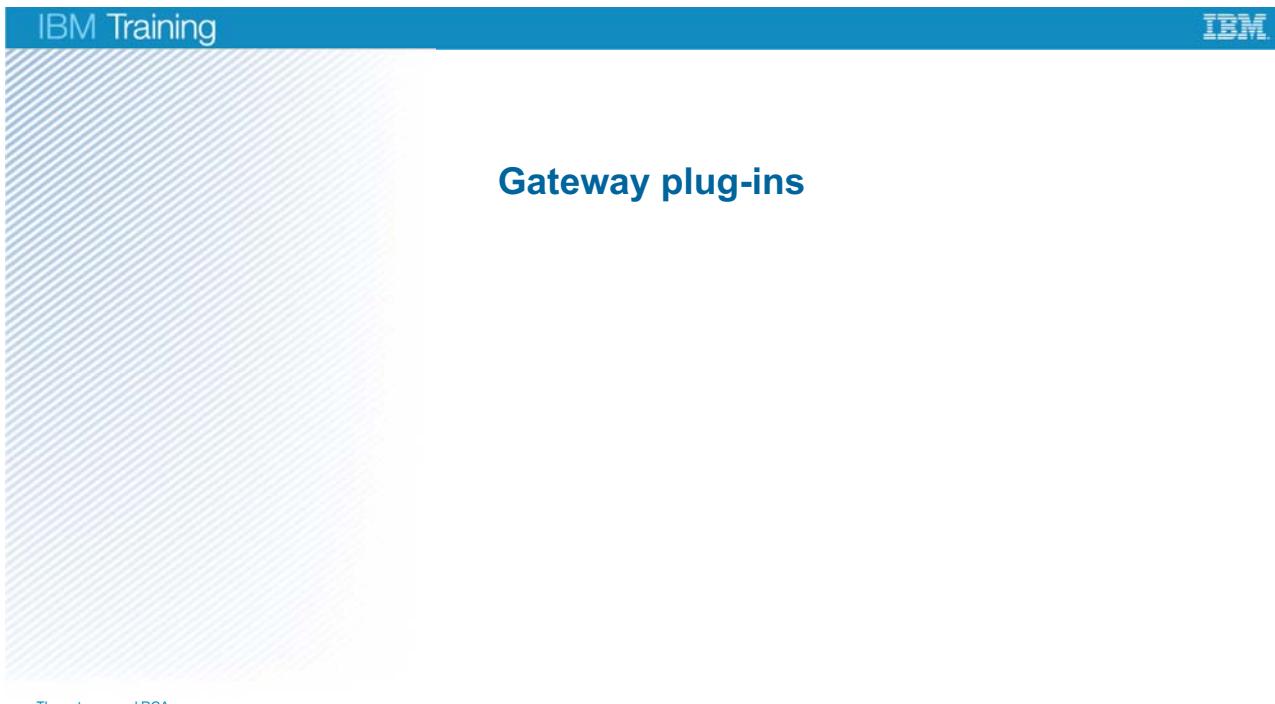


Figure 11-30. *Gateway plug-ins*

The pages in this section are provided primarily for support personnel. Your instructor might skip ahead to the slide entitled “Determining which plug-ins are enabled.”

## New fields in mojo.events

- Now in alphabetical order
- **FirstOccurrence** and **LastOccurrence** fields from OMNIbus
- Other fields are from the Tivoli Network Manager gateway

```
create table mojo.events
(FirstOccurrence time not null,
 IsIsolationPoint int not null,
 IsLoopbackInterface int not null,
 IsOrphan int not null,
 IsMasterEvent int not null,
 LastOccurrence time not null,
 SuppressionState int not null,
 SuppressionTime time not null,
) ;
```

The gateway and RCA

© Copyright IBM Corporation

Figure 11-31. New fields in mojo.events

## RCA stitcher processing order



[The gateway and RCA](#)

© Copyright IBM Corporation

Figure 11-32. RCA stitcher processing order

Two other stitchers complete the core stitchers in Tivoli Network Manager 4.2:

- **ProcessResolutionEvent.stch:** Handles resolution events for events that are previously determined to be either symptom or root cause events
- **TimedEventSuppression.stch**

The RCA plug-in always prefers domain-specific versions of these files over non-domain-specific files. It uses `ProcessProblemEvent.ITNM4GO.stch` in the **ITNM4GO** domain instead of `ProcessProblemEvent.stch`.

## RCA stitchers

### **ProcessEvent.stch**

- This stitcher is the head stitcher
- The RCA plug-in calls this stitcher each time a trigger event is passed to it
- It calls:
  - **ProcessProblemEvent.stch** for **Type=1** events
  - **ProcessResolutionEvent.stch** for **Type=2** events
- It has no parameters
  - The in-scope record (at level 0) is the trigger event record

### **ProcessProblemEvent.stch**

- It processes occurred, recurred, and reawakened events

### **ProcessResolutionEvent.stch**

- It processes cleared events and deleted events

The gateway and RCA

© Copyright IBM Corporation

Figure 11-33. RCA stitchers

## Timed RCA rule

- **TimedEventSuppression.stch**
  - This stitcher runs as a timed trigger
  - It runs every 30 seconds
- When it runs, it processes events with the following characteristics:
  - **TimedEscalation=1**
  - A time stamp (**CreateTime**) of at least 30 seconds ago
- After this stitcher runs and processes an event, it sets **TimedEscalation=2** to mark this event as done
  - The gateway marks events that can flap with a flag of **EventCanFlap=1**
  - If **EventCanFlap=1**, then **TimedEscalation** is set to 1 in the RCA engine

The gateway and RCA

© Copyright IBM Corporation

Figure 11-34. Timed RCA rule

## Customizing plug-ins

For customizations, edit `ProcessProblemEvent.stch` and call a custom stitcher near the end of the file

Precautions and consequences:

- Before you modify any RCA rule, back up the existing rules first
  - Create a domain-specific copy of any stitcher you modify
  - Leave the original stitcher in its original state
- Restart `ncp_g_event` to get it to recognize the new rule
- Parse errors are logged in `ncp_g_event.log` with stitcher line numbers
- Parse errors prevent the RCA plug-in from loading but do not otherwise prevent the process `ncp_g_event` from running

The gateway and RCA

© Copyright IBM Corporation

Figure 11-35. Customizing plug-ins



### Note

Tivoli Network Manager reprocesses the suppressed status of cleared or deleted events automatically. You do not need to modify `ProcessResolutionEvent.stch` to do anything extra for cleared events.

## Example modification

- Simple modifications can be made in the existing stitcher
- **ProcessProblemEvent.stch**

```
// Example (1): restrict contained suppression such
// that only cards/modules can contain suppressed events
// on contained entities

int entityType = eval(int, '&EntityType');
if (entityType == 5)
{
 numberOfSuppressedEvents = ExecuteStitcher('ContainedEntitySuppression');
}
```

The gateway and RCA

© Copyright IBM Corporation

*Figure 11-36. Example modification*

## 1+1=2 Example

### ProcessProblemEvent.stch

```
// Example (1): restrict contained suppression such
// that only cards/modules can contain suppress events
// on contained entities
int entityType = eval(int, '&EntityType');
if (entityType == 5)
{
 numberOfSuppressedEvents = ExecuteStitcher('ContainedEntitySuppression');
}
```

The preceding slide shows a modification to **ProcessProblemEvent.stch**. This modification restricts contained entity suppression to cards and modules.

## Calling a stitcher from ProcessProblemEvent.stch

- Business case: Customer wants events on certain crucial interfaces to always be treated as root cause events
- Created custom `CrucialInterfaceEventHandler.stch` and called from `ProcessProblemEvent.stch`
  - See example of code in your Course Guide

```

...
}
} // end if precedence > 0
// Example (2): call customized rules after completion of the default rules
ExecuteStitcher('CrucialInterfaceEventHandler');
success = 1;
SetReturnValue(success);
}
}
}

```

The gateway and RCA

© Copyright IBM Corporation

Figure 11-37. Calling a stitcher from `ProcessProblemEvent.stch`

In this example, a customer required a new RCA rule for a custom solution. The customer has devices with a special crucial interface on them. Faults on these interfaces need to be identified as the root cause of any other event on the same device.

You can write a stitcher, `CrucialInterfaceEventHandler.stch`, to ensure that events on crucial interfaces are always designated as the root cause event on that main node. These root cause events override any conflicting RCA results that the core rules generated before the execution of `CrucialInterfaceEventHandler.stch`.

```

CrucialInterfaceEventHandler.stch
// This stitcher looks for a particular type of
// interface event, with EventId =
// 'CrucialIfaceEvent', and, if found, marks it
// as the root cause event on a device, suppressing
// any other events (including chassis events) on
// that device. This is an example customization of
// the RCA rules that can be triggered from the
// ProcessProblemEvent stitcher
UserDefinedStitcher
{
 StitcherTrigger
 {
 // Called from the ProcessProblemEvent stitcher
 }
}

```

```

StitcherRules
{
text eventId = eval(text, '&EventId');
int serial = eval(int, '&Serial');
int mainNodeId = eval(int, '&NmosObjInst');
int suppressedSerial = 0;
if (eventId == 'CrucialIfaceEvent')
{
// Make the crucial in-scope event root cause.
AmosSetCause(eval(int, '$RCA_ROOT_CAUSE'));
// Look for any other events on the same device/main
// node...
RecordList suppressedEvents = RetrieveOQL(
"select Serial from mojo.events where NmosObjInst = eval(int, '$mainNodeId') and
Serial !=
eval(int, '$serial');"
);
foreach (suppressedEvents)
{
suppressedSerial = eval(int, '&Serial');
// ...and if found, mark each one as a symptom of the crucial interface
// event
AmosSetSymptom(suppressedSerial, serial, eval(int, '$RCA_CUSTOM_SUPPRESSION')
);
}
}
else
{
//
// If there were any crucial events on this chassis,
// they should suppress this event (currently
// assumes a single crucial event per chassis)
// Look for any existing CrucialIfaceEvent events on
// the same device/main node...
//
Record crucialSuppressed = RetrieveSingleOQL(
"select Serial from mojo.events where EventId = 'CrucialIfaceEvent' and
NmosObjInst = eval(int,
'$mainNodeId');"
);
if (crucialSuppressed <> NULL)
{
int crucialSerial = @crucialSuppressed.Serial;
// ...if found, make the crucial event the root
// cause (currently assumes a single crucial event
// per chassis)...
AmosSetCause(crucialSerial, eval(int, '$RCA_ROOT_CAUSE'));
}
}
}

```

```
// ...and mark the trigger event as a symptom of the
// crucial event
AmosSetSymptom(crucialSerial, eval(int, '$RCA_CUSTOM_SUPPRESSION'));
}
}
SetReturnValue(1);
}
}
```

---

**Note**

The Serial field in the ObjectServer **alerts.status** database table is a locked column because it is the primary key field in that table.

---

## More stitcher functions

These extra functions allow greater control over event suppression:

- **AmosRetrieveEntities**
- **AmosGetContainedEntities**
- **AmosGetConnectedEntities**
- **AmosGetIsolatedEntities**
- **AmosGetEvents**
- **AmosUpdateEvent**
- **AmosUpdateSuppressees**
- **AmosSetSymptom**
- **AmosEnrichEvent**

*Figure 11-38. More stitcher functions*

The following list describes more stitcher functions:

- **AmosRetrieveEntities:** This function retrieves entities that are contained or connected or isolated by the trigger entity. For example, it gets all entities that are contained by the trigger entity.
- **AmosGetContainedEntities:** This function retrieves all entities that are contained by the entity that has entityId = 'entityId'.

```
RecordList = AmosRetrieveEntities(SuppressionType)
int suppressionType = eval(int, '$RCA_CONTAINED_SUPPRESSION');
RecordList myEntities = NULL;
myEntities = AmosRetrieveEntities(SuppressionType);
RecordList = AmosGetContainedEntities(EntityId)
int entityId=0;
entityId = eval(int, '&NmosEntityId');
RecordList myEntities = NULL;
myEntities = AmosGetContainedEntities(entityId);
```

- **AmosGetConnectedEntities:** This function retrieves all entities that are connected to the entity that has entityId = 'entityId'.
- **AmosGetIsolatedEntities:** This function retrieves all entities that are isolated from the entity that has entityId = 'entityId'.

- **AmosGetEvents:** This function retrieves all events on the entity where entityId = 'entityId'.

```

RecordList = AmosGetConnectedEntities(EntityId, EntityType)
int entityId=0;
entityId = eval(int,'&NmosEntityId');
int entityType=0;
entityType = eval(int,'&EntityType');
RecordList myEntities = NULL;
myEntities = AmosGetConnectedEntities(entityId, entityType);
RecordList = AmosGetIsolatedEntities(EntityId, PollerEntityId)
int entityId=0;
entityId = eval(int,'&NmosEntityId');
int pollerEntityId=0;
pollerEntityId = eval(int,'&PollerEntityId');
RecordList myEntities = NULL;
myEntities = AmosGetIsolatedEntities(entityId, pollerEntityId);
RecordList = AmosGetEvents(EntityId)
RecordList myEvents = NULL;
myEvents = AmosGetEvents(myEntityId);

```

- **AmosUpdateEvent:** This function updates the event in the RCA table, **mojo.events**. It uses the serial number of the in-scope event at nest level 0. This event is the one referenced with a single ampersand (&). This function is used to locate and update the event in **mojo.events**, according to the supplied parameters. It returns an integer (treated as a Boolean value) indicating the success or failure of the update: 0=failure, 1=success. It updates only the actual record in **mojo.events**. It does not update the **inscope(0)** record itself. The in-scope record just goes out of scope at the end of the stitcher.

```

Boolean = AmosUpdateEvent("<Attribute Name>" , <Attribute Value>)
int success = 0;
success = AmosUpdateEvent("NmosCauseType" , 1);
update mojo.events set NmosCauseType=1 where Serial=<Serial>;
success = AmosUpdateEvent("NmosSerial" , "0");
success = AmosUpdateEvent("SuppressionState" , "0");
success = AmosUpdateEvent("SuppressionTime" , "0");
int success = 0
text myname = "NmosCauseType";
int myvalue = 2;
success = AmosUpdateEvent(myname , myvalue);

```

- **AmosUpdateSupressees:** This function updates events that the trigger record event causes. It flags these symptom events as orphans. It sets the field **Orphaned** to a value of 1 in each of the suppressed events. The orphan events are reprocessed when **AmosReprocessSupresses** is called later.
- **AmosSetSymptom:** This stitcher function specifies the relationship between a symptom event and its root cause event. It specifies the suppressor **Serial** and the suppression type. In the following example, the suppressed event with specified **Serial=12444** is updated to indicate

that the event with **Serial=12333** is the root cause event. The suppression type is set to \$RCA\_CUSTOM\_SUPPRESSION.

```
int suppressionType = eval(int, '$RCA_CUSTOM_SUPPRESSION');
success = AmosSetSymptom(12333, 12444, suppressionType)
Integer = AmosUpdateSuppressees("Attribute Name", Attribute Value)
int numberOfSuppressees = 0;
numberOfSuppressees = AmosUpdateSuppressees("Orphaned", 1);
Boolean = AmosSetSymptom([optional Serial], SuppressorSerial,
SuppressionType)
```

- **AmosEnrichEvent:** This function enriches a specified field of an event with a specified value.

## Optional stitcher functions (1 of 2)

### **AmosRetrieveEntities (suppressionType)**

- This function gets entities that have a relationship with the trigger entity
  - For example, it gets all entities that are contained by the trigger entity

### **AmosGetContainedEntities (entityId)**

- This function gets all entities that are contained by the entity that has `entityId = 'entityId'`

### **AmosGetConnectedEntities (entityId, entityType)**

- This function gets all entities that are connected to the entity that has `entityId = 'entityId'`

Figure 11-39. Optional stitcher functions (1 of 2)

## Optional stitcher functions (2 of 2)

- **AmosGetIsolatedEntities(entityId, pollerEntityId)**
  - This function looks at the event entity that has `entityId = 'entityId'` and retrieves a list of other network entities that the specified event entity isolates
  - These other network entities appear isolated from the perspective of the polling station entity where `entityId = 'pollerEntityId'`
- **AmosGetEvents(entityId)**
  - This function gets all events on the entity that has `entityId = 'entityId'`

Figure 11-40. Optional stitcher functions (2 of 2)

## Suppression types in RCA rules

These suppression types can be used in the RCA stitchers

| Suppression type            | Integer value |
|-----------------------------|---------------|
| \$RCA_NO_SUPPRESSION        | 0             |
| \$RCA_CUSTOM_SUPPRESSION    | 1             |
| \$RCA_ENTITY_SUPPRESSION    | 2             |
| \$RCA_CONTAINED_SUPPRESSION | 3             |
| \$RCA_CONNECTED_SUPPRESSION | 4             |
| \$RCA_ISOLATED_SUPPRESSION  | 5             |
| \$RCA_PEER_SUPPRESSION      | 6             |

Figure 11-41. Suppression types in RCA rules

## Determining what plug-ins are enabled

- Not all gateway plug-ins are enabled by default
  - The **Disco** plug-in is not enabled by default in 4.2
  - This plug-in can rediscover a device
- Determine what plug-ins are enabled:

```
$ ITNMHOME/scripts/perl/scripts/ncp_gwplugins.pl -domain ITNM4GO
```

| Plug-in name     | Is enabled |
|------------------|------------|
| Adaptive Polling | true       |
| Disco            | false      |
| Failover         | true       |
| RCA              | true       |
| SAE IP Path      | true       |
| SAE ITNM Service | true       |
| SAE MPLS VPN     | true       |
| zNetView         | false      |

[The gateway and RCA](#)

© Copyright IBM Corporation

*Figure 11-42. Determining what plug-ins are enabled*

The `ncp_gwplugins.pl` script can be used to:

- Determine whether a plug-in is enabled
- Enable a plug-in
- List the event maps in which each plug-in has a subscribed interest
- Subscribe a plug-in to have an interest in a particular event map

You have an opportunity to use this command in the exercise that is associated with this unit.

## Enabling a plug-in

- To enable for a single domain:

```
cd /opt/IBM/tivoli/netcool/precision/scripts/perl/scripts
./ncp_gwplugins.pl -domain ITNM4GO -plugin Disco -enable
Enabled 'Disco' plug-in in domain ITNM4GO
```

- To enable for all domains:

```
./ncp_gwplugins.pl -domain ITNM4GO -plugin Disco -enable -global
Enabled 'Disco' plug-in in the global domain
```

Figure 11-43. Enabling a plug-in

## 11.4. New rules for special events

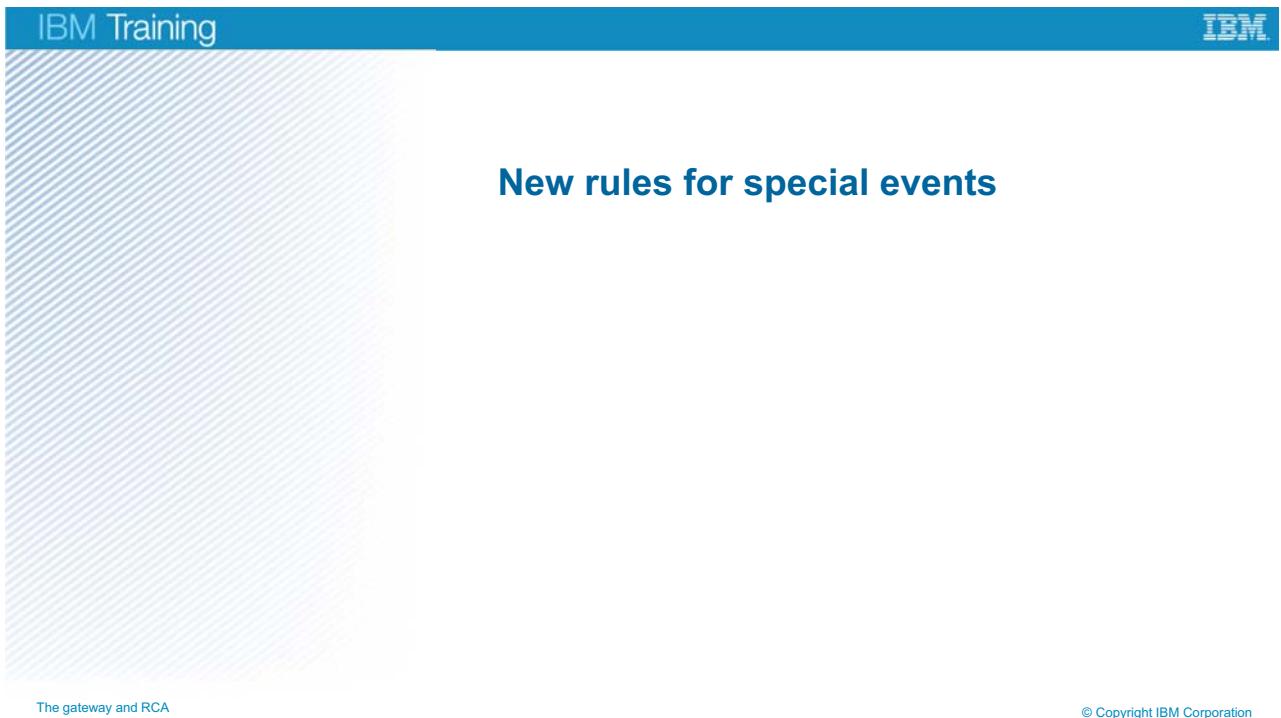


Figure 11-44. *New rules for special events*

## 1. Does the gateway already handle the event?

A customer wants to automatically rediscover any device that reports a cold start trap.

Determine whether gateway is already configured to handle these events:

- If you are using version 4.2 or earlier of NcKL, is the **EventId** listed in the gateway **NcoGateInserts.cfg config.precedence** table?
- If you are using version 4.3 or later, was the field assigned in a NcKL rules file?
  - **SNMPTRAP-coldStart** is not listed in the table and has an empty NmosEventMap
  - The **genericip-event** eventMap handles events that are not tied to other eventMaps
- Is the **NmosEventMap** field of an event populated in a rules file?

*Figure 11-45. 1. Does the gateway already handle the event?*

The easiest way to determine whether a gateway already handles an event is to **grep** for that **eventMap** in the `$NCHOME/etc/precision` directory. The `NcoGateInserts.cfg` file contains the entries into the **config.precedence** table. All SNMP traps in that file begin with `SNMPTRAP-trapName`.

Verify whether specific events that you want to process in a rule have the correct **NmosEventMap**. You can determine whether an event has the correct **NmosEventMap** in two ways. The easiest way is to look at the details of an existing event. Alternatively, carefully view the applicable rules file for that event to see how this field is set.

All events that are not explicitly tied to event maps get assigned to the **genericip-event** eventMap.

In the preceding case scenario, the `$NCHOME/etc/rules/snmptrap.rules` file parses the trap. The following code shows the relevant section of that file for cold start traps:

```
switch($generic-trap)
{
 $OS_EventId = "SNMPTRAP-coldStart"
 @AlertGroup = "Generic"
 @AlertKey = ""
 $DEFAULT_Severity = 2
 $DEFAULT_Type = 13
 $DEFAULT_ExpireTime = 1800
 @Identifier = @Node + " " + @AlertGroup + " " + @Type + " " + @Agent + "
" + @Manager + " " + $generic-trap
}
```

This rules file does not set the value for **NmosEventMap**. This **EventId** is not listed in the `NcoGateInserts.cfg` file. The **genericip-event** eventMap processes events for which no eventMap is specified.

## 2. Determine whether plug-in handles event

- Is the event handled by the Disco plug-in? (or other specific plug-in)
- Determine whether the specific plug-in handles the **eventMap** for the specific event

```
netcool@itnm41server:/opt/IBM/tivoli/netcool/precision/scripts/perl/scripts> ncp_perl
./ncp_gwplugins.pl -domain ITNM4GO -plugin Disco

Plug-in Disco is enabled in domain ITNM4GO

This plug-in has registered interest in eventMaps:
Reconfiguration
This plug-in handles events with states:
Occurred
This plug-in has the configuration variables set:
StitcherSubDir = Disco
```

Figure 11-46. 2. Determine whether plug-in handles event

Plug-ins handle all events within registered eventMaps. In this example, the Disco plug-in is interested in the eventMap **Reconfiguration**. So, if the cold start traps and syslog configuration change events can have this eventMap set, the Disco plug-in can handle them automatically.

### 3. Reassign event to trigger plug-in

- Edit the `EventGatewaySchema.cfg` file to trigger the Disco plug-in without having other events also trigger rediscovery
- Either:
  - Define a new `eventMap` for use by the events  
*or*
  - Reuse existing event maps that are already used by the **Disco** plug-in

The gateway and RCA

© Copyright IBM Corporation

Figure 11-47. 3. Reassign event to trigger plug-in

## Example of reusing an existing event map

The **SNMPTRAP-coldStart** event currently has no **eventMap** assignment

- By default, **coldStart** events are processed with the **genericip-event** event Map
- The Disco plug-in works on events with the **Reconfiguration** event map
- Both the **genericip-event** and **Reconfiguration** event maps call the **LookupMainNode** stitcher
  - You do not lose any functions by reassigning coldStart traps to the **Reconfiguration** event Map instead of the **genericip-event** map

The easiest method is to modify the probe rules file to specify the correct event map that triggers the Disco plug-in:

- In the probe rules file for the cold start event, this specification happens with a simple assignment:

```
@NmosEventMap="Reconfiguration"
```

- This rules file is in **\$NCHOME/etc/rules/snmptrap.rules**

Figure 11-48. Example of reusing an existing event map

In the example that is shown in this slide, a syslog message that indicates a configuration change had its eventMap changed from **EntityFailure** to the new **EntityFailureWithRediscovery** eventMap. By running the **ncp\_gwplugins** script, you can see that three other plug-ins operated on previous occurrences of this event before you change the eventMap name. If other plug-ins did need to process this alarm, those plug-ins need a registered interest in the new eventMap name. For this customer, the zNetView plug-in is not used, so you do not need to enable it.

The Adaptive Polling plug-in is used to populate the **activeEvent** table in NCIM, allowing poll policies to be defined. If the customer uses Adaptive Polling, the new eventMap must feed in to the Adaptive Polling plug-in.

This event continues to be used for RCA unless the customer disables it. Run the following commands to register the interest of these plug-ins in the new eventMap name:

```
ncp_perl $ITNMHOME/scripts/perl/scripts/ncp_gwplugins -domain ITNM4GO
 -plugin 'Adaptive Polling' -add -eventMap EntityFailureWithRediscovery
ncp_perl $ITNMHOME/scripts/perl/scripts/ncp_gwplugins -domain ITNM4GO
 -plugin RCA -add -eventMap EntityFailureWithRediscovery
```

## Example of creating an eventMap (1 of 2)

Creating an event map that triggers a plug-in is a three-step process:

1. Insert the new eventMap into the **config.eventMaps** table (in the **EventGatewaySchema.cfg** file)
  - Specify the starting stitcher

```
New eventMap copied from preceding section
insert into config.eventMaps
(
 EventMapName,
 Stitcher
)
values
(
 "EntityFailureWithRediscovery",
 "LookupMainNode";
)
```

The gateway and RCA

© Copyright IBM Corporation

Figure 11-49. Example of creating an eventMap (1 of 2)

In this unit, you see two ways to get events handled by a gateway plug-in. Creating an **eventMap** is the more complex of the two solutions that are presented. Another less complex solution is presented later in this unit.

## Example of creating an eventMap (2 of 2)

- Configure rules file to assign appropriate event map name and appropriate weighted priority

```
$OS_EventId = "SNMPTRAP-coldStart"
@NmosEventMap = "EntityFailureWithRediscovery"
```

- Configure the plug-in to register interest in the new eventMap

```
ncp_gwplugins -domain ITNM4GO -plugin Disco -add -eventMap
EntityFailureWithRediscovery
```

**Caution:**

- If you reassign an event to a different eventMap, determine whether other plug-ins that use the old eventMap can still handle the event
- Example: If you change a syslog configuration message to use the **EntityFailureWithRediscovery** map instead of **EntityFailure**, determine what other plug-ins have a registered interest in **EntityFailure**

```
ncp_gwplugins -domain ITNM4GO -eventMap EntityFailure
```

```
The following plug-ins have registered interest in event map 'EntityFailure':
Adaptive Polling, RCA, zNetView
```

Figure 11-50. Example of creating an event map (2 of 2)

#### 4. Restart the gateway

- The gateway does not see any of the changes to eventMaps or plug-in registrations until it is restarted
- Because the gateway runs under the **ncp\_ctrl** process, you can end the gateway process and it automatically restarts
- Syntax errors can cause the RCA plug-in to not work while other gateway functions remain intact

The gateway and RCA

© Copyright IBM Corporation

Figure 11-51. 4. Restart the gateway

Anytime you change gateway configuration files, you must restart the gateway to get it to process the updated configuration files. When you start most other Tivoli Network Manager processes, they fail to start if the configuration files contain any errors. However, if you make an error in a gateway plug-in configuration file, only the specific plug-in fails to start. The gateway and other plug-ins continue to work.

## 11.5. Troubleshooting gateway event processing

Most of the material in this section is for support personnel. Others can regard this section as reference material.

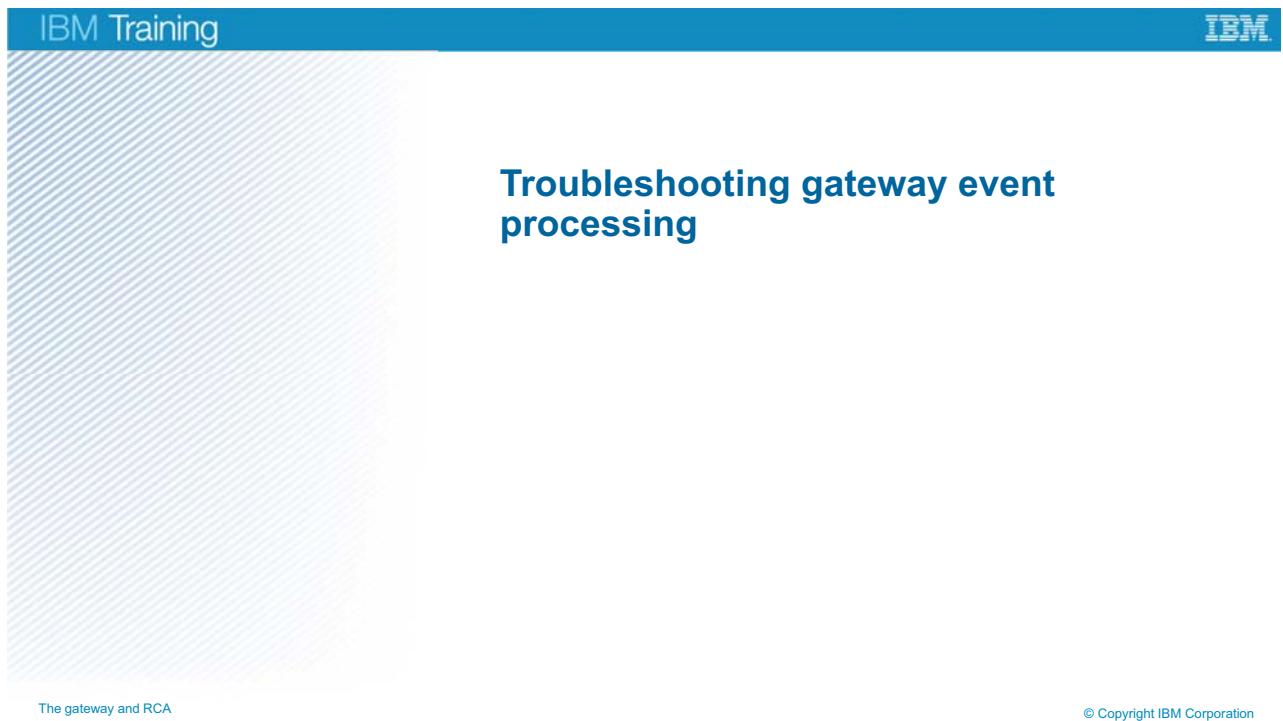


Figure 11-52. Troubleshooting gateway event processing

## Troubleshooting with trace and log files

- Trace file: `ncp_g_event.domainName.trace`
- Log file: `ncp_g_event.domainName.log`: Generally more useful than the `*.trace` file for troubleshooting RCA

```
ncp_g_event.log
```

```
2010-10-06T16:43:51: Information: I-EVE-002-012: [2995145632t] RCA plug-in is enabled
2010-10-06T16:43:51: Information: I-AMO-001-026: [2995145632t] Initializing RCA plug-in
2010-10-06T16:43:51: Information: I-AMO-001-043: [2995145632t] Initializing Amos RCA
Manager
2010-10-06T16:43:51: Information: I-AMO-001-043: [2995145632t] Initializing Amos Stitcher
Manager
2010-10-06T16:43:51: Information: I-AMO-001-044: [2995145632t] Initialization of Amos
Stitcher Manager succeeded
2010-10-06T16:43:51: Information: I-AMO-001-043: [2995145632t] Initializing Amos Store
Manager
2010-10-06T16:43:51: Information: I-AMO-001-043: [2995145632t] Initializing Amos Event
Queue
```

The gateway and RCA

© Copyright IBM Corporation

Figure 11-53. Troubleshooting with trace and log files

## Tracking progress of event (1 of 2)

```
Passing event 479 to RCA plug-in
<Serial=479> Received this event: State=5 (Occurred)
<Serial=479> Added this event to queue: IsRequeueable=0 (QueueLength=2)
<Serial=479> Popped this trigger event from queue (QueueLength=0)
<Serial=479> Processing this trigger event from queue: State=5 (Occurred)
<Serial=479> Setting TimedEscalation=1: This trigger event will be processed by the Timed
 Suppression rule shortly.
<Serial=479> Inserting this new event: EntityId=106 EntityName=pe5-
 cr38.core.eu.test.lab_SLOT_I13_R6 EventId=SNMPTRAP-riverstone-RIVERSTONE-NOTIFICATIONS-MIB-
 rsEnvirFanFailed
<Serial=479> Finished processing this trigger event from queue
...
<Serial=479> The Timed Suppression rule will process this trigger event now.
Updating event Serial=479, setting TimedEscalation=2
<Serial=479> Firing stitcher 'ProcessEvent': State=5 (Occurred)
...
```

The gateway and RCA

© Copyright IBM Corporation

Figure 11-54. Tracking progress of event (1 of 2)

## Tracking progress of event (2 of 2)

```
<Serial=479> Executing RCA stitcher rule 'AmosGetContainedEntities' for this trigger event
The number of ContainedSuppression entities found for EntityId=106 is 0
<Serial=479> Executing RCA stitcher rule 'AmosGetConnectedEntities' for this trigger event
The number of ConnectedSuppression entities found for EntityId=106 is 0
<Serial=479> Executing RCA stitcher rule 'AmosGetIsolatedEntities' for this trigger event
Finding entities isolated by EntityId=106 (PollerEntityId=7051)
The number of IsolatedSuppression entities found for EntityId=106 is 0
<Serial=479> Executing RCA stitcher rule 'AmosGetEvents' for this trigger event
The number of events on EntityId=7051 is 1
<Serial=479> Executing RCA stitcher rule 'AmosGetEvents' for this trigger event
The number of events on EntityId=106 is 1
Stitcher 'ProcessEvent' completed
Timed Event Suppression: 1 timed events were found and processed
Timed rules processor has now unlocked mojo store
Stitcher 'TimedEventSuppression' completed
```

The gateway and RCA

© Copyright IBM Corporation

Figure 11-55. Tracking progress of event (2 of 2)

## Troubleshooting the handling of events (1 of 2)

- Does the event match the **nco2ncp EventFilter** in **EventGatewaySchema**?
  - If so, does the gateway know how to process the event?
    - Is the **EventId** of the event listed in the **config.precedence** inserts in the schema files?
    - Is the **NmosEventMap** field of the event populated?
  - If so, do the fields of the event contain the expected data?
    - The expected data format for each event map is given in **EventGatewaySchema**
  - Does the entity match an entity in the topology?
    - If so, the **NmosObjInst** field contains the main node entity ID
    - If not, the event is not passed to any plug-ins

Figure 11-56. Troubleshooting the handling of events (1 of 2)

## Troubleshooting the handling of events (2 of 2)

- If the entity matches one in the topology, does a specific plug-in, such as the RCA plug-in, see an event?

Use `ncp_gwplugins.pl` to list the event maps and states that the plug-in processes:

```
ncp_perl ncp_gwplugins.pl -domain NCOMS -plugin
RCA
```

- At `-messagelevel info`, all serial numbers that are passed to all plug-ins are logged

```

Plugin RCA is enabled in domain NOI_AGG_P

The Amos Root-Cause Analysis (RCA) plugin
This plugin has registered interest in eventMaps:
ChassisFailure
EMSNonPollingEvent
EMSPollingEvent
EntityFailure
EntityIfDescr
EntityMibTrap
ItmmlinkDownIfIndex
LinkDownIfDescr
LinkDownIfIndex
LinkDownIfName
NbrFail
NbrFailIfDescr
OSPFIfStateChange
OSPFIfstateChangeIP
OspfIfState
PollFailure
PrecisionMonitorEvent

This plugin handles events with states:
Cleared
Deleted
Occurred
ReAwakened
ReOccurred
ReSync
Updated
```

© Copyright IBM Corporation

The gateway and RCA

Figure 11-57. Troubleshooting the handling of events (2 of 2)

## Troubleshooting entity selection

Did the event map expect to match an entity?

- The event map **Stitcher** field is required to do a topology lookup
- Only network events are matched to entities
  - Other events, such as Tivoli Network Manager Status events, do not correspond to entities
- If so, is the expected entity in the NCIM cache?  

```
ncp_oql -domain NCOMS -service EventGateway -query
"select * from ncimCache.entityData where filter;"
```
- Is the expected entity in NCIM?  

```
ncp_oql -domain NCOMS -service ncim -username ncim -query "select * from entityData
where filter;"
```
- Check that no mismatch exists between NCIM and the cache
  - The data must always be consistent

Figure 11-58. Troubleshooting entity selection

## Troubleshooting field updates

- Was an entity found?
  - Is **NmosObjInst** nonzero? If it has a zero value, the entity was not in the discovery
- Is there a stitcher that has a matching event map for the event?
  - What stitcher populates that field?
  - Did the event match an **EventMap** in the stitcher?
  - Modify the stitcher to add the correct **EventMap** or change the rules file on the appropriate probe to send an **EventMap** that matches one in the stitcher
- What type of entity was found?
  - Check the **entityType** of the **NmosEntityId** found
  - Some fields are available only for chassis; some are only available for interfaces
- Is the expected value populated for the NCIM entity?
  - Some values can be NULL

The gateway and RCA

© Copyright IBM Corporation

Figure 11-59. Troubleshooting field updates

## 11.6. Service-affecting events

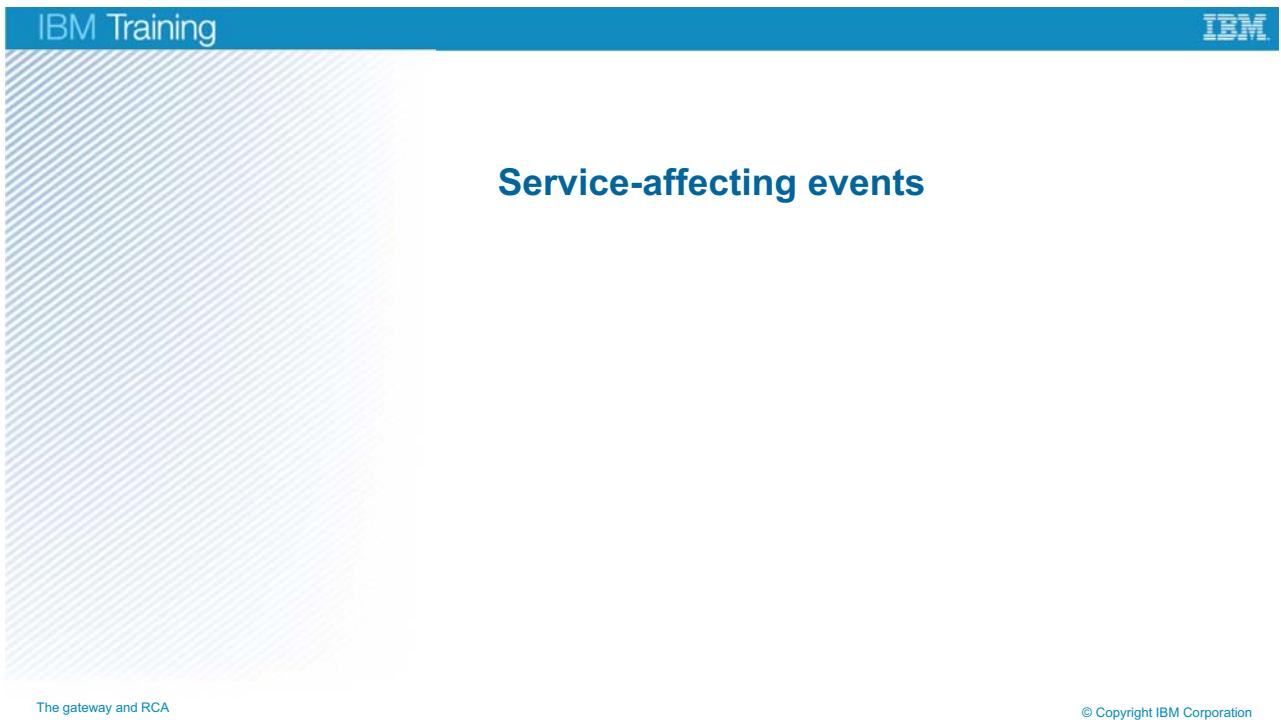


Figure 11-60. Service-affecting events

## Introduction

- A service-affecting event (SAE) alert warns operators that one or more events are impacting a critical customer service
- Two examples of SAE events that are generated on two different customer VPNs:
  - SAE generated on customer-1 VPN because of an **MPLS VRF Down** trap on a provider edge (PE) router interface
  - SAE generated on customer-3 VPN because of a **LinkDown** trap on a customer edge (CE) router interface
- Each SAE appears as an alert in the **Active Event List (AEL)** or **Event Viewer**
  - Warns operators that events negatively affect a customer VPN
  - Operators can right-click the SAE and issue a command to view the underlying events that caused the SAE

[The gateway and RCA](#)

© Copyright IBM Corporation

*Figure 11-61. Introduction*

A service-affecting event (SAE) alert warns operators that events are negatively affecting a critical customer service.

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 that are 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

Operators can right-click the SAE and issue a command to view the underlying events that caused the SAE.

## Service-affecting events plug-in

- The SAE engine is now a plug-in to the gateway application
- New configuration file name: **SAESchema.cfg**
- Each service type has a configuration file:
  - **SaeMplsVpn.cfg**
  - **SaeIPPath.cfg** aids in tracing path for SAE events
  - **SaeItnmService.cfg**
- A new configuration has **ncmonitor** database tables:
  - **ncmonitor.gwPlugins**
  - **ncmonitor.gwPluginConf**

The gateway and RCA

© Copyright IBM Corporation

*Figure 11-62. Service-affecting events plug-in*

You can configure the SAE plug-in to generate more SAE types than the three that are provided by default. For example, you can configure the plug-in to create SAE events for MPLS VPN edge entities (one type of SAE) and for MPLS VPN core entities (another type of SAE).

In this example, the existing configuration file `SaeMplsVpn.cfg` is customized to add an extra MPLS VPN SAE service type 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 `$NCHOME/etc/precision/SAEMplsVpn.cfg`.

Open the `SAEMplsVpn.cfg` configuration file. The default insert creates an MPLS VPN Edge Service:

```
insert into config.serviceTypes
(
 ServiceTypeName,
 CollectionEntityType,
 ConstraintFilter
)
values
(
 "MPLS VPN Edge Service",
 17, -- networkVpn
 "networkVpn->VPNTYPE <> 'MPLS Core'"
);
```

Add an insert after the existing insert. The new insert should read as follows:

```
insert into config.serviceTypes
(
 ServiceTypeNames,
 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 table such as **networkVpn (17)**, as described in this example. In this case, the SAE service types are mutually exclusive sets. Otherwise, one wins over the other where they overlap. For example, the service types that are described in this example do not overlap because they have complementary **ConstraintFilter** settings as follows:

```
networkVpn->VPNTYPE <> 'MPLS Core'
networkVpn->VPNTYPE = 'MPLS Core'
```

## Configuring summary field information in service-affecting events

To make service-affecting events more meaningful for operators, you can configure the SAE plug-in to insert customer-related information into the Summary field of a service-affecting event.

The configuration files in the SAE plug-in where you implement this change are as follows:

- \$NCHOME/etc/precision/SaeIpPath.cfg
- \$NCHOME/etc/precision/SaeMplsVpn.cfg
- \$NCHOME/etc/precision/SaeItnmService.cfg

The field that is 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.

Open the `SaeMplsVpn.cfg` configuration file.

Modify the insert statement by adding the text in bold to insert data from a relevant field in the service record in the NCIM cache into the **CustomerNameField** field. For example, the following statement inserts 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"
)
```

## SAESchema.cfg

### CollectionEntityType

- This value correlates with the field entityType of the table **ncim.entityData** in **ncimCache.entityData**
- Common values:
  - 17: networkVpn
  - 34: itnmService
  - 80: ipPath

```
create table config.serviceTypes
(
 ServiceTypeName text not null primary key,
 CollectionEntityType int not null,
 ConstraintFilter text,
 CustomerNameField text,
 unique (ServiceTypeName)
);
```

The gateway and RCA

© Copyright IBM Corporation

Figure 11-63. SAESchema.cfg

## Checking ServiceType queries (1 of 2)

- SAE services can be created from collections or dependencies
- For a specific **ServiceType** such as MPLS VPN Edge Service, the SAE plug-in finds all services of this type with a query such as:

```
SELECT ENTITYID, entityData->DISPLAYLABEL FROM ncimCache.entityData WHERE
ENTITYTYPE = ? AND networkVpn->VPNTYPE <> 'MPLS Core'; Arg (17)
```

- Then, for each service, such as **VC\_100**, find all the members in collections (collects) and all the members in the table dependency

Figure 11-64. Checking ServiceType queries (1 of 2)

## Checking ServiceType queries (2 of 2)

```
ncp_g_event.log
2010-10-01T14:09:52: Debug: D-SAE-001-065: [2995485600t] The following query
will be used to obtain the collected entities for service type VC_100: SELECT
* FROM ncimCache.collects WHERE ENTITYID = ? Arg (<ServiceEntityId>)
2010-10-01T14:09:52: Debug: D-SAE-001-063: [2995485600t] Found 2 collection
members for service VC_100 (404)
2010-10-01T14:09:52: Debug: D-SAE-001-066: [2995485600t] The following query
will be used to obtain the dependency entities for service type VC_100: SELECT
* FROM ncimCache.dependency WHERE dependency(ANY) ->DEPENDENCYTYPE = ? AND
ENTITYID = ? Args (1, <ServiceEntityId>)
2010-10-01T14:09:52: Information: I-SAE-001-062: [2995485600t] No dependent
entities found for ServiceEntityId 404
```

Figure 11-65. Checking ServiceType queries (2 of 2)

## Key ncimCache tables

- The stitcher that processes service-affecting events searches for service members in collections in the topology
  - The SAE process identifies the **CollectionEntityId** and **ConstraintFilter** for each Service Entity found in the configuration file
  - It then identifies the group of entities that are members of the Service Entity by examining the **collects** and **dependency** tables
  - **ncimCache.collects**: SAE uses all collections that are found for the Service Entity ID
  - **ncimCache.dependency**: This table contains SAE-selected groups of entities for those Service Entities that have **dependencyType=1** in this table

Figure 11-66. Key ncimCache tables

## Exercise: The gateway and RCA

The gateway and RCA

© Copyright IBM Corporation  
2017

Figure 11-67. Exercise: The gateway and RCA

Complete the exercise for this unit.

## Review questions

1. What script do you use to see what plug-ins process specific eventMaps?
2. In what file do you find precedent values for different types of events?
3. What field do you set in a probes rules file to specify a particular event map apply to an event?
4. From what gateway stitcher do you call a new custom gateway stitcher that is used to analyze an event to specify a root cause?
5. What value do you set in the `RCASchema.cfg` file to force Tivoli Network Manager to always attempt to do network fault isolation?

The gateway and RCA

© Copyright IBM Corporation

Figure 11-68. Review questions

Write your answers here:

## Unit summary

- Disable and enable Tivoli Network Manager gateway plug-ins
- Enrich events with topology-related information

The gateway and RCA

© Copyright IBM Corporation

*Figure 11-69. Unit summary*

## Review answers

1. Use the `ncp_gwplugins.pl` script to see what plug-ins process specific eventMaps.
2. The precedent values for different types of events are set in the `NcoGateInserts.cfg` table.
3. In a probes rules file, set the `NmosEventMap` field to specify a particular event map for an event.
4. Call custom gateway stitchers from the `ProcessProblemEvent.stch` if the new stitcher analyzes event for root cause.
5. Set the `HonourManagedStatus` field in the `RCASchema.cfg` file to force Tivoli Network Manager to always attempt to do network fault isolation (even on unmanaged devices).

The gateway and RCA

© Copyright IBM Corporation

Figure 11-70. Review answers

# Unit 12. Tivoli Network Manager scripts

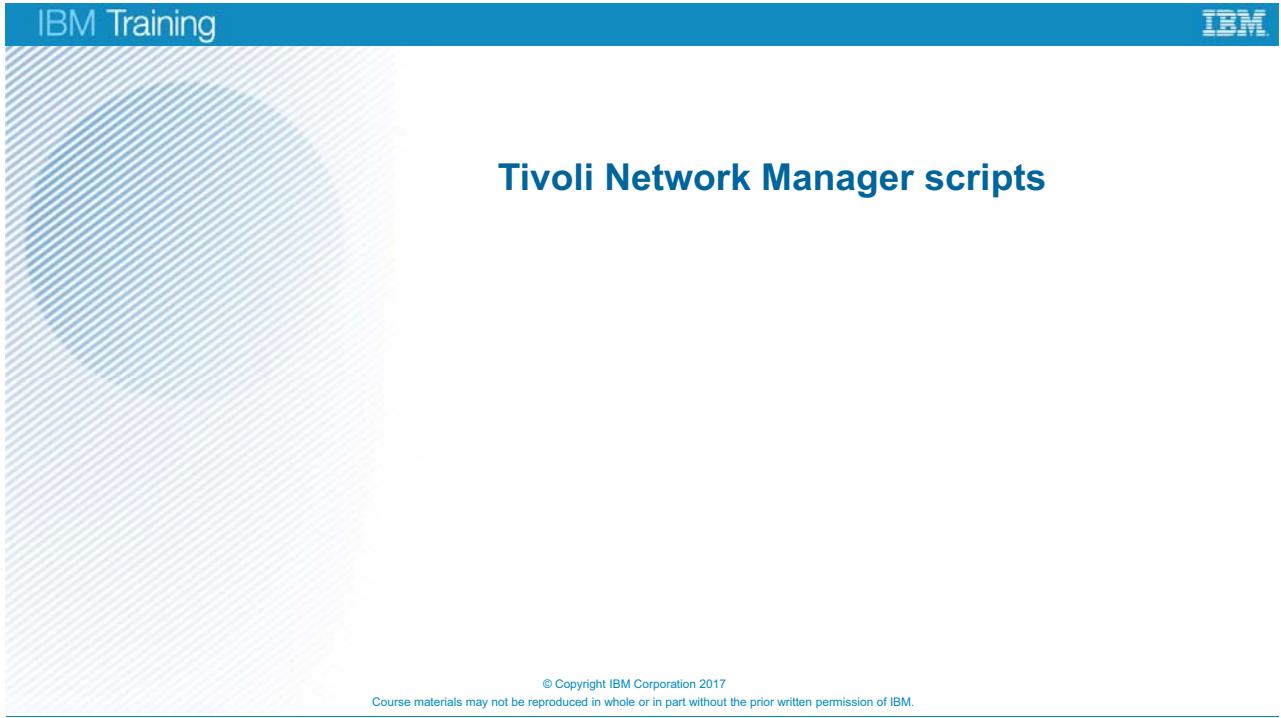


Figure 12-1. *Tivoli Network Manager scripts*

## Estimated time

00:30

## Overview

In this unit, students learn how to use provided scripts to simplify or automate tasks with Tivoli Network Manager.

## How you will check your progress

This unit has no student exercises.

## Objectives

- Locate administrative Perl scripts
- Use a script to gather discovery data to provide to IBM technical support

Figure 12-2. Objectives

## \$ITNMHOME/scripts/perl/scripts (1 of 3)

- `audit.pl` generates a status report for the discovery process, agents, and stitchers (refreshes periodically) in the `audit.html` file
- `BuildSeedList.pl` generates a hosts file listing all hosts with main IP addresses that can be used to shorten future discovery times
- `discoAgentsUsed.pl` determines which discovery agents were used to discover the most recently discovered devices in the domain
- `domainCreate.pl` creates a set of domain-specific files for a new Tivoli Network Manager domain
- `domainDrop.pl` removes the network domain and its polling policies, but leaves the configuration information and topology cache

Figure 12-3. \$ITNMHOME/scripts/perl/scripts (1 of 3)

## \$ITNMHOME/scripts/perl/scripts (2 of 3)

- `GetDiscoCache.pl` creates a `.tar` file with records of every discovery agent, every stitcher, and the network topology
- `get_policies.pl` is a script that backs up poll policies and can move them between domains
- `get_network_views.pl` exports domain network views to a file
- `itnmMetaDiscoAudit.pl` generates a report that contains audit information on device classification, and missing device metadata
- `ncp_upload_expected_ips.pl` ensures that a list of IP addresses is being properly pinged
- `ITNMIP_Listener.pl` listens to messages that are passed between processes on the bus such as updates, notifications, and status events
- Examples are provided for you to create your own custom scripts:
  - `oql_example.pl` provides examples of Perl-scripted queries into the OQL database
  - `snmp_example.pl` provides examples of SNMP queries into a specified device

Figure 12-4. \$ITNMHOME/scripts/perl/scripts (2 of 3)

To run the `itnmMetaDiscoAudit.pl` script to generate a report, use a command similar to the following example:

```
$NCHOME/precision/bin/ncp_perl $NCHOME/precision/scripts/perl/scripts/
itnmMetaDiscoAudit.pl -domain NCMS -report > my_report.txt
```

### \$ITNMHOME/scripts/perl/scripts (3 of 3)

- `ncp_db_access.pl` accesses the topology database, historical polling database, and distributed polling database to determine whether a firewall prevents access to the database
- `listEntities.pl` retrieves device information from the `ncimCache.entityData` database table and outputs the information in HTML format
- `itnm_poller.pl` enables, disables, or checks the status of poll policies
- `PrintCacheFile.pl` prints portions of a specified cache file in human-readable format for scripting and debugging purposes
- `restart_disco_process.pl` stops the currently running discovery and starts a new instance, provided that `ncp_ctrl` started the discovery process
- `snmp_walk.pl` troubleshoots connection and topology issues by walking device and outputs `*.mimic` and `*.snmpwalk` files

Figure 12-5. \$ITNMHOME/scripts/perl/scripts (3 of 3)

## \$ITNMHOME/bin

- **AddNode.pl** is used to add devices to your network topology
- **ManageNode.pl** sets the status of one or more unmanaged devices back to a managed state
- **itnm\_disco.pl** starts and stops network discoveries and shows the status of running discoveries
- **read\_ncp\_cfg.pl** queries **ncp\_ctrl** to extract the current service state of processes that run by **ncp\_ctrl** (mimics **itnm\_status ncp**)
- **RemoveNode.pl**
  - In versions before 4.2: Sets a specified device to unmanaged state and marks a device for removal by the next discovery
  - In version 4.2: Deletes the node from the discovery database (run a partial rediscovery to recalculate connections)
- **scheduleDiscovery.pl** shows when the next full discovery is scheduled
  - You can also use this script to schedule full discoveries
- **UnmanageNode.pl** sets the status of one or more managed devices to an unmanaged state

Figure 12-6. \$ITNMHOME/bin

For example, to set a monthly schedule for discovery, use a command line similar to the following syntax:

```
$NCHOME/precision/bin/ncp_perl $NCHOME/precision/bin/scheduleDiscovery.pl
-domain NCOMS -date 0..31 -time 24_hour_time -v
```

## \$ITNMHOME/scripts/upgrade

- `ITNMDataImport.pl` imports Network Manager configuration files
- `ITNMDataExport.pl` exports Network Manager configuration files
- `ITNMEexportNetworkViews.pl` exports and imports network views and filter data
- `ITNMEexportHistoricalData.pl` exports historical polling data

For more information on scripts, see the *Network Manager IP Edition Version 4 Release 2 Administration Guide*

Figure 12-7. \$ITNMHOME/scripts/upgrade

## Other scripts

- The `$ITNMHOME/scripts/sql` directory contains scripts for creating database tables
- Place custom scripts into the appropriate subdirectory (such as `db2`, `sqlite`, `oracle`) when you need to create custom tables

Figure 12-8. Other scripts

## Unit summary

- Locate administrative Perl scripts
- Use a script to gather discovery data to provide to IBM technical support

Figure 12-9. Unit summary

# Unit 13. Tivoli Network Manager failover



Figure 13-1. Tivoli Network Manager failover

## Estimated time

00:30

## Overview

In this unit, students learn how to configure Tivoli Network Manager failover what takes place when a Tivoli Network Manager process becomes unavailable.

## How you will check your progress

- Answer checkpoint questions
- This unit has no student exercises.

## Objectives

When you complete this unit, you should be able to do the following tasks:

- Configure basic failover
- Explain what happens when failover occurs

*Figure 13-2. Objectives*

## 13.1. Failover basics

Failover in Tivoli Network Manager 4.2 is simpler than it was for versions 3.9 and earlier. It is easy to configure and works well. This section helps you understand the basics of how failover works.

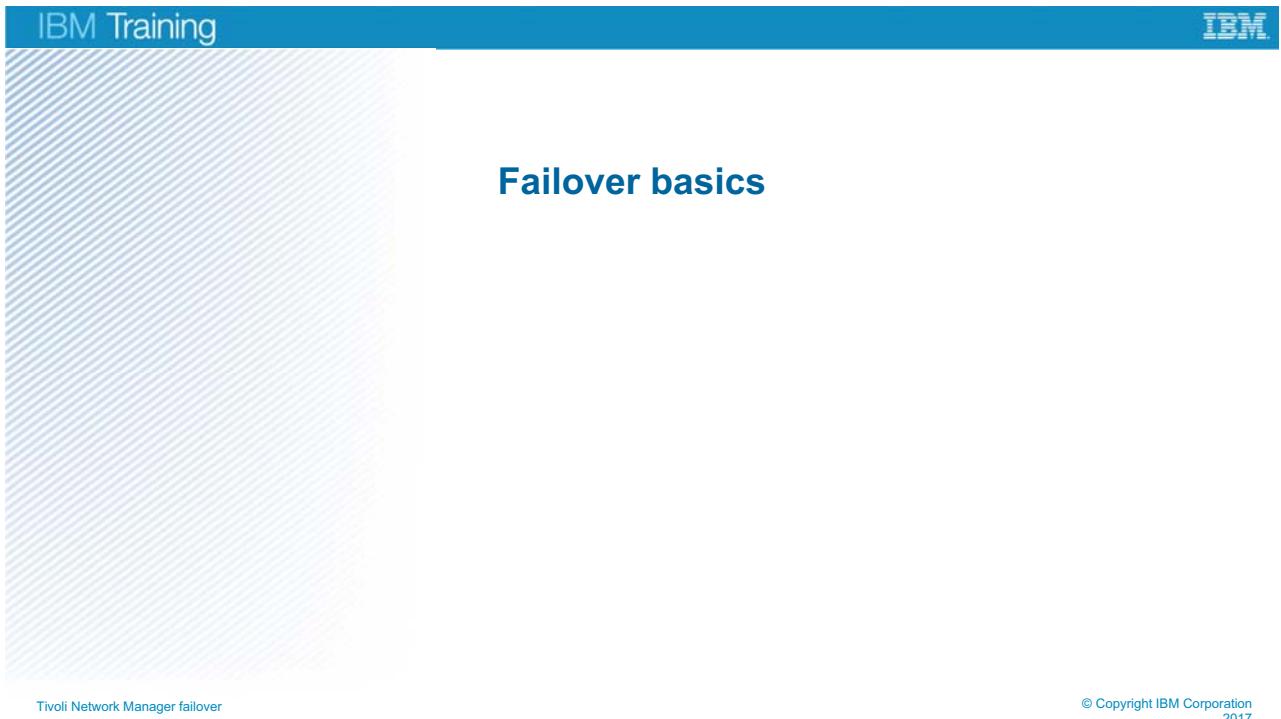


Figure 13-3. Failover basics

## Basic concepts of failover

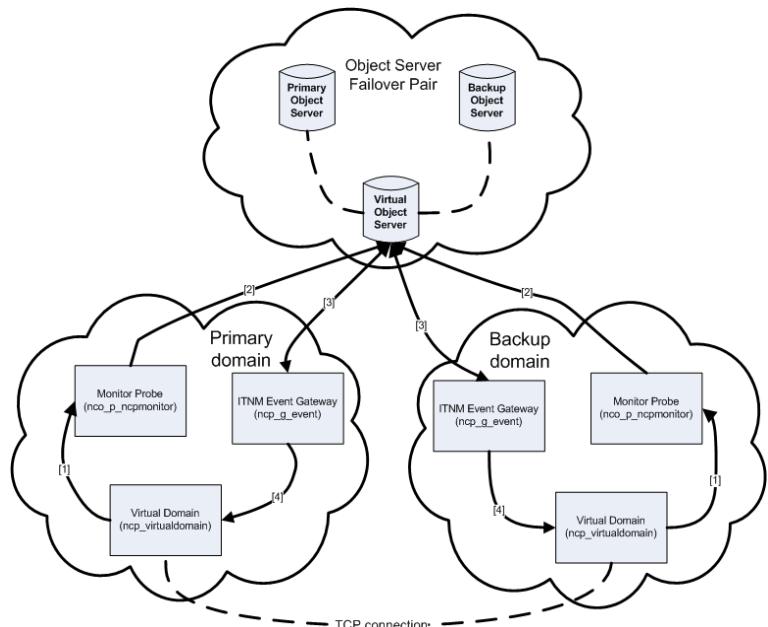
- Failover for Tivoli Network Manager means that critical processes can fail over from one server to another
- Failover from the primary server to the backup server occurs when one of the following events occurs:
  - The gateway fails (**ncp\_g\_event**)
  - The probe fails (**nco\_p\_ncpmonitor**)
  - The MODEL service (**ncp\_model**) fails
  - The connection to the primary domain NCIM database is lost
- The **ncp\_virtualdomain** process sends topology and polling information from the primary server to the backup server
- No data is copied from the backup to the primary

Figure 13-4. Basic concepts of failover

## Architecture overview

- ObjectServer can be virtual with servers in failover mode
- TCP connection is used to ensure data transfer from primary to backup
- The diagram to the right helps show the flow of data before and during a failover event
  - Both domains raise Health Check events and send them to the ObjectServer with the probe
  - Events from both domains are from the ObjectServer by the gateways in each domain
  - Event gateway failover plug-in passes events back to the virtual domain

Tivoli Network Manager failover



© Copyright IBM Corporation 2017

Figure 13-5. Architecture overview

Only the primary server runs the DISCO service. The backup server is intended to temporarily assume the duties of polling, event enrichment, and network fault isolation. It does not need to run discoveries.

The backup server does not run discovery. If the primary server was disabled for a lengthy period, you can manually start a discovery on the backup server. However, when the primary server comes back online, the topology data of the primary server prevails. When you run a new discovery, then the servers synchronize data.

## Event gateway

- The gateway has a failover plug-in that detects conditions that trigger a failover
- The failover plug-in must be enabled in both primary and backup domains in order for failover to work
- Events that are related to failover are routed through the virtual domain process between servers
- The standby domain never updates the ObjectServer
  - Under steady-state conditions, the backup domain is the standby server
  - If the primary domain is still running during failover, it goes into standby mode

Figure 13-6. Event gateway

The event gateway schema contains distinct **EventFilter** and **StandbyEventFilter** options.

The **StandbyEventFilter** must allow at least health check events for failover to work.

If a process failure triggers a failover according to the filter rules in the **VirtualDomainSchema** file, the primary domain goes into standby mode.

## Database replication

- Tivoli Network Manager does not do database replication
- The database engine must be configured to replicate data
- The appendix contains basic information about configuring DB2 database replication

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-7. Database replication*

If you need to do maintenance activities on a Tivoli Network Manager server, the failover server needs to access the NCIM database. For this reason, it is a good idea to configure the database to replicate somewhere the backup server can access.

## Replicating failover

- In replicating failover, both IBM Tivoli Network Manager servers connect to a DB2 database that is configured for high availability disaster recovery (HADR)
- Tivoli Network Manager sends data to the DB2 HADR server configuration
  - Tivoli Network Manager does not replicate the information from one database server to the other
- The primary database server replicates its data to the backup database server
- Data is copied unidirectionally from the primary server to the backup server:
  - Topology data
  - Configured network polls
  - Configured network views
  - SNMP configuration
- Work with your database administrator to configure high availability

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-8. Replicating failover*

## 13.2. Configuring failover

Two steps are necessary to configuring failover. Edit the `ConfigItnm.cfg` file. Then, edit the `ServiceData.cfg` file to force the use of static ports for processes that communicate between the two servers.

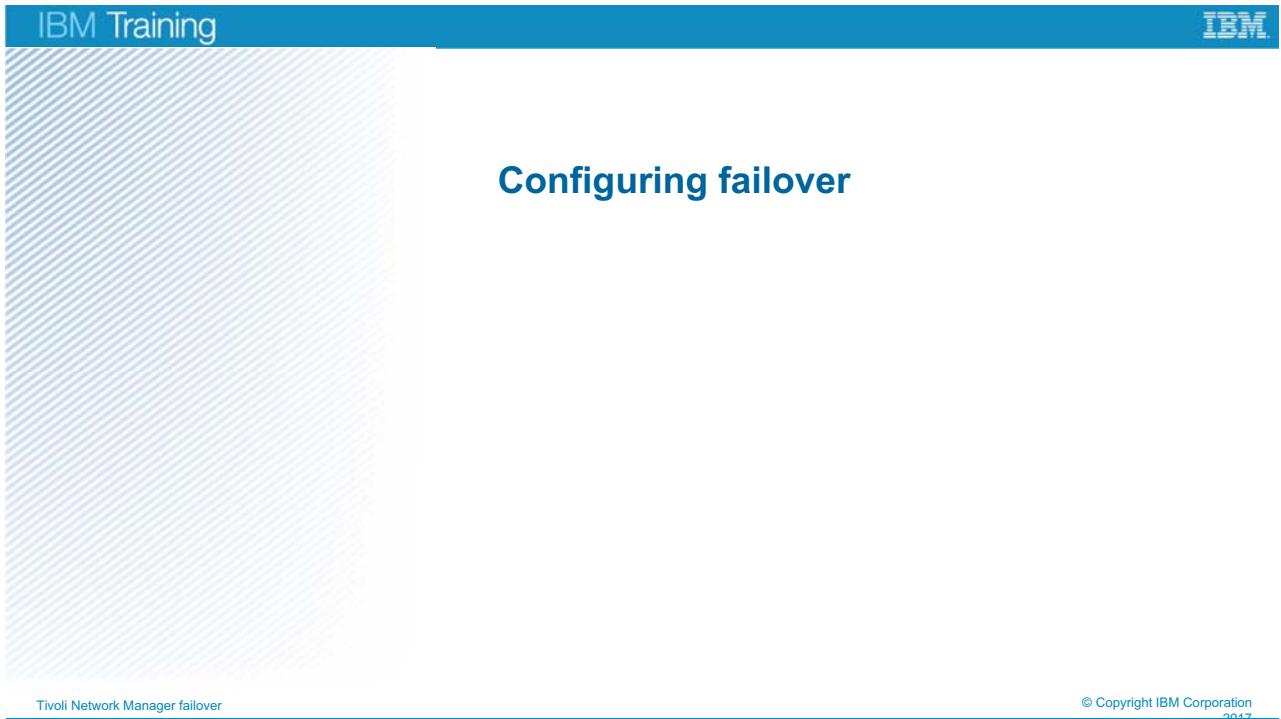


Figure 13-9. Configuring failover

## Failover configuration steps

1. Make sure the OMNIbus ObjectServers are installed first
2. Install the primary Network Manager core components on host 1
3. Install the backup Network Manager core components on host 2
4. Install the Network Manager GUI framework on host 3
  - The GUI framework includes the following software:
    - WebSphere Application Server
    - Dashboard Application Services Hub
    - Network Manager GUI components
    - Tivoli Netcool/OMNIbus Web GUI
    - Reporting Services
5. Configure the primary Network Manager by editing the `$NCHOME/etc/precision/ConfigItnm.cfg` file for failover and start it
6. Copy the file from the primary server to the backup server: `$NCHOME/etc/precision/ServiceData.cfg`
7. Configure the backup Network Manager by editing the `$NCHOME/etc/precision/ConfigItnm.cfg` file for failover, and start it

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-10. Failover configuration steps*

Tivoli Network Manager core components must be installed before you install the GUI framework components.



### Note

If you install the Dashboard Application Services Hub on a machine with no other products, performance is likely to be better than if you install it on a machine with other products.

## The ConfigInm.cfg file

- Installation creates a domain-specific copy of this file
- Specifies the ObjectServer name even if failover is not enabled
- The file must have the same contents on the primary and backup servers
- Every failover-aware process gets information from this file

```

insert into itnmDomain.failover
(
 FailoverEnabled,
 PrimaryDomainName,
 BackupDomainName,
 VirtualDomainName
)
values
(
 0,
 "ITNM4GO_P",
 "ITNM4GO_B",
 "ITNM4GO_V"
);

insert into itnmDomain.objectServer
(
 ServerName
)
values
(
 "NCOMS"
);

```

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-11. The ConfigInm.cfg file*

## Locking ports in ServiceData.cfg

1. Start the first server
2. Make a backup copy of the `$NCHOME/etc/precision/ServiceData.cfg` file
3. Edit the 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
```
4. Change the PORT setting to the required value
  - Change the string `DYNAMIC:YES` to `DYNAMIC:NO`
  - This change forces the process to use the same address and port next time it starts
5. Save the `ServiceData.cfg` file
6. On the second server, make a backup copy of the `ServiceData.cfg` file
7. Copy the relevant line from the `ServiceData.cfg` file on the first server to the `ServiceData.cfg` file on the second server
8. Save the `ServiceData.cfg` file

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-12. Locking ports in ServiceData.cfg*

Follow these steps to define fixed TCP ports for Tivoli Network Manager processes. Configuring static ports in the `ServiceData.cfg` file ensures communications for key processes that are used in failover. It is important that the `ncp_virtualdomain` process is configured to have a static port. If the servers must communicate through an intervening firewall, notify the firewall administrator of the ports that are used for communication between servers.

## 13.3. What happens with failover

When one of four core processes fails or hangs on the primary server, the backup server learns about it through one of two methods. If the primary server has some processes still running, it can send a message about its failed process to the backup server by using the **ncp\_virtualdomain** process. If the primary server has a catastrophic failure, the backup server sees that the primary server is no longer sending health check messages to the Netcool/OMNibus Objectserver. When either of these conditions occurs, the backup server becomes the acting primary server.



Figure 13-13. What happens with failover

## When failover happens

- The virtual domain triggers failover of the domain
- The backup domain becomes active
  - Backup poller starts polling
  - Back up event gateway switches from the StandbyEventFilter to the EventFilter and starts updating events in the ObjectServer
- The primary domain, if it still runs, goes into standby
  - Primary poller suspends polling
  - Primary event gateway switches from the EventFilter to the StandbyEventFilter and no longer updates events in the ObjectServer

Figure 13-14. When failover happens

## When fallback happens

- The virtual domain triggered failover of the domain
- The primary domain becomes active
  - Primary poller resumes normal polling activities
  - Primary event gateway switches from the **StandbyEventFilter** to the **EventFilter** and starts updating events in the ObjectServer
- The backup domain reverts to standby
  - Backup poller suspends polling
  - Backup event gateway switches back to the **StandbyEventFilter** from the **EventFilter** and no longer updates events in the ObjectServer

Figure 13-15. When fallback happens

## Tivoli Network ManagerHealthChk events

- Each virtual domain process periodically raises health check events about the local domain
- Filters in the **VirtualDomainSchema** govern whether Tivoli Network Manager sends a success or failure health check message to the ObjectServer
- If no health check is received from the remote domain after a configurable period, the virtual domain raises a failure event on behalf of the remote domain
  - Handles the condition where the primary server goes down
  - Node field must identify the domain of the event
- A health check failure event for the primary domain triggers failover
  - A health check success event on the primary domain triggers fallback

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-16. Tivoli Network ManagerHealthChk events*

Only the primary domain name exists in the NCIM database.

All Tivoli Network Manager events are raised with the primary domain name in the **NmosDomainName** field.

- The `$Domain` constant in the monitor probe rules file evaluates to the primary domain name.
- If the `ConfigItnm.cfg` file is used, the `$DOMAIN_NAME` constant in the `EventGatewaySchema` `EventFilter` evaluates to the primary domain name.
- If the `ConfigItnm.cfg` file is used, the `$DOMAIN_NAME` constant in the gateway stitchers evaluates to the primary domain name.

## ItnmDatabaseConnection event

- This new problem event type is generated in 4.1.1 when **ncp\_model** or **ncp\_poller** loses connection to the NCIM database
  - It is cleared upon reconnection
- Can be used to generate a local domain health check failure event
  - Filter on the NCIM service in the **VirtualDomainSchema** file
  - If the connection is lost and not regained within 5 minutes of the initial failure, the default behavior is to raise a health check failure
- To react to this event, the **StandbyEventFilter** in the **EventGatewaySchema** must accept it
  - The default is to allow both **ItnmHealthChk** and **ItnmDatabaseConnection** events

Figure 13-17. ItnmDatabaseConnection event

## Tivoli Network Manager failover events

A problem event indicates that failover occurred, and the backup domain is now active

- These events are cleared upon failback when:
  - The primary domain becomes active
  - The backup domain reverts to standby

A resolution event is generated at start to clear any previous problem events that were in the ObjectServer

The **Node** field identifies the domain that raised the event:

- If the primary server goes down, a single problem event is raised
- To ensure that notification reaches the ObjectServer when failover occurs, each domain raises a problem event
- The virtual domain process raises the **ItnmFailover** event
  - This process is the last one to start
  - A resolution event is not raised at server start until all other processes are fully initialized

Figure 13-18. Tivoli Network Manager failover events

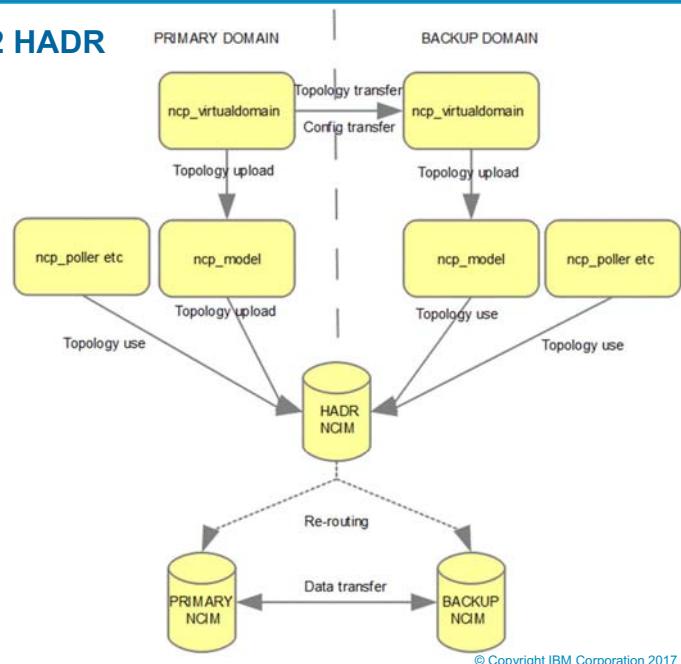
## ItnmFailoverConnection events

- A problem event indicates a loss of the TCP connection between the virtual domain processes
  - This event is cleared when the connection is reestablished
- A loss of the TCP connection between servers can be a precursor to failover
  - If the primary server goes down, the backup loses the connection
- A loss of connection between the primary and backup does not start failover
  - Loss of the connection between domains does not prevent the primary from doing its task
  - The virtual domain processes in each domain must establish a connection with the partner domain
  - Only after this connection completes, can the virtual domain process trigger a failover
- Without the TCP connection, transfer of data from the primary to the backup domain cannot be completed
  - Long-term loss of the connection is a problem
  - Data is resynchronized when the connection is reestablished

Figure 13-19. *ItnmFailoverConnection events*

## Database high availability with DB2 HADR

- Database replication relies upon DB2 HADR capabilities
- Primary and backup Tivoli Network Manager servers share the NCIM database
- The **ncp\_virtualdomain** process transfers configuration and topology information from primary server to backup server
- The MODEL service on the backup accesses NCIM but does not upload to it

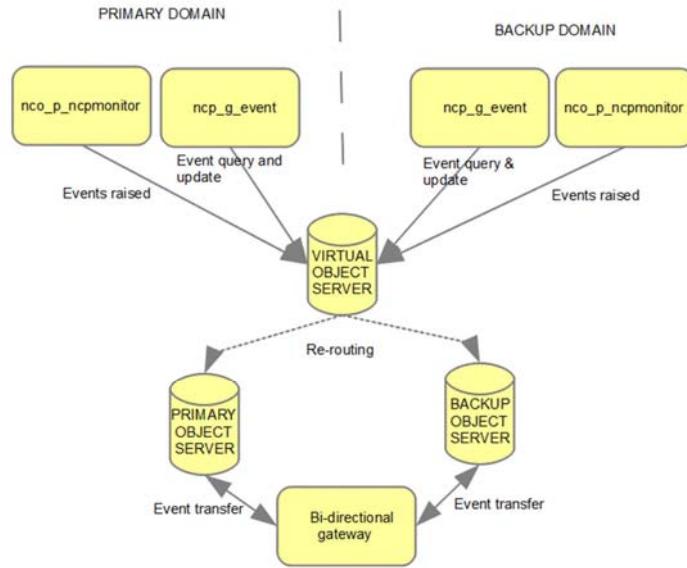


© Copyright IBM Corporation 2017

Figure 13-20. Database high availability with DB2 HADR

## ObjectServer failover

The primary and backup Tivoli Network Manager servers connect to the same virtual ObjectServer



Tivoli Network Manager failover

© Copyright IBM Corporation 2017

Figure 13-21. ObjectServer failover

## DB2 installed by Tivoli Network Manager

- The Tivoli Network Manager installer can install the bundled DB2 and OMNIbus
  - These software packages are colocated with Tivoli Network Manager as backend processes.
- If DB2 is installed with the Tivoli Network Manager installer, the DB2 NCIM database is created with:
  - A database name of **NCIM**
  - User name of **ncim**

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-22. DB2 installed by Tivoli Network Manager*

## High availability installations

- The ObjectServers and NCIM databases are typically installed on different hosts than Tivoli Network Manager in large customer environments
- This configuration requires that DB2 and OMNIbus are installed with their own installers
- The required images can be downloaded separately

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-23. High availability installations*

## Replicating failover

- In replicating failover, both IBM Tivoli Network Manager servers connect to a DB2 database configured for high availability disaster recovery (HADR)
- Tivoli Network Manager sends data to the DB2 HADR server configuration
  - Tivoli Network Manager does not replicate the information from one database server to the other
- The primary database server replicates its data to the backup database server
- Data is copied unidirectionally from the primary server to the backup server:
  - Topology data
  - Configured network polls
  - Configured network views
  - SNMP configuration
- Work with your database administrator to configure high availability

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

Figure 13-24. Replicating failover

## 13.4. Troubleshooting failover

Troubleshooting information in this section is primarily for support personnel. Other students can regard this section as reference material.

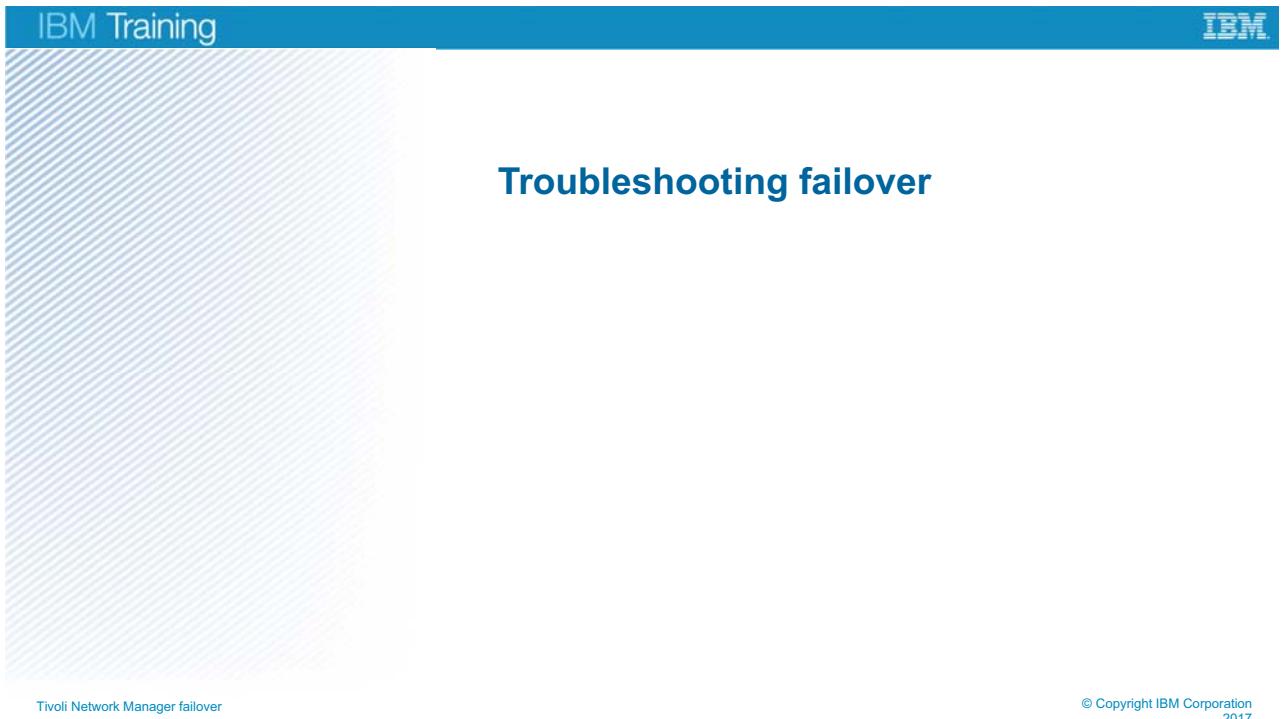


Figure 13-25. Troubleshooting failover

## Determining why failover occurred

- Which domain initiated failover?
  - Read the description of the health check problem event
  - Check the virtual domain logs
- If the primary domain initiated failover, check these items:
  - A **VirtualDomainSchema** filter (representing a local process) failed
  - Check the primary virtual domain log

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-26. Determining why failover occurred*

## No health check success message is received from the primary server

- Is the primary server down?
- Is the primary virtual domain process running?
- Do the `ConfigItnm.cfg` files match in the primary and backup domains?
  - Check both the failover and ObjectServer configuration
- Is the event gateway **Failover** plug-in enabled in both domains?
- Do the event gateway **EventFilter** and **StandbyEventFilter** allow Tivoli Network **ManagerHealthChk** events?
- Has the primary domain **nco\_p\_ncpmonitor** probe rules file been modified?
  - Check the values that are written to the **Node** and **Type** fields of the Tivoli Network **ManagerHealthChk** events

Figure 13-27. No health check success message is received from the primary server

You can monitor the IBM Tivoli Network Manager configuration with an agent.

## TCP connection problems

- Check that the entry for **ncp\_virtualdomain** in the **ServiceData.cfg** file of both domains is identical
  - Change services other than **ncp\_disco** to **DYNAMIC:NO** to force components to use same ports each time
    - Check that the ports in the **ServiceData.cfg** file are available for use on both the servers
- Check that no firewalls block the connection on the configured IP and port
- Check that the virtual domain has dependencies in **CtrlServices** on all processes that it monitors
- Check that **ncp\_config** is running

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

Figure 13-28. TCP connection problems

When you configure two Network Manager servers to work in failover mode, make sure that both servers are running the **ncp\_virtualdomain** process on the same static port.

## Virtual domain log messages

- **Failed to meet health check criterion for test: *description***
  - The test with the specified description failed in the local domain
  - A health check failure for the local domain is generated, triggering failover
- **Generating health check failure event for domain *domainName***
  - Identifies the domain that raised the problem event, triggering failover
- **Initiating failover**
  - Confirms that failover is being initiated
- **Initiating fallback**
  - Confirms that the server is failing back after an earlier failover activity
- **Service “ncp\_xyz” might be hung**
  - This message indicates that the virtual domain process failed to receive expected heartbeats for a non-exited process ncp\_xyz

Figure 13-29. Virtual domain log messages

## Troubleshooting domain issues in failover

- If the topology cache is removed from the primary domain, emptying NCIM, it must also be removed from the backup domain
  - Otherwise, the backup domain continues to use the old topology until the devices are deleted with the **LingerTime** feature
- The backup domain is read-only
  - The backup server is intended only as a temporary replacement
  - Configuration changes made on the backup domain (such as new network views) are not copied from the backup domain to the primary domain upon failback

Figure 13-30. Troubleshooting domain issues in failover

## Review questions

1. What file contains settings for the **ncp\_virtualdomain** process to be locked to identical ports on each Tivoli Network Manager server?
2. What file do you change to configure failover between servers?
3. How do you configure Tivoli Network Manager so that topology data from the DB2 NCIM database is available even when the primary server fails?

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-31. Review questions*

Write your answers here:

## Unit summary

After you completed this unit, you should be able to do the following tasks:

- Configure basic failover
- Explain what happens when failover occurs

*Figure 13-32. Unit summary*

## Review answers

1. Configure the `$NCHOME/etc/precision/ServiceData.cfg` file to lock processes to specific ports to maintain a consistent connection between the primary and backup servers.
2. Change the `$NCHOME/etc/precision/ConfigItnm.cfg` file to configure failover between servers.
3. The DB2 database administrator must configure the HADR high availability between database servers. Tivoli Network Manager does not do data replication.

Tivoli Network Manager failover

© Copyright IBM Corporation 2017

*Figure 13-33. Review answers*

# Unit 14. Installing Tivoli Network Manager

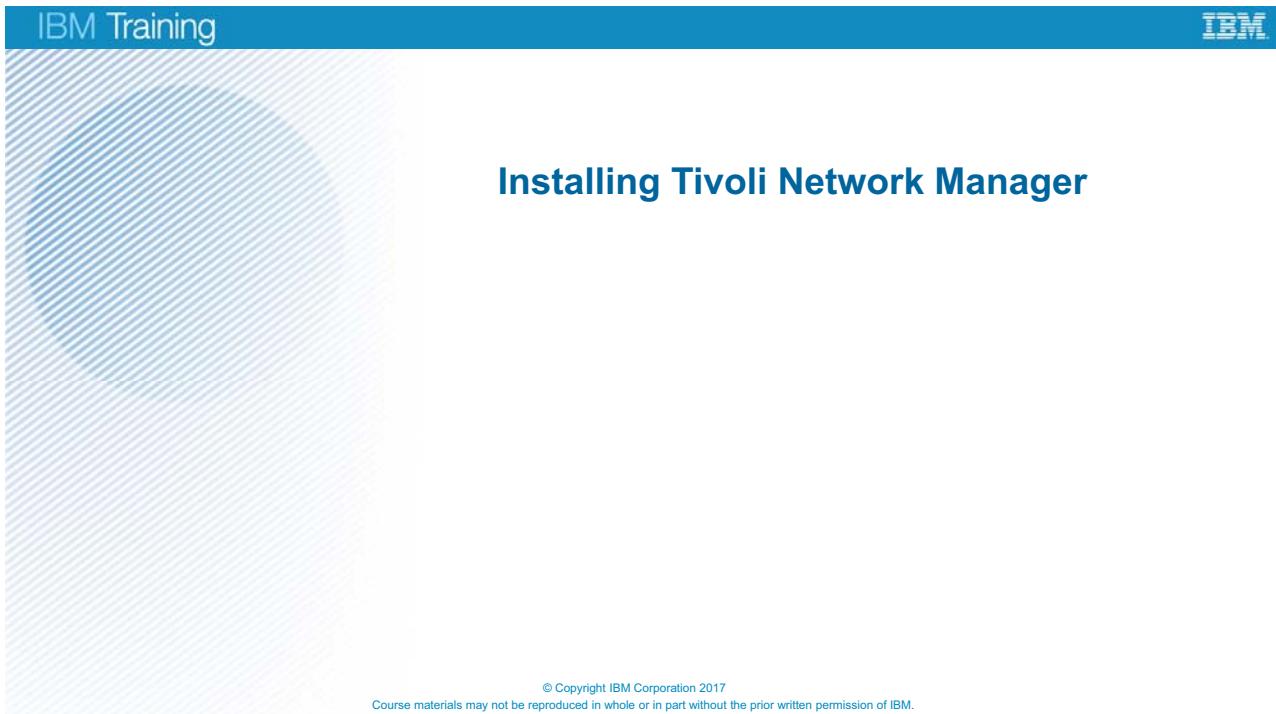


Figure 14-1. *Installing Tivoli Network Manager*

## Estimated time

01:00

## Overview

In this unit, students learn the proper method for installing Tivoli Network Manager.

## How you will check your progress

Complete review questions

IBM released several versions of Tivoli Network Manager in the last several years.

- Version 3.9 with fix pack 4 is a stable 32-bit version of software. This version does have limited ability to import information from the Alcatel EMS system and CSV files. It does not have support for 2G, 3G, or optical devices.

- Tivoli Network Manager 4.1 introduced the ability to discover optical and radio access networks (RAN) including 2G and 3G networks.
- Tivoli Network Manager 4.2 is the first 64-bit version of the product. It extends support for RAN to 4G LTE and microwave devices. Since MPLS networks can have huge amounts of discovery data in them, this 64-bit version of software is better suited to large-scale MPLS deployments.

## Unit objectives

After you complete this unit, you should be able to do the following tasks:

- Install Tivoli Network Manager in a single-server environment
- Install Tivoli Network Manager in a multiple-server environment

*Figure 14-2. Unit objectives*

# 14.1. Installation preparation

Tivoli Network Manager 4.2 has a number of installation prerequisites. The IBM Prerequisite Scanner can check for the required components and libraries. However, following the preinstallation steps in this chapter can save you a great deal of time. This section covers the tasks that must be completed before you install Tivoli Network Manager.



Figure 14-3. Installation preparation

## Supported operating systems

- On Intel and Advanced Micro Devices (AMD) x86 processors, the following versions are supported:
  - Red Hat Enterprise Linux 6 (x86-64)
  - Red Hat Enterprise Linux 7 (x86-64)
  - SuSE Linux Enterprise Server (SLES) 11.0 (x86-64) SP2 and SP3
- AIX
  - AIX 6.1 iSeries and pSeries
  - AIX 7.1 iSeries and pSeries
- Python must either be installed or it is installed as a part of the IBM Tivoli Network Manager 4.2 installation process
  - On Linux, Network Manager requires Python 2 (not Python 3), Version 2.6.6 or later
  - On AIX, Network Manager requires Python 2 (not Python 3), Version 2.7.5 or later

Figure 14-4. Supported operating systems

- Before you begin the installation, verify that the system meets hardware and operating system prerequisites.
- On Red Hat Enterprise Linux 6, you must install the 32-bit Red Hat Enterprise Linux libraries in addition to the 64-bit libraries. The default 64-bit Red Hat Enterprise Linux 6 installation does not include the 32-bit libraries so they must be downloaded and installed separately.
- Install the Jazz SM GUI before you install Tivoli Network Manager.
- Make sure that you disable Security-Enhanced Linux (**SELinux**) or set it to permissive in the SELinux configuration file before you attempt to install Network Manager. Linux systems that run Security-Enhanced Linux are not supported. Also, Linux systems that run **AppArmor** are not supported. Disable **AppArmor** to allow the installation to continue.

## Included base products and components

- Tivoli Netcool/OMNibus Versions 7.4 or 8.1 Fix Pack 5 or later
- Tivoli Netcool/OMNibus Web GUI Version 8.1 Fix Pack 4 or later
- Dashboard Application Services Hub Version 3.1.2.1 with fix pack 2 or later
- IBM WebSphere Application Server Version 8.5.5.5 or later
  - For FIPS 140-2 compliance, install WebSphere Application Server Version 8.5.5.7
- IBM WebSphere SDK Java Technology Edition Version 7
- IBM Installation Manager Version 1.8.4 or later
- Jazz for Service Management Version 1.1.2.1 or later
- Database
  - IBM DB2 Versions 10.1 Enterprise Server Edition, 10.5 Workgroup Server Edition, or 10.5 Enterprise Server Edition
  - Oracle Database Version 11 g or 12c Enterprise Edition with Partitioning option (support added in Network Manager 4.2 interim fix 2)

*Figure 14-5. Included base products and components*

Tivoli Network Manager 4.2 supports the following components:

- Reporting Services in Jazz for Service Management Version V3.1.2 FixPack 1 or later
  - Jazz for Service Management V1.1.2.1 contains Reporting Services V3.1.2.1.
  - Ensure that you have at least 17 GB of free space in the location where you want to install Reporting Services.
- IBM Tivoli Netcool Configuration Manager Version 6.4.2 or later
- IBM Tivoli Monitoring Version 6.3 Fix Pack 2 or later
  - The Agent itself (IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition) runs only on Linux and AIX. Windows is not a supported platform for Network Manager 4.2.



### Note

Note: IBM Tivoli Monitoring Version 6.3 Fix Pack 2 is included in the Network Manager V4.2 package.

- 
- IBM Tivoli Business Service Manager 6.1.1
  - IBM Tivoli Application Dependency Discovery Manager and IBM Tivoli Change and Configuration Management Database Version 7.2.1 or later

## Library requirements for Linux

| Library                              | Red Hat package                                                   | SuSE package                    | Required by component  |
|--------------------------------------|-------------------------------------------------------------------|---------------------------------|------------------------|
| <code>libstdc++.so.6 (64-bit)</code> | <code>libstdc++-4.4.4-13+</code><br><code>libstdc++-4.8.2+</code> | <code>libstdc++46-4.6.1+</code> | Network Manager Core   |
| <code>libpam.so.0 (64-bit)</code>    | <code>pam-1.1.1</code>                                            | <code>pam-64bit</code>          | Tivoli Netcool/OMNibus |

Figure 14-6. Library requirements for Linux

## Supported browsers for GUI

- Internet Explorer 10 or 11
- Mozilla Firefox 38 Extended Support Release

*Figure 14-7. Supported browsers for GUI*

## Processors, memory, and bandwidth requirements

- Processors
  - Deployments of medium or large customer networks require four processors
  - 64-bit processors only
- Memory
  - Single-server deployments (basic installation): 16 GB RAM minimum (for single-server deployments where web applications, topology database, and OMNIbus are all installed on the same server)
- Network bandwidth
  - Place server in the data center with LAN speed connections of 100 Mbps fast Ethernet or Gigabit Ethernet to the DNS system and the core network devices to be discovered and managed

*Figure 14-8. Processors, memory, and bandwidth requirements*

When you select the amount of memory and processor to put on a Tivoli Network Manager server, consider these factors:

- Does this installation use a single-server deployment or a distributed deployment?
- What are the size and complexity of your network?
  - Small demonstration or educational system deployment
  - Small customer network
  - Medium customer network
  - Telecommunications company or service provider network
  - Large customer network
  - Very large customer network
- Are you installing a failover server?
- What is the number of operations staff to require system access to the IBM Tivoli Network Manager server?



### Note

For multiple core processors, individual core speed can be more important than the number of cores. You can use any speed processor. However, selecting the fastest core speed and largest on-chip cache significantly improves processing of information about large networks.

## Disk space requirements

Approximately 104 GB disk space needed

- At least 1 GB space in the `/var` directory and in the `/tmp` directory
- 10 GB of space in the data location for IBM Installation Manager
  - This location is where packages are unpacked during installation and where packages are stored for rolling back
- At least 350 MB of space in the installing user home directory
- For a single-server deployment, where the Network Manager core components are, you need adequate disk space
  - 5 GB hard disk space to store the software
  - 4 GB hard disk space per domain for cache storage
  - Log file space: Assuming that each log file is 1 GB and six processes are set to full debug level, you would require 24 GB of disk
  - 50 GB free disk space to run Network Manager

Figure 14-9. Disk space requirements

## Disable SELinux and firewall before installation

To disable SELinux, turn off SELinux enforcing by completing the following steps:

- Open `/etc/sysconfig/selinux`
- Edit the file:
  - Find line that says `SELINUX=enforcing`
  - Change it to read `SELINUX=disabled`
- Restart the server
- Disable firewall



© Copyright IBM Corporation 2017

*Figure 14-10. Disable SELinux and firewall before installation*

## Create essential users

Create the **ncoadmin** user and group if you are installing OMNibus on this same server



Figure 14-11. Create essential users

Creating the **ncoadmin** group and user prevents errors when you run the script that configures the database for non-root users.

## Prepare the operating system

- Make certain that the server is running the available patches that are specified at:
  - <http://tinyurl.com/kuuxovi> (4.1.1)
  - <http://tinyurl.com/lem9wsy> (4.2)
- Some 32-bit libraries are prerequisite for installation
- Edit `/etc/yum.conf` and add `multilib_policy=all` to the file so that `yum` installs both 32-bit and 64-bit libraries
- As the `root` user, create a directory  
`mkdir /opt/IBM`  
`chown -R netcool:netcool IBM`
- Enable multicast
  - Put a statement in `/etc/sysconfig/network-scripts/route-eth0` that says: `224.0.0.0/4 dev eth0`

```
[student@itnm411 network-scripts]$ netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.191.101.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
192.168.132.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
224.0.0.0 0.0.0.0 240.0.0.0 U 0 0 0 eth0
```

© Copyright IBM Corporation 2017

Figure 14-12. Prepare the operating system

Linux has several ways to enable multicasting. In this example, the `/etc/sysconfig/network-scripts/route-eth0` file contains a single statement:

```
224.0.0.0/4 dev eth0
```

After that file change is made, either reboot the server or restart networking. Run the `netstat -rn` command to verify that you see a route to the 224.0.0.0/4 or other multicast subnet. Without multicast enabled, not all Tivoli Network Manager processes start. The log files have errors that indicate that multicast is not working.

## Setting file limits for the non-root user

- The **root** user must set appropriate hard and soft limits for the non-root user who installs the software
- Set the maximum number of open files to a minimum of 8192
- As the root user, do the following tasks:
  - Edit `/etc/sysctl.conf` to include the line `fs.file-max=8192` (can set to 16384)
  - Edit `/etc/security/limits.conf`
  - Below the commented line, add the parameters to set limits

```
#<domain> <type> <item> <value>
Add the following lines:
* soft nproc 16384
* hard nproc 16384
* soft nofile 8192
* hard nofile 8192
```
  - You can use higher values than these example values
- Log out of the session and log in as the non-root user
  - View the current setting with the `ulimit -n` command and confirm that it is 8192

```
[netcool@localhost ~]$ ulimit -n
8192
```

Figure 14-13. Setting file limits for the non-root user

Set the maximum number of files to a value of **8192** or a higher value.

## Verify write permissions

- Verify that the non-root user has write permissions to the directory where software is going to be installed
- The default path is `/opt/IBM/tivoli/netcool`
- If you are installing as a non-root user, set permissions with `chown` on the `/opt/IBM` directory to enable the non-root user to create the necessary subdirectories and copy files into it

```
cd /opt
```

```
mkdir IBM
```

```
chown -R student:student IBM (or other user that runs the ncp processes)
```

```
drwxr-xr-x 3 student student 4096 Nov 17 19:00 IBM
```

Figure 14-14. Verify write permissions

## Install required software components

The following software components must already be installed **before** you install Tivoli Network Manager:

- Netcool/OMNIbus
- DB2 or Oracle
- Tivoli Common Reporter database
- JazzSM

*Figure 14-15. Install required software components*

## 14.2. Installing Tivoli Network Manager

As soon as the prerequisites are met, the IBM Installation Manager makes installing Tivoli Network Manager 4.2 easy. This section shows the screens that you see when you install Tivoli Network Manager so that you know what to expect.



Installing Tivoli Network Manager

© Copyright IBM Corporation 2017

Figure 14-16. *Installing Tivoli Network Manager*

## Run the prerequisite scanner

- Download the Tivoli Network Manager configuration file for the prerequisite scanner
  - <http://www.ibm.com/support/docview.wss?uid=swg21970155>
  - **TNM\_04200000.cfg**
- Download the Prerequisite scanner from <http://tinyurl.com/zxy7yma>
- On the Tivoli Network Manager Core server, set the environment variable **tnmCORE=True**
- Run the prerequisite scanner

```
su - netcool
export tnmCORE=True
./prereq_checker.sh "TNM 04200000" details
```
- Resolve any issues that fail the prerequisite check
- Run the prerequisite scanner again until items pass the prerequisite check

Figure 14-17. Run the prerequisite scanner

## Configure the NCIM database

- Extract the `db2_creation_scripts.tar.gz` into a temporary folder on the DB2 database server

```
tar xzvf /mnt/ITSO_SHARE/ITNM/Base/db2_creation_scripts.tar.gz
```

- As the **root** user, create the **ncim** user on the operating system

```
useradd ncim -g db2iadm1
passwd ncim (netcool)
```

- As **db2inst1** create the database

```
su - db2inst1
./create_db2_database.sh NCIM ncim
```

```
DB20000I The CREATE DATABASE command completed successfully.
```

```
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 10.5.3
```

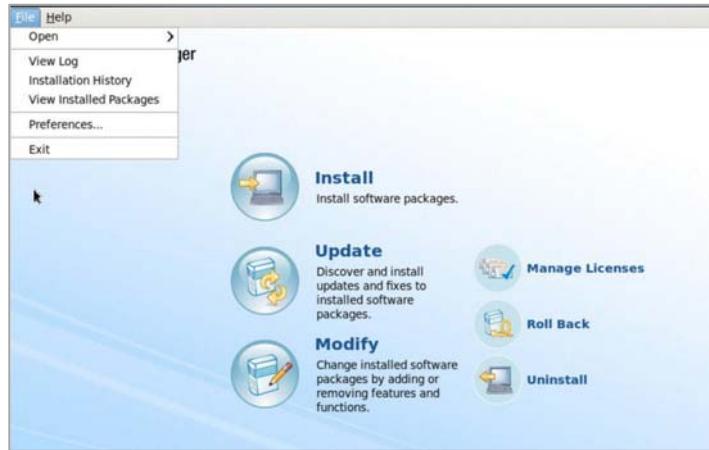
Figure 14-18. Configure the NCIM database

## Using the Installation Manager

1. Run the Installation Manager

```
cd /opt/IBM/netcool/IM/InstallationManager/eclipse
./IBMMIM
```

2. Select File > Preferences



Installing Tivoli Network Manager

© Copyright IBM Corporation 2017

Figure 14-19. Using the Installation Manager

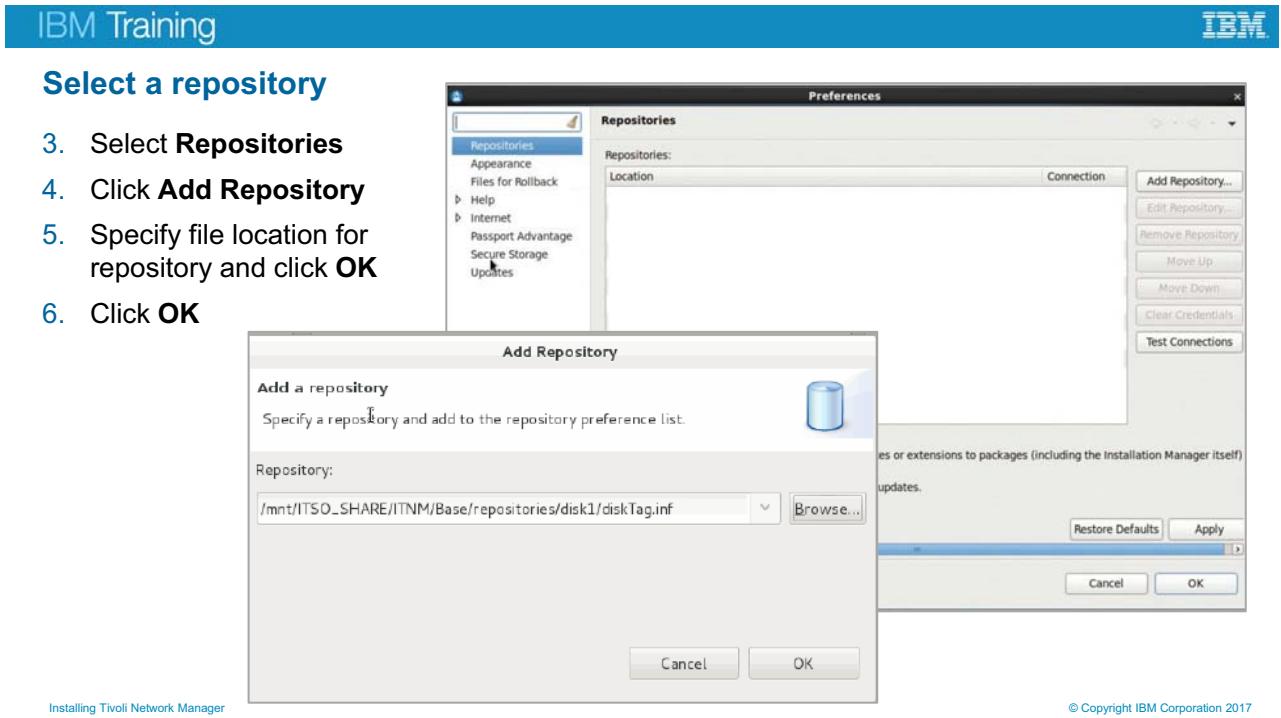


Figure 14-20. Select a repository

## Install the Network Manager core components

7. Select the **Network Manager Core Components**

8. Click **Next**

9. Accept the license agreement

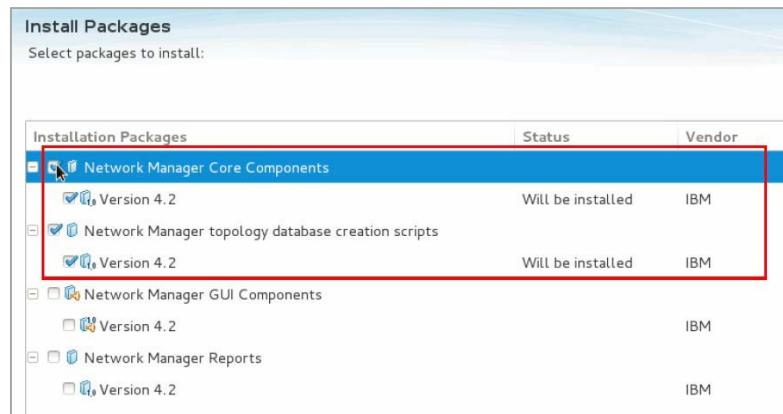


Figure 14-21. Install the Network Manager core components

**IBM Training**

**Start the installation**

**10. Accept the default value for the shared resources directory**

**11. Click Next**

**12. Confirm the installation directory**

**13. Click Install**

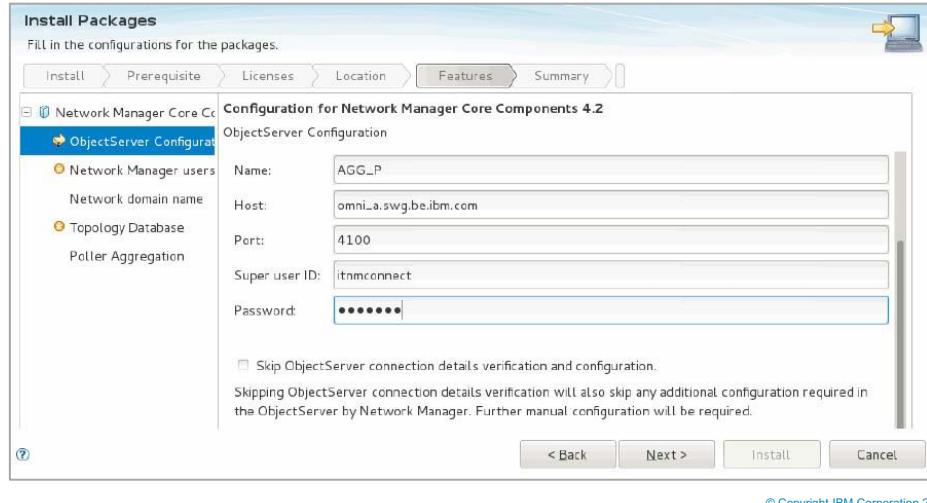
© Copyright IBM Corporation 2017

Figure 14-22. Start the installation

## Configure ObjectServer options

14. Configure ObjectServer information

15. Click **Next**



Installing Tivoli Network Manager

© Copyright IBM Corporation 2017

*Figure 14-23. Configure ObjectServer options*

## Select passwords

### 16. Choose passwords for **itnmadmin** and **itnmuser**



Installing Tivoli Network Manager

© Copyright IBM Corporation 2017

Figure 14-24. Select passwords

## Specify Tivoli Network Manager domain name

### 17. Specify the network domain name

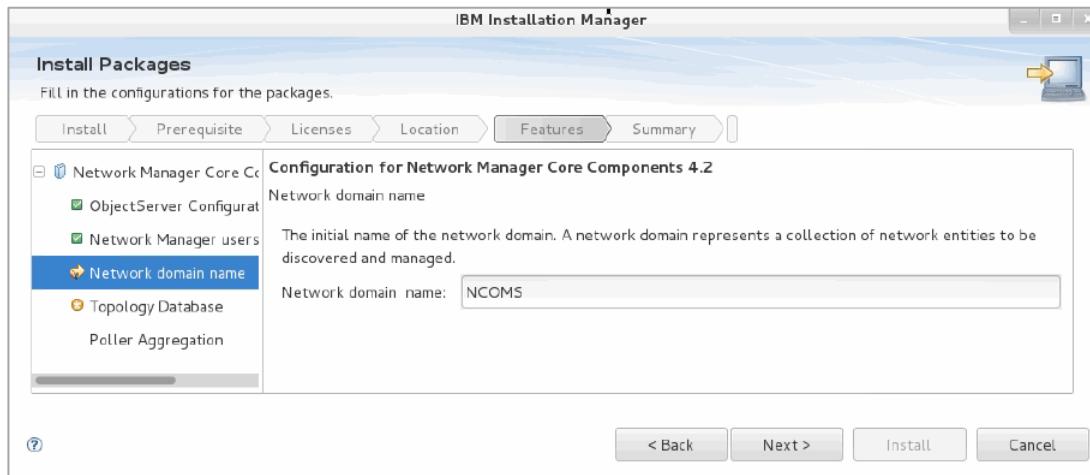
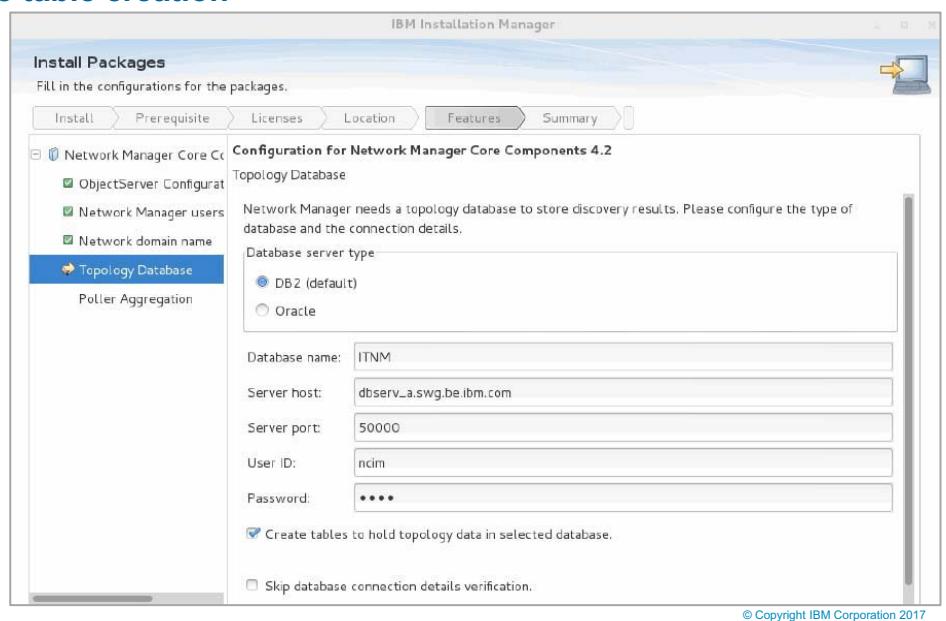


Figure 14-25. Specify Tivoli Network Manager domain name

## Configure database table creation

18. Make sure that DB2 or Oracle server is running
19. Configure database table creation and click **Next**



Installing Tivoli Network Manager

© Copyright IBM Corporation 2017

Figure 14-26. Configure database table creation

## Confirm installation directory for Python

20. Confirm Python installation directory and click **Next**

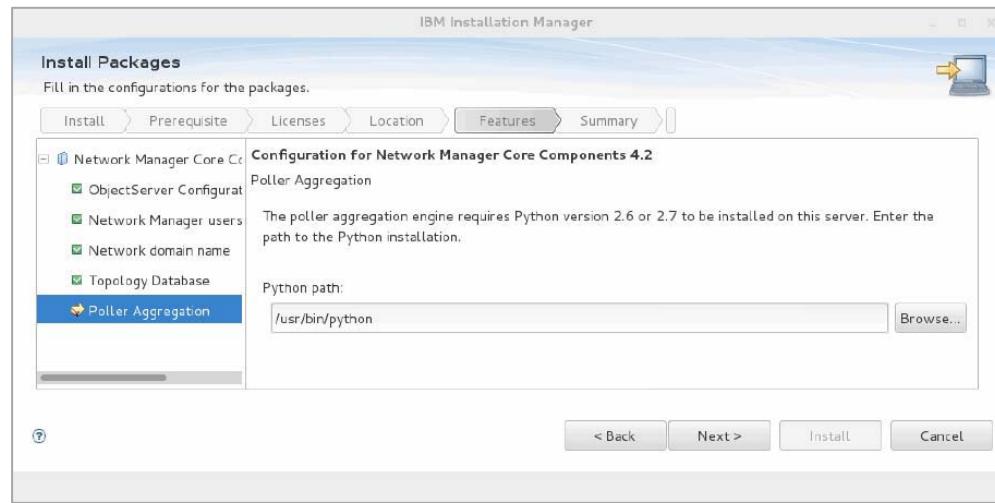


Figure 14-27. Confirm installation directory for Python

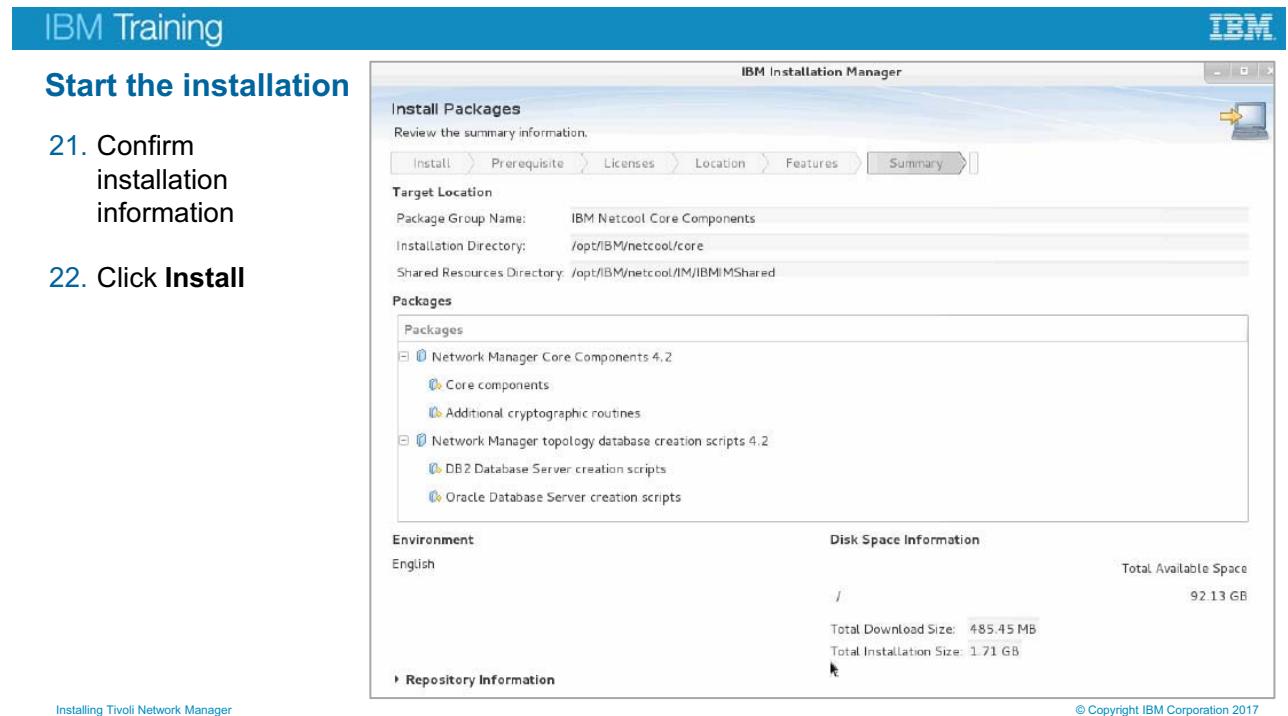


Figure 14-28. Start the installation

## 14.3. Component communications

Understanding how Tivoli Network Manager 4.2 components communicate can help you to resolve problems in multi-server deployments.

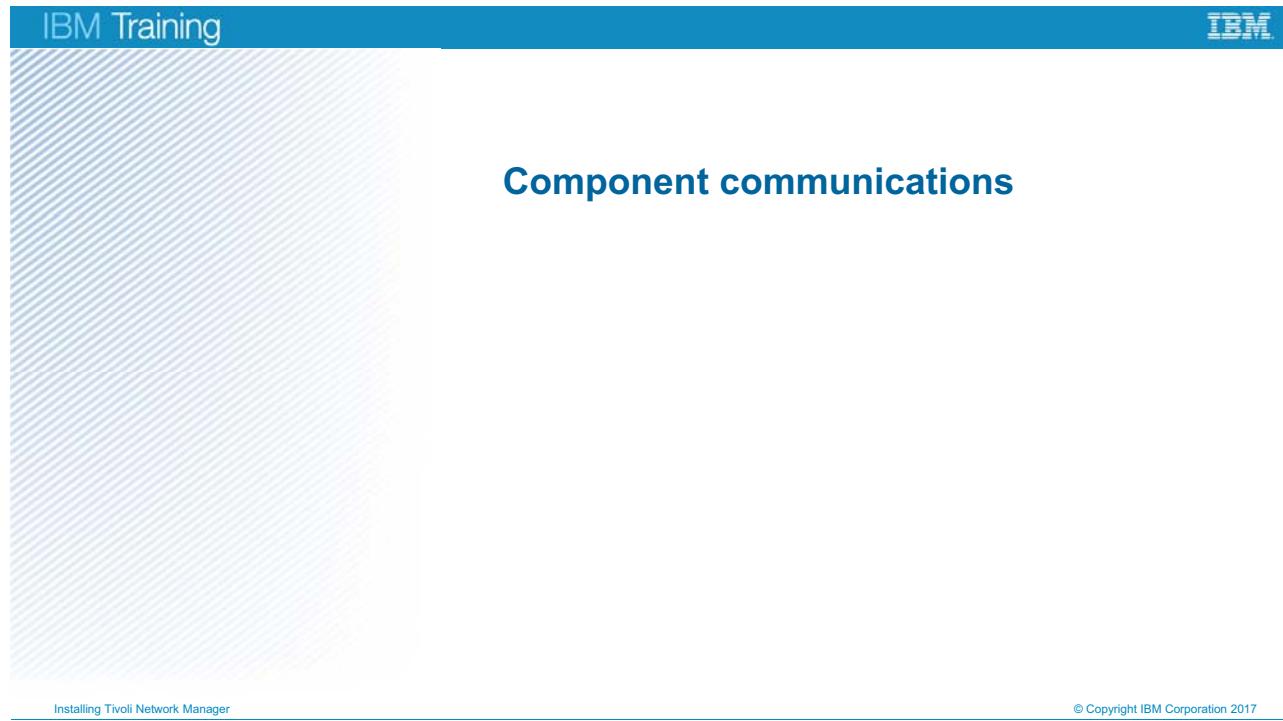


Figure 14-29. Component communications

## Core component communication configuration (1 of 2)

Some core Tivoli Network Manager services create socket (IP address + available port) connections during startup

IP address and port data are written to the `$NCHOME/etc/precision/ServiceData.cfg` file

- The socket assignment for **ncp\_config** is set to **DYNAMIC: NO** by default
  - The **ncp\_model**, **ncp\_store**, and **ncp\_class** processes depend on **ncp\_config**
  - If the **ncp\_config** entry is set to an unavailable IP address or port, these processes do not start
- Processes set to **DYNAMIC: YES**
  - Tries to use the same IP address and port in **ServiceData.cfg**
  - If that socket is not available, the process dynamically creates a socket connection on the first Ethernet interface (typically, **eth0**)

Figure 14-30. Core component communication configuration (1 of 2)

Several core Tivoli Network Manager services create socket (IP address + available port) connections during startup for their communications.

IP address and port data are written to the `$NCHOME/etc/precision/ServiceData.cfg` file.

The socket assignment for **ncp\_config** is set to **DYNAMIC: NO** by default. Other processes depend on **ncp\_config** to be running: **ncp\_model**, **ncp\_store**, and **ncp\_class**. If the entry for **ncp\_config** is set to an unavailable IP address or port, these dependent processes do not start.

Other processes are set to **DYNAMIC: YES** by default. The next time that the process starts, it attempts to use the same IP address and port configuration that is described in **ServiceData.cfg**. If it cannot start on that socket, it dynamically creates a socket with the IP address on the first Ethernet interface (typically, **eth0**).

## Core component communication configuration (2 of 2)

Configure a static communications socket when you are doing the following tasks:

- Making a portable VMware image
- Configuring processes to communicate through a firewall
- Configuring a failover IBM Tivoli Network Manager server
  - Certain processes must operate on a static socket configuration, ensuring that each server always knows how to communicate to the other server in the failover pair

The IP address in `ServiceData.cfg` must match the actual location of the device that is running the binary

- If you move a portable image of a Tivoli Network Manager server, make sure that the IP addresses in `ServiceData.cfg` reflect the IP address of the new location that is hosting the image

*Figure 14-31. Core component communication configuration (2 of 2)*

This file can require modification under the following circumstances:

- Some customers choose to install Tivoli Network Manager to run as a virtual server. This platform choice can make it easier to deploy multiple servers. If you move a Tivoli Network Manager image to a new location, you must update the IP addresses in the applicable `ServiceData.cfg` file.
- Make the socket connections for Tivoli Network Manager processes static to reduce any need to reconfigure a firewall.
- If you configure Tivoli Network Manager for failover, the entry for the `ncp_virtualdomain` process must be configured for a static host name and port combination. This static entry ensures that both Tivoli Network Manager servers always use the same port and socket to communicate with one another.

## ServiceData.cfg example

The Server data file: Contains information about servers and the general multicast address to use

```
SERVICE: Multicast Service DOMAIN: ANY_PRECISION_DOMAIN ADDRESS: 225.13.13.13 PORT: 33000
```

```
SERVICE: ncp_config DOMAIN: itnm4go ADDRESS: 10.191.101.46 PORT: 7968 SERVERNAME: itnm4go DYNAMIC: NO
```

```
SERVICE: Helper DOMAIN: itnm4go ADDRESS: 10.191.101.46 PORT: 58142 SERVERNAME: itnm4go DYNAMIC: NO
```

**Port value does not change (static)**

```
SERVICE: ncp_disco.4173 DOMAIN: itnm4go ADDRESS: 192.168.1.111 PORT: 56682 SERVERNAME: itnm4go DYNAMIC: YES
```

**Port configuration value can change (dynamic)**

Figure 14-32. ServiceData.cfg example



### Important

If you are running servers in failover mode, the **ncp\_virtualdomain** process must be configured to a fixed socket (**DYNAMIC: NO**) on both servers. This configuration assures that the partner processes on each server can establish a connection with each other.



### Hint

Several processes are dependent upon **ncp\_config**. If you move an installation to another computer, make sure that the entry for **ncp\_config** has a correct available port and IP address.

## Uninstalling Tivoli Network Manager 4.2

- Use the installation manager to uninstall Tivoli Network Manager 4.2
- Do not attempt to remove Tivoli Network Manager by just deleting directories
- For more information on installing Tivoli Network Manager 4.2 and related products, see:  
<http://tinyurl.com/hpd2svj>



Figure 14-33. Uninstalling Tivoli Network Manager 4.2

## 14.4. Postinstallation tasks

Some Tivoli Network Manager discovery processes need to run as a root user equivalent. Therefore, you need to do some postinstallation tasks to ensure that these components can run properly.

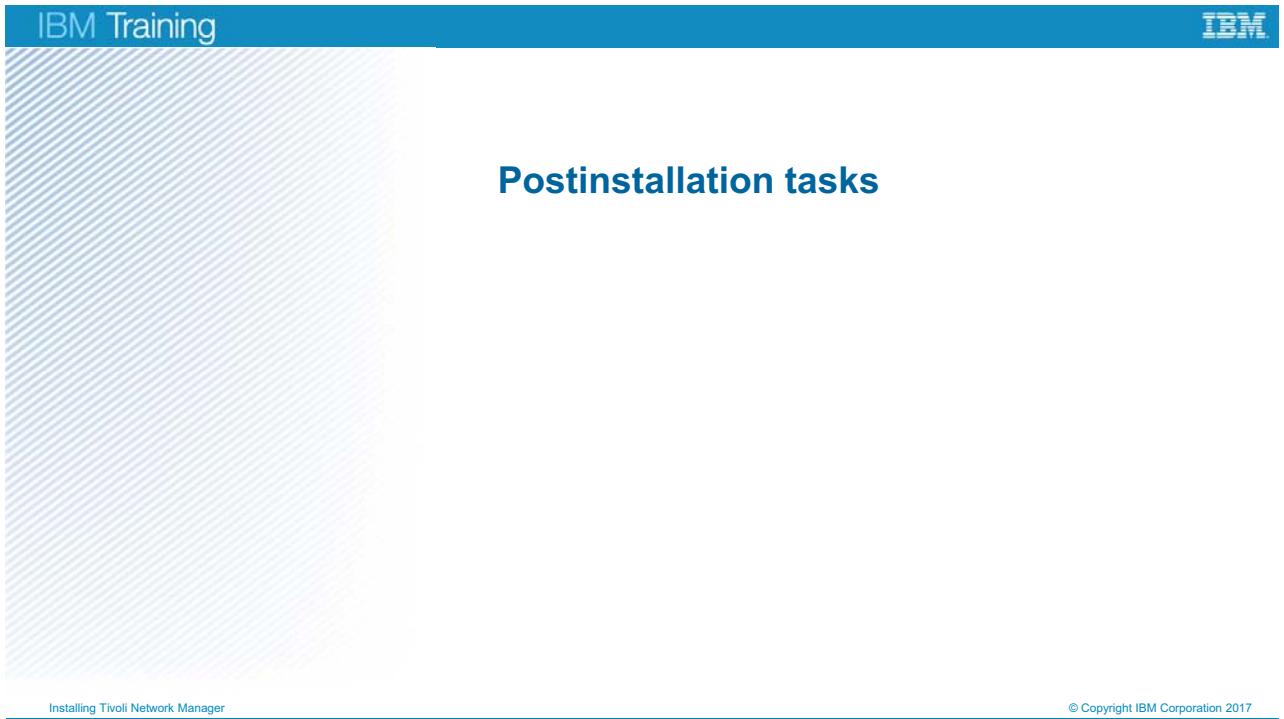


Figure 14-34. Postinstallation tasks

## Set processes to run as non-root

### 1. Set environment variables

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

### 2. Run scripts to set Tivoli Network Manager processes to run as non-root

```
su - root
cd /opt/IBM/netcool/core/precision/scripts/
. ./setup_run_as_setuid_root.sh
```

### 3. Set Tivoli Network Manager to automatically start when the system restarts

```
cd /opt/IBM/netcool/core/precision/install/scripts
. ./create_all_control.sh -auto_only
```

Figure 14-35. Set processes to run as non-root

## Verify that OMNIbus processes are run as non-root

Check `$OMNIHOME/etc/nco_pa.conf` to make sure that the ObjectServer and probes are not set to **run as 0** (the root user) but as the non-root user instead:

```
nco_process 'MasterObjectServer'
{
 Command '$OMNIHOME/bin/nco_objserv -name NCMS -pa NCO_PA' run as 1001
 Host = 'localhost'
 Managed = True
 RestartMsg = '${NAME} running as ${EUID} has been restored on ${HOST}.'
 AlertMsg = '${NAME} running as ${EUID} has died on ${HOST}.'
 RetryCount = 0
 ProcessType = PaPA_AWARE
}
```

*Figure 14-36. Verify that OMNIbus processes are run as non-root*

In this example, the **Netcool** user is user number 1001 as seen in the `/etc/passwd` file. To make the ObjectServer and other processes run as non-root, change the run as value to the number of the appropriate non-root user.

The **nco\_pad** process is the process agent daemon and must be run as **root**. However, all other OMNIbus processes can run as non-root processes.

## Configure non-root running of nco\_p\_mttrapd

- As **root**, change the owner of the probe binary

```
cd $OMNIHOME/probes/arch (such as $OMNIHOME/probes/linux2x86)
```

```
chown root nco_p_mttrapd
```

- As **root**, enable the probe binary to run as **setuid** root

```
chmod +s nco_p_mttrapd
```

- As **root**, edit the file **/etc/ld.so.conf** adding the following line to the end of the file:

```
/opt/IBM/tivoli/netcool/omnibus/platform/linux2x86/lib
```

- As **root**, run the following command:

```
ldconfig -v
```

- As the non-root user, run the probe from the **\$OMNIHOME/probes** directory

*Figure 14-37. Configure non-root running of nco\_p\_mttrapd*

Because the SNMP (mttrapd) probe uses port 162, it does not start as a non-root user. Port 162 is a system port and requires the probe to run as root or with SUID root access.

An entry in **/etc/ld.so.conf** for **/opt/IBM/tivoli/netcool/platform/<arch>/lib** was created by running the non-root installation script.

## 14.5. Process control

Just as Netcool/OMNibus has a process control to keep its components running, Tivoli Network Manager has the ncp\_ctrl process that starts processes and then monitors them. The processes and their startup parameters are specified in the `CtrlServices.cfg` file. If a process fails or is stopped, the CTRL service attempts to restart the process.

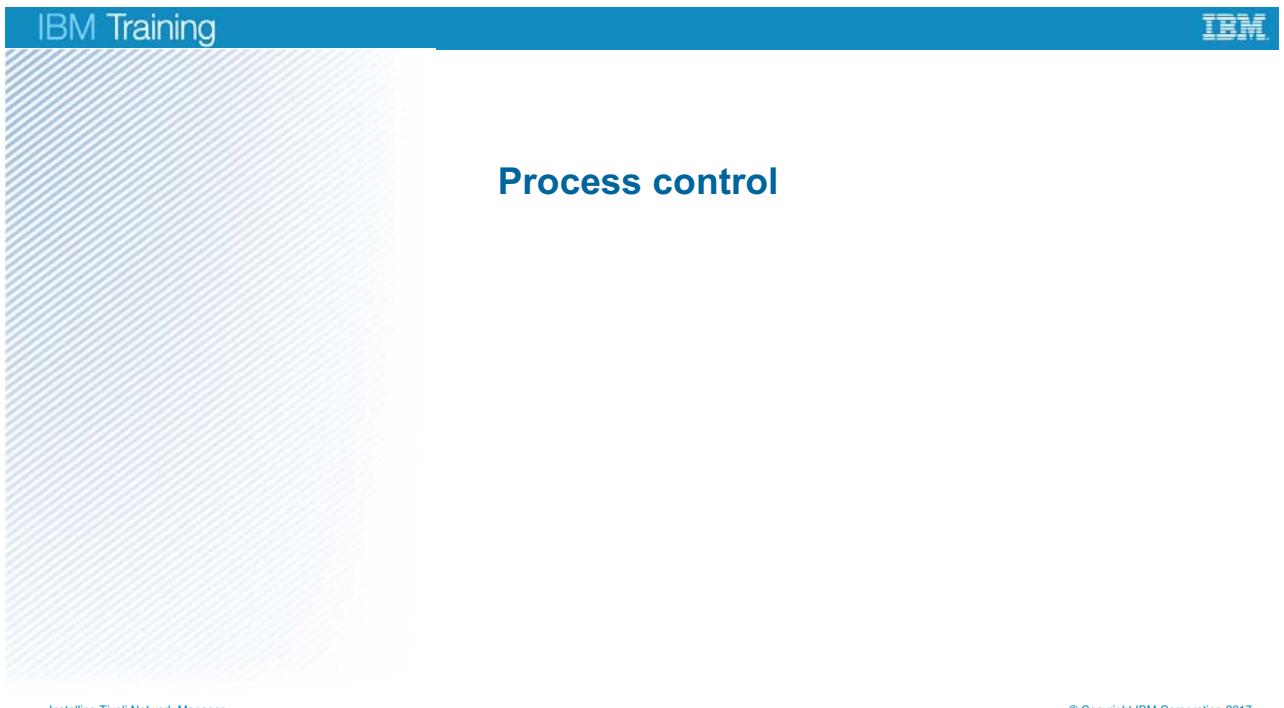


Figure 14-38. Process control

## CtrlServices.cfg file

- A domain-specific version of `$NCHOME/etc/precision/CtrlServices.cfg` is created when:
  - Installation of Tivoli Network Manager completes
  - A new domain is added
- This file specifies the process control for Tivoli Network Manager
  - Individual processes are enabled or disabled in this file
  - The **latency** parameter specifies the amount of time one process waits for a response from another process before it generates a timeout failure message
- After you edit this file, restart Tivoli Network Manager to get it to reread the file

Figure 14-39. CtrlServices.cfg file

## A managed process entry

- Entries into `services.inTray` indicate processes that are monitored and restarted as necessary by `ncp_ctrl`

```

insert into services.inTray
(
 serviceName,
 binaryName,
 servicePath,
 domainName, ← Indicates managed service
 argList,
 dependsOn,
 retryCount
)
values
(
 "ncp_model",
 "ncp_model",
 "$PRECISION_HOME/platform/$PLATFORM/bin",
 "$PRECISION_DOMAIN",
 ["-domain", "$PRECISION_DOMAIN", "-latency", "100000", "-debug", "0", "-messagelevel", "warn"],
 ["ncp_config", "ncp_store", "ncp_class"],
 5 ← Value populated automatically by environment variable
);

```

Command-line arguments

ncp\_model will not start until these processes are started

- Latency in this entry is 100 seconds (specified in milliseconds)
- The `dependsOn` parameter specifies processes that must be running before the specified service can start

*Figure 14-40. A managed process entry*

The `retryCount` of 5 for this entry specifies that `ncp_ctrl` makes five attempts to start this process. If the process does not start within five attempts, it logs an error and then quits. If that happens, you can fix the problem that prevents the process from starting, and do one of the following actions:

- Start the process manually
- Stop Tivoli Network Manager and restart

Many processes do not have a `dependsOn` parameter.

## An unmanaged process entry

- The **ncp\_ctrl** process starts entries in the **services.unManaged** table, but it does not monitor or restart these processes
- These processes are unmanaged for one of the following reasons:
  - The process is for a third-party product
  - The entry starts scripts that forks into multiple processes
  - Network Manager 4.2 does not have any default services that are unmanaged
- The **servicePath** variable points to the directory where the script or executable file is located
- The following example shows a service from an earlier version of Tivoli Network Manager where an unmanaged process is started

```
insert into services.unManaged
(
 serviceName,
 servicePath,
 argList
)
values
(
 'informix_control.sh',
 '/opt/IBM/tivoli/netcool/precision/bin',
 ['start']
);
```

*Figure 14-41. An unmanaged process entry*

## Discovery startup in CtrlServices.cfg

- The `ncp_disco` process takes a `-discoOnStartup` parameter
- This value specifies whether to have a new discovery start automatically when DISCO starts up
  - If set to **0**, discovery does not start automatically when DISCO starts (default value)
  - If set to **1**, discovery starts automatically when DISCO starts

Figure 14-42. Discovery startup in CtrlServices.cfg

## 14.6. Starting Tivoli Network Manager

Though Tivoli Network Manager can be started in a number of ways, the simplest way is to use the `itnm_start` script.



Figure 14-43. Starting Tivoli Network Manager

## Run the `itnm_start` command

- To start manually:
  - `itnm_start -domain domainName`
  - `itnm_start -domain ITNM4GO`
  - If you do not specify a domain name, the `itnm_start` command uses the value of `$PRECISION_DOMAIN` that was created during the installation
- Use the `itnm_status` command as the non-root user to verify that processes are running

Figure 14-44. Run the `itnm_start` command

## Review questions

1. In what file can you specify socket connections for Tivoli Network Manager processes?
2. Your non-root installation of Tivoli Network Manager is not starting correctly. Someone previously logged in as root and ran the software successfully. What is a likely remedy to make the software successfully run as a non-root user again?

*Figure 14-45. Review questions*

Write your answers here:

## Unit summary

After you completed this unit, you can do the following tasks:

- Install Tivoli Network Manager in a single-server environment
- Install Tivoli Network Manager in a multiple-server environment

*Figure 14-46. Unit summary*

## Review answers

1. The `ServiceData.cfg` file specifies the socket connections for Tivoli Network Manager processes.
2. Have the `root` user delete the log files from the `$NCHOME/log/precision` directory and then restart Tivoli Network Manager as a non-root user.

Figure 14-47. Review answers

# Unit 15. Tivoli Network Manager reports



Figure 15-1. Tivoli Network Manager reports

## Estimated time

00:45

## Overview

In this unit, students learn how to access on-demand reports containing information on discovered network devices and event data. Students also learn how to schedule reports.

## How you will check your progress

- Review questions
- Hands-on exercises

## References

<http://tinyurl.com/262m3q9>

## Unit objectives

- View an asset report for IBM Tivoli Network Manager
- Create the same report in several different report formats
- Create a report snapshot
- Schedule a report snapshot

Figure 15-2. Unit objectives

## 15.1. Overview

Reports that are based on asset and event information provide powerful insights for planning and troubleshooting. Tivoli Common Reporting provides the engine for report scheduling and generation. When you install Tivoli Network Manager, you can also install preconfigured reports that show discovery data and event data.

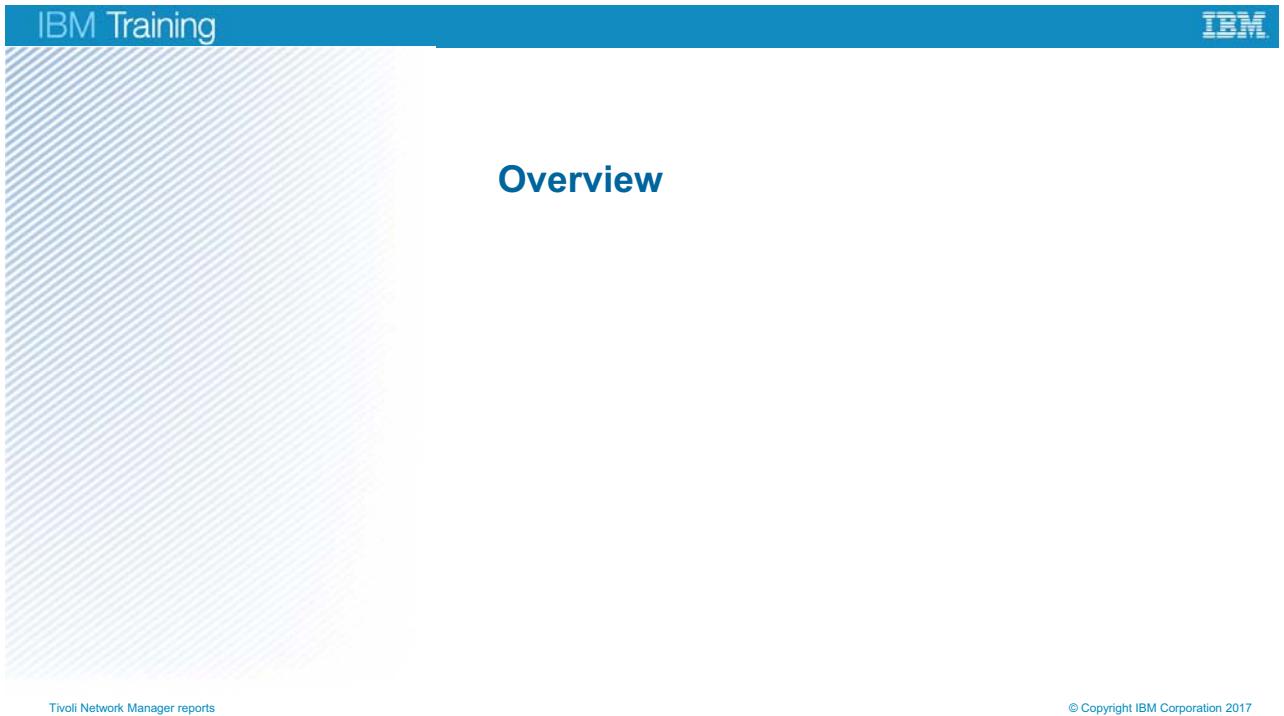


Figure 15-3. Overview

## Tivoli Common Reporting for Network Manager

- Reports in IBM Tivoli Network Manager 4.2 were created with Cognos
- Reports that are created with Business Intelligence Reporting Tool (BIRT) are eliminated in version 4.2
- Generate reports on demand in the Tivoli Common Reporting viewer
- Schedule reports to run daily
- Customize reports with Cognos Report Studio
- Cognos Viewer can show the report as a PDF, HTML, Excel, CSV, or XML file
- The Tivoli Common Reporting installation is separate from the Tivoli Network Manager installation

The screenshot shows a software interface titled 'Work with reports'. At the top, there's a toolbar with various icons. Below the toolbar, the title bar says 'tipadmin' and 'Public Folders'. The main area is a table listing report folders under 'Public Folders > Network Manager'. The columns are 'Name', 'Modified', and 'Actions'. The data in the table is as follows:

| Name                       | Modified                    | Actions |
|----------------------------|-----------------------------|---------|
| Asset Reports              | January 15, 2011 5:57:00 PM | More... |
| Current Status Reports     | January 15, 2011 5:57:02 PM | More... |
| Monitoring Reports         | January 24, 2011 3:17:11 PM | More... |
| Network Technology Reports | January 24, 2011 3:17:28 PM | More... |
| Network Views Reports      | January 15, 2011 5:57:03 PM | More... |
| Path Views Reports         | January 15, 2011 5:57:07 PM | More... |
| Performance Reports        | January 15, 2011 5:57:10 PM | More... |
| Summary Reports            | January 24, 2011 3:17:32 PM | More... |
| TroubleShooting Reports    | January 15, 2011 5:57:11 PM | More... |
| Utility Reports            | January 24, 2011 3:17:09 PM | More... |

Tivoli Network Manager reports

© Copyright IBM Corporation 2017

Figure 15-4. Tivoli Common Reporting for Network Manager

## Reports overview (1 of 2)

- Tivoli Common Reporting incorporates the full IBM Cognos 8.4 Business Intelligence (BI) Reporting and Modeling solution that includes these features:
  - Web-based report creators and editors, administration tools, navigation tools, and viewers
  - Framework manager utility that is used to extend provided product models
  - Industry-leading reporting suite to Tivoli customers at no additional charge
- Tivoli Common Reporting does not include Cognos BI 8.4 server or other BI extensions

Figure 15-5. Reports overview (1 of 2)

## Reports overview (2 of 2)

Currently, over 60 Tivoli Network Manager reports are available

Cognos is the reporting platform for Tivoli Network Manager reports

- Provides a graphical report configuration tool for users to quickly create and edit reports
- Cognos data sources are not JDBC-based and are harder to configure
- Cognos is used in Tivoli Network Manager for data abstraction
  - Report features are also included, such as simplified report editing, custom reporting, report scheduling, and report distribution by email
  - Cognos can email a link to a report or attach a PDF file to the email
- Reports can be viewed in multiple formats

Figure 15-6. Reports overview (2 of 2)

## IBM Training



## Tivoli Common Reporting menu options

© Copyright IBM Corporation 2017

- Create new report set and report
- Schedule reports
- Save default parameters values on schedule reports
- Copy and paste a report
- Start Report Studio to edit a report
- See report properties
- Define preferences like locale

Figure 15-7. Tivoli Common Reporting menu options

## Default report sets

The screenshot shows a file browser window titled "Public Folders > Network Manager". The left pane displays a list of report sets under the "Name" column, which is sorted by name. Each item is preceded by a small folder icon. The report sets listed are:

- Asset Reports
- Current Status Reports
- Monitoring Reports
- Network Technology Reports
- Network Views Reports
- Path Views Reports
- Performance Reports
- Summary Reports
- TroubleShooting Reports
- Utility Reports

Tivoli Network Manager reports

© Copyright IBM Corporation 2017

*Figure 15-8. Default report sets*

Default report sets are described as follows:

- **Asset reports:** Asset reports provide views on the discovered attributes of the network devices for inventory information.
- **Current Status reports:** Current Status reports provide useful status on any outstanding problems. They list the work queue and the waiting queue in terms of the devices that are affected, and are sorted by age of the events.
- **Monitoring reports:** Monitoring reports provide a list of devices that are polled under each monitoring policy to help you verify that you are polling the correct devices for the correct information.
- **Network Technology reports:** Network Technology reports provide insight into the states of BGP, OSPF, and VLAN networks that is based on information that is gathered during discovery.
- **Network Views reports:** These reports show details about network views.
- **Path Views reports:** You can use Path Views reports to view device and routing information for IP and MPLS TE paths.
- **Performance reports:** Performance reports show any historical performance data that the monitoring system collects and stores for diagnostic purposes. View trend and **topN** charts for data to gain insight on short-term behaviors.
- **Summary reports:** Summary reports show historical performance data that the monitoring system collected. These reports appear in a monitoring system in a dashboard style presentation.
- **Troubleshooting reports:** Troubleshooting reports help you identify problems while optimizing the discovery of the network. They also help identify possible problems that are discovered in the network, such as duplex mismatches.

- **Utility reports:** Utility reports display all discovered devices and their interfaces in different views.

## 15.2. Viewing and scheduling reports

You can view reports in a number of formats. You can also schedule reports to run on a defined schedule. For performance-related polling data, Apache Storm provides summarization tables so that you can report on longer periods of time.

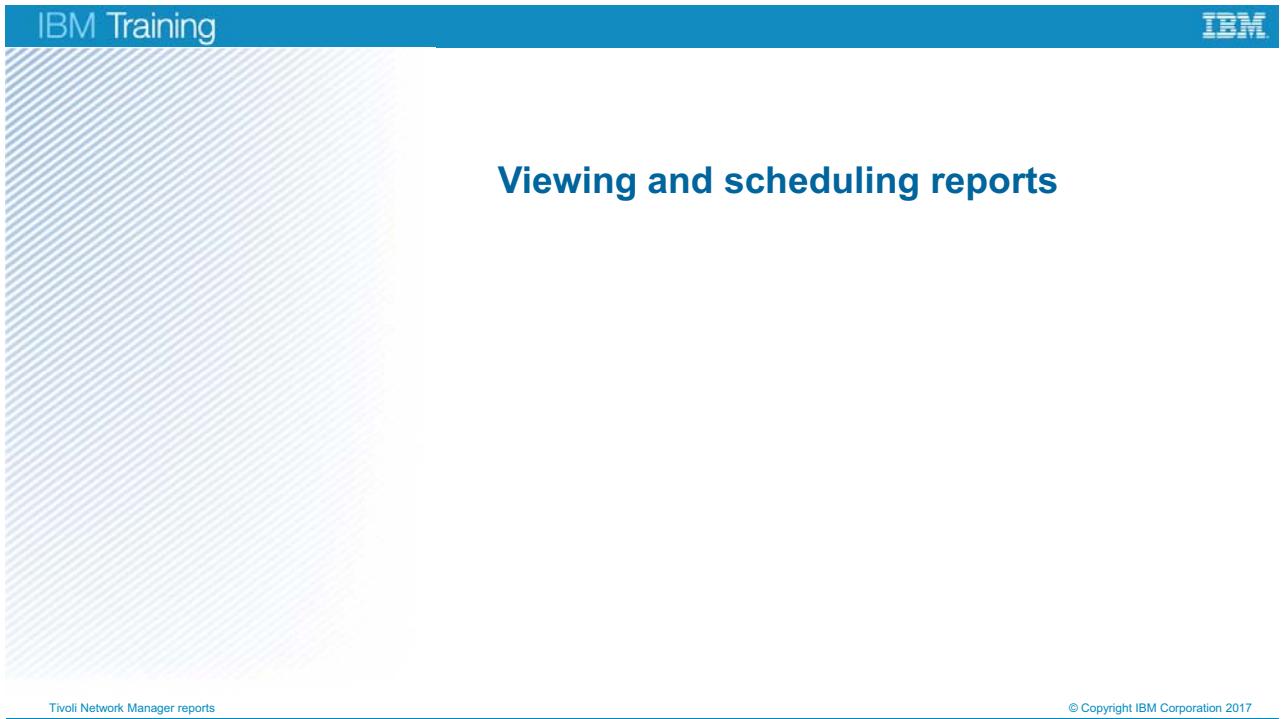


Figure 15-9. Viewing and scheduling reports

## Accessing the report list

- Click the report icon and then **Common Reporting > Network Manager**
- Click **Network Manager**

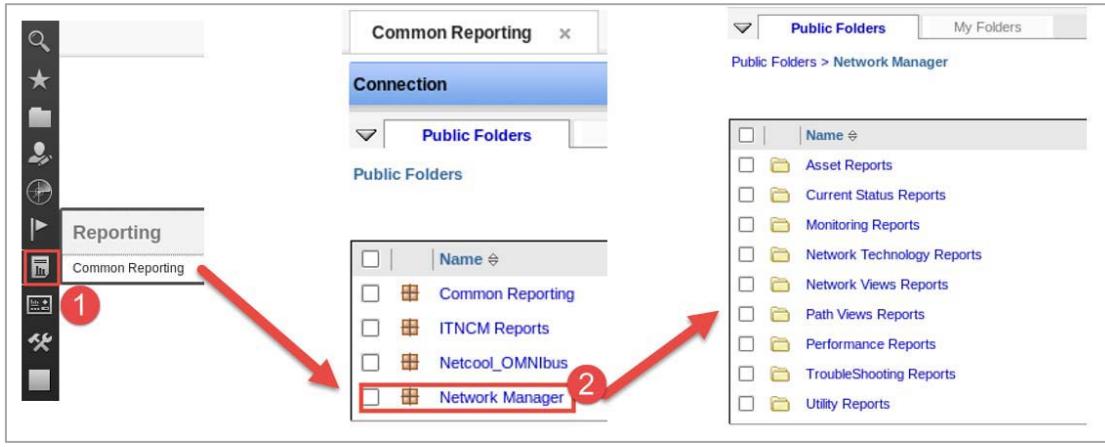


Figure 15-10. Accessing the report list

## Start a report

1. Click a report category such as **Asset Reports**
2. Click the green arrow to run a specific report
3. Select options and click **Run**

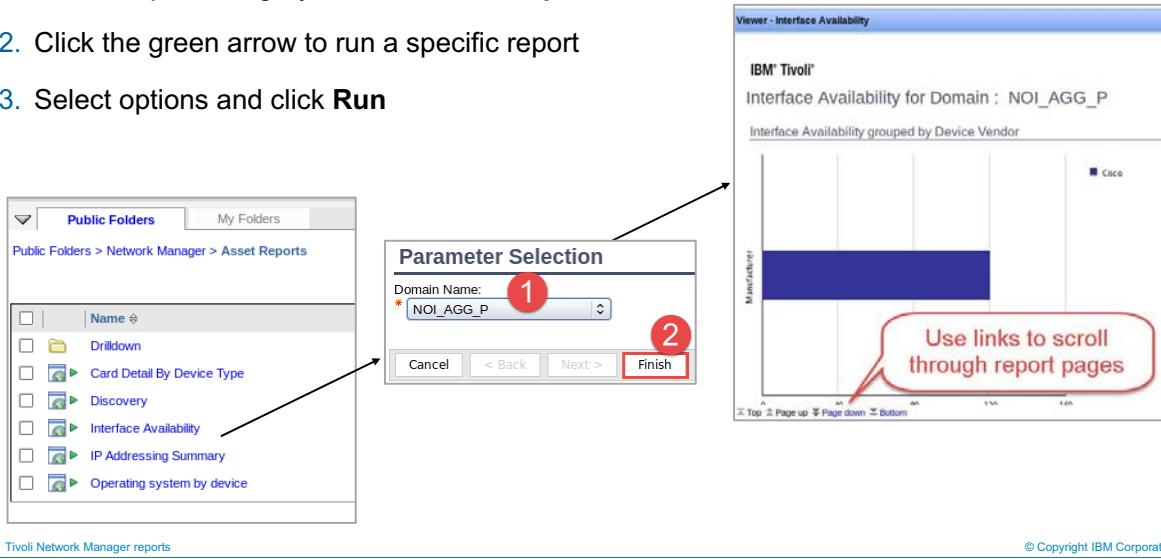
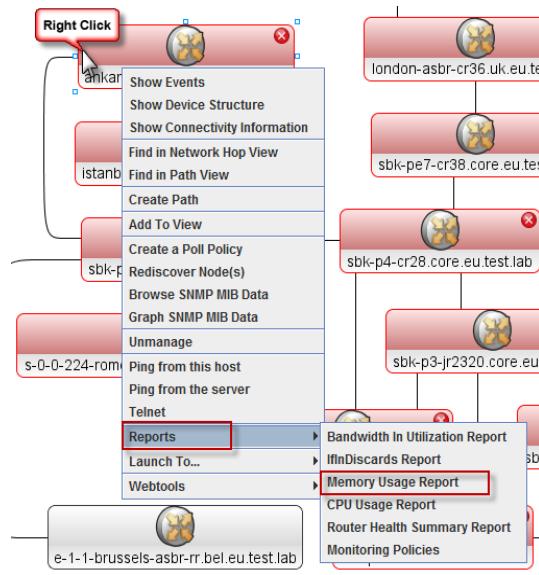


Figure 15-11. Start a report

## Start polling reports from device icon



Tivoli Network Manager reports

© Copyright IBM Corporation 2017

Figure 15-12. Start polling reports from device icon

## Available report formats

- Portable Document Format (PDF)
- Hypertext Markup Language (HTML)
- Microsoft Excel spreadsheet
- Comma-separated value (CSV) file
- Extensible Markup Language (XML)

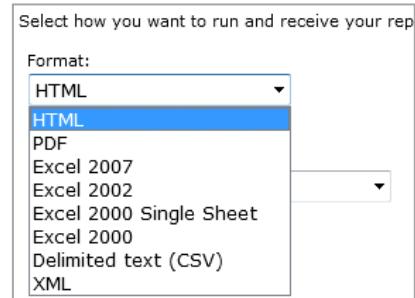


Figure 15-13. Available report formats

**IBM Training**

**View a report**

- Click **Add this report** to save report configuration

itnadmin | [About](#)

[Add this report](#)

IBM.

Specify a name and description - New Shortcut wizard

Name: Shortcut to Interface Availability

Description: Displays the availability of all interfaces grouped by node, class and vendor.

Screen tip: Displays the availability of all interfaces grouped by node, class and vendor.

Location: My Folders [Select another location...](#)

Cancel < Back Next > Finish

*Figure 15-14. View a report*

## Toolbar selections



Figure 15-15. Toolbar selections

**IBM Training**

**Report options**

- Select **More** to view report options
  - Create a report view
  - View report output versions (select from saved reports)

|                          | Name                       | Modified                    | Actions                                                                                             |
|--------------------------|----------------------------|-----------------------------|-----------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | Drilldown                  | January 22, 2016 5:42:39 PM | <input type="button"/> More...                                                                      |
| <input type="checkbox"/> | Card Detail By Device Type | January 22, 2016 5:42:39 PM | <input type="button"/> <input type="button"/> <input type="button"/> <input type="button"/> More... |
| <input type="checkbox"/> | Discovery                  | January 22, 2016 5:42:39 PM | <input type="button"/> <input type="button"/> <input type="button"/> <input type="button"/> More... |
| <input type="checkbox"/> | Interface Availability     | January 22, 2016 5:42:39 PM | <input type="button"/> <input type="button"/> <input type="button"/> <input type="button"/> More... |
| <input type="checkbox"/> | IP Addressing Summary      | January 22, 2016 5:42:39 PM | <input type="button"/> <input type="button"/> <input type="button"/> <input type="button"/> More... |
| <input type="checkbox"/> | Operating system by device | January 22, 2016 5:42:39 PM | <input type="button"/> <input type="button"/> <input type="button"/> <input type="button"/> More... |

Common Reporting X

Perform an action - Interface Availability

Available actions:

- Set properties
- View report output versions
- View my permissions...
- Run with options...
- Open with Report Studio
- Open with Cognos Workspace Advanced
- New schedule...
- Move...
- Copy...
- Create a shortcut to this entry...
- Create a report view of this report...
- Delete

Cancel

Tivoli Network Manager reports

© Copyright IBM Corporation 2017

Figure 15-16. Report options

## Click column names to sort reports

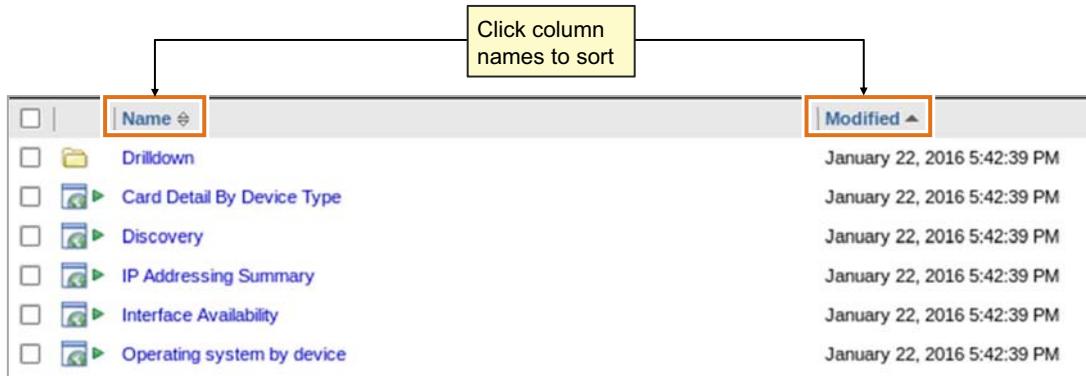


Figure 15-17. Click column names to sort reports

The screenshot shows the IBM Cognos Report Editor interface. At the top, there's a toolbar with various icons for file operations, font selection, and style. Below the toolbar is a main workspace where a report is being edited. The report title is "Discovery for Domain : NOI AGG P". The content area shows a table titled "Cisco" and "Cisco36xx" with the heading "BRU-ACS-01". The table lists network interfaces with the following data:

| Interface Name        | Interface Description | Interface Type   | Operation Status | Admin Status | IP Address   | MAC Address       | Entity ID |
|-----------------------|-----------------------|------------------|------------------|--------------|--------------|-------------------|-----------|
| BRU-ACS-01[ 0 [ 1 ] ] |                       |                  |                  |              | 10.10.255.11 |                   | 353       |
| BRU-ACS-01[ 0 [ 2 ] ] |                       |                  |                  |              | 10.10.2.25   |                   | 355       |
| BRU-ACS-01[ 0 [ 3 ] ] |                       |                  |                  |              | 10.10.2.29   |                   | 357       |
| BRU-ACS-01[ 0 [ 4 ] ] |                       |                  |                  |              | 10.10.2.10   |                   | 359       |
| BRU-ACS-01[ Fa0/0 ]   | FastEthernet0/0       | ethernet-csmacd  |                  |              | 10.10.2.10   | CC:0D:14:74:00:00 | 130       |
| BRU-ACS-01[ Fa0/0 ]   | FastEthernet0/0       | ethernet-csmacd  |                  |              |              | CC:0D:14:74:00:00 | 130       |
| BRU-ACS-01[ Fa1/0 ]   | FastEthernet1/0       | ethernet-csmacd  |                  |              | 10.10.2.25   | CC:0D:14:74:00:10 | 132       |
| BRU-ACS-01[ Fa1/0 ]   | FastEthernet1/0       | ethernet-csmacd  |                  |              |              | CC:0D:14:74:00:10 | 132       |
| BRU-ACS-01[ Fa2/0 ]   | FastEthernet2/0       | ethernet-csmacd  |                  |              | 10.10.2.29   | CC:0D:14:74:00:20 | 134       |
| BRU-ACS-01[ Fa2/0 ]   | FastEthernet2/0       | ethernet-csmacd  |                  |              |              | CC:0D:14:74:00:20 | 134       |
| BRU-ACS-01[ Fa3/0 ]   | FastEthernet3/0       | ethernet-csmacd  |                  |              |              | CC:0D:14:74:00:30 | 136       |
| BRU-ACS-01[ Lo0 ]     | Loopback0             | softwareLoopback |                  |              | 10.10.255.11 |                   | 138       |
| BRU-ACS-01[ Lo0 ]     | Loopback0             | softwareLoopback |                  |              |              |                   | 138       |

At the bottom right of the workspace, there's a red arrow pointing to a "More..." button in the toolbar. The toolbar also includes icons for file operations, font selection, and style.

Figure 15-18. Edit a Cognos report

You can view reports in XML, CSV, PDF, and other formats. You can also email reports to specified recipients.

IBM Training

**Schedule reports**

Common Reporting

**Schedule - Interface Availability**

Schedule this entry to run at a recurring date and time. You can run using the default values or specify the options. You can disable the schedule without losing any of its details.

Disable the schedule Priority: 3

Start: May 9, 2017

Frequency: 3 : 17 AM

Select the frequency by clicking on a link.

Every  week(s) on:

Monday  Tuesday  Wednesday  Thursday  
 Friday  Saturday  Sunday

Daily Frequency:

Every  Minute(s)  between 9 : 00 AM  and 5 : 00 PM

Credentials: itnmadmin (itnmadmin)

Options

Override the default values

Formats:  
Default

Accessibility:

Click to schedule report snapshot

Tivoli Network Manager reports © Copyright IBM Corporation 2017

Figure 15-19. Schedule reports

## Ways to run and visualize reports (1 of 2)

- On-demand report generates a report from the Tivoli Common Reporting viewer
- Snapshot shows the report that was scheduled as a PDF, HTML, XML, or Excel format from the Tivoli Common Reporting viewer
- When scheduling reports, save the report with the most appropriate parameter values
- View a list of available reports

```
cd /opt/IBM/JazzSM/reporting/bin
./trcCmd.sh -list -reports -username itnmadmin -password password
```

(The **smadmin** user can also run this script)

Figure 15-20. Ways to run and visualize reports (1 of 2)

## Ways to run and visualize reports (2 of 2)

- **On-demand report** generates a report from the Tivoli Common Reporting viewer
- **Snapshot** shows the report that was scheduled as a PDF, HTML, XML, or Excel format from the Tivoli Common Reporting viewer
- When scheduling reports, save the report with the most appropriate parameter values

Figure 15-21. Ways to run and visualize reports (2 of 2)

## Data model

Network Manager data model

- Each product that uses the Cognos data model provides namespaces, which contain query subjects to build up reports

The Network Manager data model provides the following namespaces for designing reports:

- **Event** contains query subjects to create status reports
- **Monitoring Data** contains query subjects to create performance reports
  - The data for the Monitoring Data namespace comes from the **NCPOLLDATA** database
- **Network** contains query subjects to create asset and troubleshooting reports
  - The data for the Network namespace comes from the NCIM database
- **Network Views** contains query subjects to create reports about network views and policies that update views
  - The data for the Network Views namespace comes from the **NCPGUI** and **NCMONITOR** databases
- **Path Views** contains query subjects to create Path Views reports

*Figure 15-22. Data model*

The data model and namespaces are described as follows:

- **Network Manager data model:** Each product that uses the Cognos data model provides namespaces, which contain query subjects to use to build up reports.
- **Namespaces:** The Network Manager data model provides the following namespaces for designing reports:
  - **Event:** The Event namespace contains query subjects to create Current Status reports.
  - **Monitoring Data:** The Monitoring Data namespace contains query subjects to create Performance reports. The polled data time stamp has a time dimension relationship to allow time dimension reports. The data for the Monitoring Data namespace comes from the **NCPOLLDATA** database.
  - **Network:** The Network namespace contains query subjects to create Asset and Troubleshooting reports. The data for the Network namespace comes from the **NCIM** database.
  - **Network Views:** The Network Views namespace contains query subjects to create reports about network views and policies that update views. The data for the Network Views namespace comes from the **NCPGUI** and **NCMONITOR** databases.
  - **Path Views:** The Path Views namespace contains query subjects to create Path Views reports.
  - **Shared:** The Shared namespace contains query subjects that can be shared to prevent query subject duplicates.

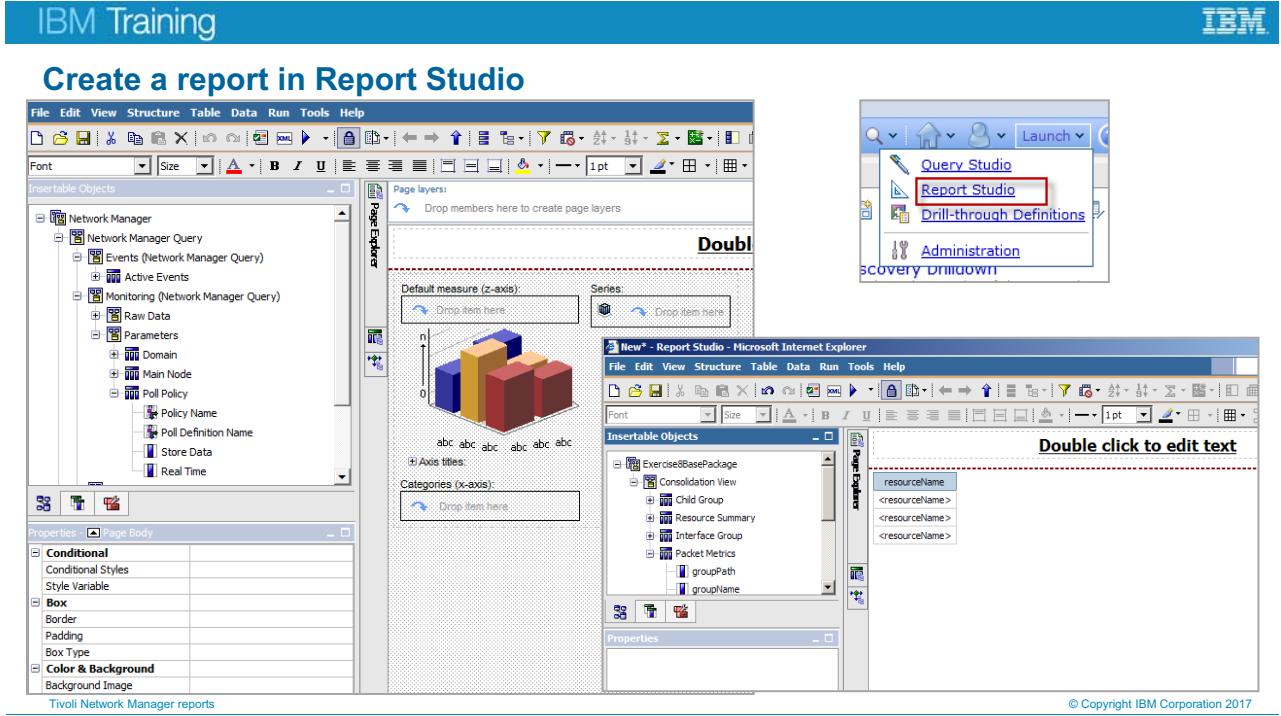


Figure 15-23. Create a report in Report Studio

Create reports in a web-based tool that professional report authors use to build sophisticated, multiple-page, multiple-query reports against multiple databases. You can create any reports that your company requires.

To create a report in Report Studio, do the following steps:

1. From a Windows PC, log on to the Tivoli Common Reporting interface, and go to **Common Reporting**.
2. In the **Work with reports** window on the right, choose the Report Studio from the Launch menu.
3. The **Report Studio** opens.
4. Use the menu controls to create a new report or edit existing ones by formatting the layout and manipulating the data that appears in the report.
5. Save your report, and run it anytime you need to present its underlying data.

## 15.3. Included reports

Tivoli Network Manager 4.2 eliminates all of the BIRT reports that are found in previous versions.  
All reports are now based on the Cognos engine.



Figure 15-24. Included reports

## Asset reports

Public Folders > Network Manager > Asset Reports

| <input type="checkbox"/> | Name                                                                                                                                                                                                                                                                   |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | Drilldown                                                                                                                                                                                                                                                              |
| <input type="checkbox"/> | Drilldown Reports                                                                                                                                                                                                                                                      |
| <input type="checkbox"/> |  <b>Card Detail By Device Type</b><br>Displays the same data source as the Network Manager Card Detail by Card Type report, except it is organized by device type.                    |
| <input type="checkbox"/> |  <b>Discovery</b><br>Displays the results of the ITNM network discovery. Using this report, the user can view a table showing the devices on your network organized by device vendor. |
| <input type="checkbox"/> |  <b>IP Addressing Summary</b><br>Displays information on the IP addresses used in the network grouped according to CIDR notations.                                                    |
| <input type="checkbox"/> |  <b>Interface Availability</b><br>Displays the availability of all interfaces grouped by node, class and vendor.                                                                      |
| <input type="checkbox"/> |  <b>Operating system by device</b><br>Displays information on the Operating Systems running on the various devices in your network.                                                   |

Figure 15-25. Asset reports

## Current Status reports

Public Folders > Network Manager > Current Status Reports

|                          | Name                                                                                                                                    |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | Acknowledged Events by First Occurrence<br>Displays a list of events that are acknowledged and have severity higher than warning.       |
| <input type="checkbox"/> | Non Acknowledged Events by First Occurrence<br>Displays a list of events that are unacknowledged and have severity higher than warning. |

Figure 15-26. Current Status reports

## Network Technology reports

Public Folders > Network Manager > Network Technology Reports

|                          | Name                                                                                                                   |
|--------------------------|------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | Drilldown                                                                                                              |
| <input type="checkbox"/> | Drilldown Reports                                                                                                      |
| <input type="checkbox"/> | BGP Details                                                                                                            |
|                          | Displays charts and tables with BGP Session and Autonomous System Summary Information.                                 |
| <input type="checkbox"/> | BGP Summary                                                                                                            |
|                          | Displays charts and tables with BGP Session and Autonomous System Summary Information.                                 |
| <input type="checkbox"/> | LTE Interfaces                                                                                                         |
|                          | Displays the LTE interfaces grouped by type.                                                                           |
| <input type="checkbox"/> | MPLS VPN Details                                                                                                       |
|                          | Displays detailed information about discovered MPLS VPNs including VRFs, Route Distinguishers, Route Targets and VPWS. |
| <input type="checkbox"/> | MPLS VPN Summary                                                                                                       |
|                          | Displays charts and tables with MPLS VPN Summary information.                                                          |
| <input type="checkbox"/> | VLAN Details                                                                                                           |
|                          | Displays information about VLANs and trunk ports.                                                                      |

Figure 15-27. Network Technology reports

## Monitoring reports

| Public Folders > Network Manager > Monitoring Reports |                                                                          |
|-------------------------------------------------------|--------------------------------------------------------------------------|
|                                                       | Name                                                                     |
| <input type="checkbox"/>                              | Drilldown                                                                |
| <input type="checkbox"/>                              | Drilldown Reports                                                        |
| <input type="checkbox"/>                              | Monitored Device Details                                                 |
|                                                       | Displays detailed information about how a selected device is monitored.  |
| <input type="checkbox"/>                              | Monitored Policy Details                                                 |
|                                                       | Displays detailed information about a selected monitoring policy.        |
| <input type="checkbox"/>                              | Monitored Policy Summary                                                 |
|                                                       | Displays summary information about ITNM monitoring policies and polling. |

Figure 15-28. Monitoring reports

IBM Training 

## Network Views reports



Connection itnadmin

Public Folders My Folders

Public Folders > Network Manager > Network Views Reports

**Monitored Network Views**  
Show the poll definitions, policies and total entities been Monitored  
February 4, 2011 5:50:07 PM  
 More...

**Monitored Network Views Drilldown**  
Show the details like devices of a Monitored View  
February 4, 2011 5:50:07 PM  
 More...

Tivoli Network Manager reports

© Copyright IBM Corporation 2017

Figure 15-29. Network Views reports

## New network health reports

- **Network Availability Summary**
- **Device Inbound Traffic Health Summary**
- **Device Outbound Traffic Health Summary**
- **Router Health Summary** uses industry-standard SNMP variables that support devices from many vendors

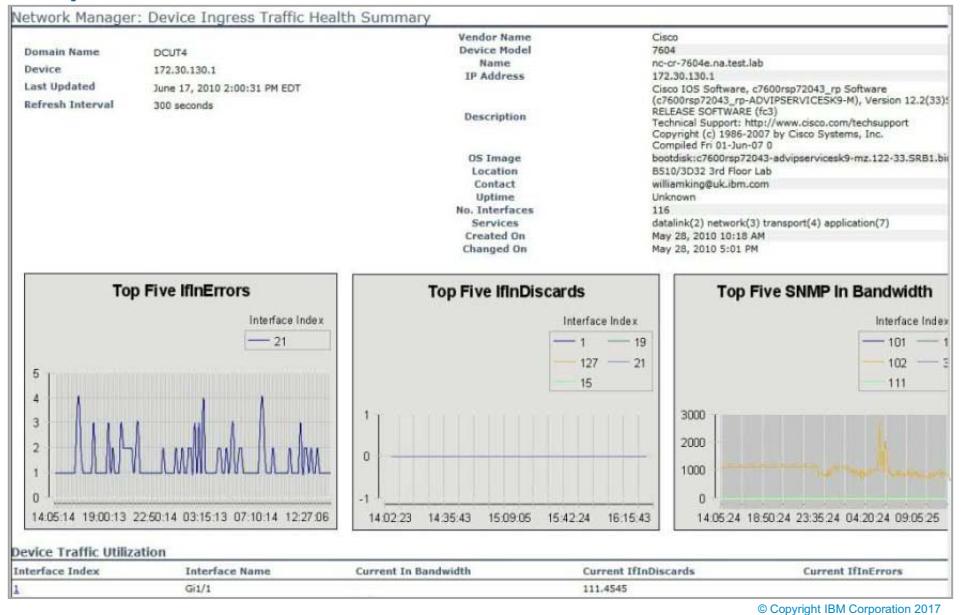


Figure 15-30. New network health reports

## Path Views reports

Public Folders > Network Manager > Path Views Reports

|                          | Name                                                                          |
|--------------------------|-------------------------------------------------------------------------------|
| <input type="checkbox"/> | Drilldown                                                                     |
| <input type="checkbox"/> | Drilldown Reports                                                             |
| <input type="checkbox"/> | IP Path Summary                                                               |
|                          | Summary Report of all IP Paths configured on the system.                      |
| <input type="checkbox"/> | IP Routing Info                                                               |
|                          | A detail routing Report for a specific list of entities on a specific Path.   |
| <input type="checkbox"/> | MPLS TE Path Summary                                                          |
|                          | A Summary Report of all MPLS-TE Paths Tunnels configured on the system.       |
| <input type="checkbox"/> | MPLS TE Routing Info                                                          |
|                          | A detail routing Report for a specific list of entities on a specific Tunnel. |

Figure 15-31. Path Views reports

## Performance reports

Public Folders > Network Manager > Performance Reports

|                          | Name                                                                                                                                                             |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | Drilldown                                                                                                                                                        |
|                          | Drilldown Reports                                                                                                                                                |
| <input type="checkbox"/> | Bandwidth Top N                                                                                                                                                  |
|                          | Displays the Bandwidth of the Top N devices. An appropriate policy with store option enabled must be set in order for report parameters to be selected.          |
| <input type="checkbox"/> | Bandwidth Utilization                                                                                                                                            |
|                          | Displays the summary of bandwidth utilization for all devices.                                                                                                   |
| <input type="checkbox"/> | Composite Trending                                                                                                                                               |
|                          | Displays the Composite Trending Report. An appropriate policy with store option enabled must be set in order for report parameters to be selected.               |
| <input type="checkbox"/> | Device Availability Summarization                                                                                                                                |
|                          | Displays the device availability summarization data for the last 7 days.                                                                                         |
| <input type="checkbox"/> | Device Summarization                                                                                                                                             |
|                          | Displays the devices summarization data.                                                                                                                         |
| <input type="checkbox"/> | Historical SNMP Top or Bottom N                                                                                                                                  |
|                          | Displays the Top N and Bottom N devices.                                                                                                                         |
| <input type="checkbox"/> | Historical SNMP Trend Analysis                                                                                                                                   |
|                          | Displays the Historical SNMP Trend Analysis Report.                                                                                                              |
| <input type="checkbox"/> | Historical SNMP Trend Quick View                                                                                                                                 |
|                          | Displays the Historical SNMP Trend Quick View Report. An appropriate policy with store option enabled must be set in order for report parameters to be selected. |
| <input type="checkbox"/> | Interface Availability Summarization                                                                                                                             |
|                          | Displays the interfaces availability summarization data for the last 7 days.                                                                                     |
| <input type="checkbox"/> | Interface Summarization                                                                                                                                          |
|                          | Displays the interfaces summarization data.                                                                                                                      |
| <input type="checkbox"/> | System Availability Summary                                                                                                                                      |
|                          | Displays the availability of the devices to being polled.                                                                                                        |

© Copyright IBM Corporation 2017

Figure 15-32. Performance reports

## Reports for troubleshooting network devices

Public Folders > Network Manager > TroubleShooting Reports

| <input type="checkbox"/> | Name                                                                                                                                            |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | <b>Connected Interface Duplex Mismatch</b><br>Connected Interface Duplex Mismatch                                                               |
| <input type="checkbox"/> | <b>Devices Pending Delete on Next Discovery</b><br>Devices Pending Delete on Next Discovery                                                     |
| <input type="checkbox"/> | <b>Devices with Unclassified SNMP Object IDs</b><br>Display those devices with SNMP Object IDs that have not been assigned to specific classes. |
| <input type="checkbox"/> | <b>Devices with Unknown SNMP Object IDs</b><br>Devices with Unknown SNMP Object IDs                                                             |
| <input type="checkbox"/> | <b>Devices with no SNMP Access</b><br>Displays those devices that have no SNMP Access.                                                          |

Tivoli Network Manager reports

© Copyright IBM Corporation 2017

Figure 15-33. Reports for troubleshooting network devices

After you run an initial discovery, print the **Devices with no SNMP Access** report. Provide this report to the network team so that they can resolve SNMP access issue problems.

Run the following reports to identify Object ID numbers that need to be added to an Active Object Class (AOC) file in the `$ITNMHOME/aoc` directory:

- **Devices with Unknown SNMP Object IDs**
- **Devices with Unclassified SNMP Object IDs**

After you edit the appropriate AOC files, rerun discovery to see these devices with their appropriate icons.

IBM Training

Utility reports

IBM

Public Folders > Network Manager > Utility Reports

| Name                                                                                                                                                                                                                                     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/>  <b>Discovered nodes and interfaces flatfile list</b><br>Displays a plain list of discovered device and interface information. |

Figure 15-34. Utility reports

## 15.4. Installing Tivoli Common Reporting

Installing Tivoli Common Reporting is a separate step. With previous versions of Tivoli Network Manager, the installation process automatically installed reporting. However, that is now a separate step because the Tivoli Common Reporting server usually runs on separate hardware from Tivoli Network Manager 4.2.



Figure 15-35. *Installing Tivoli Common Reporting*

## Installing Tivoli Common Reporting

- Tivoli Network Manager 4.2 requires Tivoli Common Reporting 2.1.1 FP2 or later
- Tivoli Common Reporting is no longer installed by the Tivoli Network Manager installer

*Figure 15-36. Installing Tivoli Common Reporting*

## Deployment options

- Tivoli Network Manager installation on an environment with a Tivoli Common Reporting
  - Use the IBM Installation Manager to install Network Manager reports
  - No postinstallation steps are necessary
- Tivoli Common Reporting is installed after the Tivoli Network Manager installation
  - The Tivoli Network Manager installer copies all the files that are necessary to deploy and configure reports during the installation
  - Select the postinstallation option to deploy and configure reports
  - You can also run `configTCR` as the `smadmin` user with the command-line option to deploy reports:  
`configTCR.sh -p smadmin_password -d db_password -i install`

Figure 15-37. Deployment options

## Content Manager configuration

You can change the Tivoli Common Reporting default Content Manager database from Derby to Cognos for greater efficiency in large environments

- Create Cognos Content Manager for DB2
  - Log in as DB2 database administrator
  - Run script to create the DB2 database:

```
$ITNMHOME/scripts/slq/db2/create_db2_cognos_database.sh ContentManagerDbName user
```

- Configure Cognos Content Manager data source by running the `modify_cognos_cm` script

```
$ITNMHOME/products/tnm/bin/modify_cognos_cm -filename
/space/ats/tipv2Components/TCRComponent/cognos/
configuration/cogstartup.xml -dbhost db_hostname -username ncim -password -
dbport 50000 -dbname ITNMCM84 -dbtype db2
```

Figure 15-38. Content Manager configuration

## Use Cognos administration to change sign-on password

1. Select **Reporting > Common Reporting** and click **Launch** and select **Administration**

Work with reports

Connection: rob

Public Folders

| Name             | Modified                      | Actions                 |
|------------------|-------------------------------|-------------------------|
| Common Reporting | September 16, 2009 7:28:25 AM | <a href="#">More...</a> |
| Network Manager  | June 18, 2012 6:52:26 PM      | <a href="#">More...</a> |

2. Click the **Configuration** tab

Work with reports

Administration

Status Security Configuration

Current Activities

- Background activities (selected)
- Interactive activities

Current Activities - Background activities

Total (0)

| Category  | Count |
|-----------|-------|
| Pending   | 0     |
| Executing | 0     |
| Waiting   | 0     |
| Suspended | 0     |

Figure 15-39. Use Cognos administration to change sign-on password

IBM Training



## Changing sign-on password (1 of 2)

**Work with reports**

**Administration**

**Configuration**

Directory > Cognos

Status Security Configuration

Data Source Connections Content Administration Distribution Lists and Contacts Printers Styles Portlets Dispatchers and Services

Name Modified Actions

IBM\_TRAM April 12, 2011 10:23:24 PM More...

NCIM April 12, 2011 10:22:42 PM More...

NCMONITOR April 12, 2011 10:22:51 PM More...

NCPGUIT April 12, 2011 10:22:59 PM More...

NCPOLLDATA April 12, 2011 10:23:07 PM More...

PARAMETERS April 12, 2011 10:23:16 PM More...

servletInventory

Last refresh time: July 12, 2013 4:39:41 PM

**3.** Repeat the following steps for each of the data sources:

- NCIM
- NCPOLLDATA
- PARAMETERS

**4.** Double-click a data source

**5.** Select the check box and click **More**

**6.** Click **Set Properties**

**Directory > Cognos > NCIM > NCIM**

Status Security Configuration

Data Source Connections Content Administration Distribution Lists and Contacts Printers Styles Portlets Dispatchers and Services

Name Modified Actions

ncim April 2, 2012 10:43:58 AM More...

Last refresh time: July 12, 2013 4:48:36 PM

© Copyright IBM Corporation 2017

*Figure 15-40. Changing sign-on password (1 of 2)*

5. Select the check box and click **More**
  6. Click **Set Properties**

3. Repeat the following steps for each of the data sources:
    - NCIM
    - NCPOLDATA
    - PARAMETERS
  4. Double-click a data source

## Changing sign-on password (2 of 2)

7. Click the **Signon** tab and then click **Edit the signon**

The screenshot shows two windows side-by-side. The left window is titled 'Work with reports' and contains a 'Administration' section with tabs for Status, Security, and Configuration. Under Configuration, there is a 'Data Source Connections' list that includes 'ncpolldata'. A modal dialog box titled 'Enter the signon - ncpodata' is open over the main window. It has fields for 'User ID' (containing 'ncim'), 'Password' (containing masked text), and 'Confirm Password' (containing masked text). The right window is titled 'Administration' and shows 'Set properties - ncim'. It has tabs for General, Signon (which is highlighted with an orange box), and Permissions. Below the tabs is a sub-section titled 'Signon' with a link 'Edit the signon...'. The right window also includes a table for managing users and groups.

8. Enter the database user name and password on the following pane  
 ▪ Click **OK** to save

© Copyright IBM Corporation 2017

Figure 15-41. Changing sign-on password (2 of 2)

## 15.5. Troubleshooting Cognos

The troubleshooting information in this unit is primarily for support personnel. Other students can regard this section as reference material.



Figure 15-42. Troubleshooting Cognos

## Tivoli Network Manager data model (1 of 2)

- The data model is developed with Cognos Framework Manager
- The Tivoli Network Manager Cognos data model provides predefined queries and relationships between queries that make it easy to drag fields to build a report

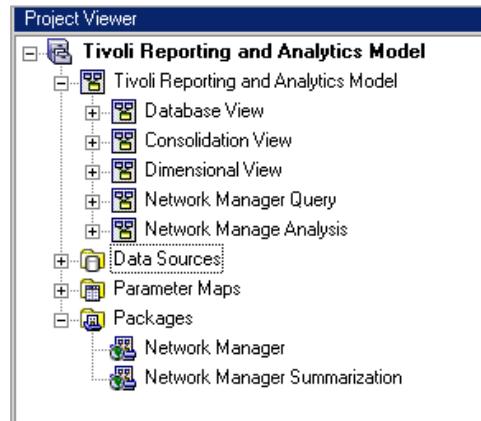


Figure 15-43. Tivoli Network Manager data model (1 of 2)

## Tivoli Network Manager data model (2 of 2)

A data model has three main View sections:

- **Database View:** Queries and relationships are defined here
- **Dimensional View:** Hierarchies and time dimensions are defined here
- **Consolidation View:** Namespaces and query subjects are defined here
  - These objects are also globalized and they are published for users

Data model **data sources** section

- A user cannot change a data model data source
- Each data model query must define which data source it uses
- The data model data source has a link to the Cognos data source that is defined in the **Cognos Administration** page where the administrator user defines database credentials

Figure 15-44. Tivoli Network Manager data model (2 of 2)

## The Cognos report package

The Cognos report package is built with Tivoli Common Reporting tools on Windows

- The Cognos build requires a Cognos server that is installed and running because Tivoli Common Reporting tools use Cognos SDK libraries and services
- The report package bundles the Tivoli Network Manager data model, report designs, and globalization files

Figure 15-45. The Cognos report package

## Tivoli Common Reporting log files

These files aid in determining problems with Tivoli Common Reporting

- Errors are captured in the system files
  - `/opt/IBM/JazzSM/profile/logs/server1/SystemOut.log`
  - `/opt/IBM/JazzSM/profile/logs/server1/SystemErr.log`
- Other useful log files are in the `/opt/IBM/JazzSM/profile/logs` directory and its subdirectories

Figure 15-46. Tivoli Common Reporting log files

## Troubleshooting Cognos

- If a Cognos report gives an SQL error, the report SQL query can be found on the report design
- In case a report is empty, verify whether the correct Tivoli Network Manager discovery agents are enabled
- If Cognos database connection failed, check to see whether the correct environment variables are set in `/opt/IBM/JazzSM/reporting/bin/setTCRenv.sh`
- This webpage has the necessary environment variables for each database:  
<http://tinyurl.com/y7ju4kt2>

Figure 15-47. Troubleshooting Cognos

## Cognos viewer errors (1 of 2)

- On AIX and Linux on System z, it takes longer to start Cognos dispatcher
- If you see a dispatcher error, wait longer
- Check again
- Restart JazzSM if necessary

DPR-ERR-2109 The dispatcher cannot service the request now. The dispatcher is still initializing. Contact your administrator if this problem persists.

Figure 15-48. Cognos viewer errors (1 of 2)

## Cognos viewer errors (2 of 2)

- Cognos uses cookies for a number of things that can cause errors on the Tivoli Common Reporting viewer
- Clear browser cookies and restart the browser to eliminate the following errors:  
com.cognos.acccman.jcam.crypto.CAMCryptoException: CAM-CRP-1346  
Failed to get the HMAC value to verify the package capability trust token.  
Invalid digest of the common symmetric key.  
Unable to find an appropriate common symmetric key to decrypt the data.



© Copyright IBM Corporation 2017

Figure 15-49. Cognos viewer errors (2 of 2)

## Tivoli Common Reporting documentation

- The Tivoli Common Reporting documentation in the IBM Knowledge Center has installation and configuration information:

<http://www.ibm.com/support/knowledgecenter/SSH2DF/welcome>

- Tivoli Common Reporting must be installed separately from Tivoli Network Manager
  - If you install Tivoli Common Reporting first, the Tivoli Network Manager asset and event reports are installed during the Tivoli Network Manager installation
  - If you install Tivoli Common Reporting after Tivoli Network Manager, you must install the asset and even reports separately

Figure 15-50. Tivoli Common Reporting documentation

## Troubleshooting an Oracle data source (1 of 3)

- The following error message shows that the data source that is called **orac** defined at **\$ORACLEHOME/network/admin /tnsnames.ora** does not match what is defined on Cognos Oracle data source

```
QE-DEF-0285 The logon failed.
QE-DEF-0323 The DSN(ODBC)/ServiceName is invalid. Either the DSN is missing or the host is
inaccessible.
RQP-DEF-0068 Unable to connect to at least one database during a multi-database attach to 1
database(s) in:
testDataSourceConnection

UDA-SQL-0031 Unable to access the "testDataSourceConnection" database.
UDA-SQL-0532 Data Source is not accessible: "orac".
ORA-12154: TNS:could not resolve the connect identifier specified
```

```
oracl =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)
(HOST = p6tpm069) (PORT = 1521))
)
(CONNECT_DATA =
(SID = oracl)))
```

- The following error message indicates that Oracle cannot connect to the database
  - The user must fix **tnsnames.ora** to have the right SID value:

```
QE-DEF-0285 The logon failed.
QE-DEF-0325 The logon failed for the following reason:
RQP-DEF-0068 Unable to connect to at least one database during a multi-database attach to 1
database(s) in:
testDataSourceConnection

UDA-SQL-0031 Unable to access the "testDataSourceConnection" database.
UDA-SQL-0107 A general exception has occurred during the operation "attach".
ORA-12505: TNS:listener does not currently know of SID given in connect descriptor
```

Figure 15-51. Troubleshooting an Oracle data source (1 of 3)

## Troubleshooting an Oracle data source (2 of 3)

The following error message indicates that the user does not have access to the database schema:

- Performance reports that use **NCPOLLDATA** schema are more prone to this error
  - The Oracle user **ncpolldata** has access to the **NCPOLLDATA** schema
    - However, the user **ncim** does not have permission to query the database
  - When users are unaware of this permissions issue, they sometimes set the data source user to **ncim**
    - This configuration error results in the following error message

RQP-DEF-0177

An error occurred while performing operation 'sqlUpdateAttach' status='-9'.  
[Details](#)

UDA-SQL-0107 A general exception has occurred during the operation "update attach".ORA-01435: user does not exist RSV-SRV-0042  
Trace back:RSReportService.cpp(771): QFException: CCL\_CAUGHT: RSReportService::process()RSReportServiceMethod.cpp(265):  
QFException: CCL\_RETHROW: RSReportServiceMethod::process(): asynchWait\_RequestRSASyncExecutionThread.cpp(787): QFException:  
RSASyncExecutionThread::checkExceptionRSASyncExecutionThread.cpp(224): QFException: CCL\_CAUGHT:  
RSASyncExecutionThread::checkExceptionRSASyncExecutionThread.cpp(224): QFException: CCL\_RETHROW:

*Figure 15-52. Troubleshooting an Oracle data source (2 of 3)*

### Troubleshooting an Oracle data source (3 of 3)

If you have problems with the integration of Tivoli Network Manager with Oracle, make sure that the necessary environment variables are set:

```
LIBPATH=:/space/ats/netcool/platform/aix5/oracleInstantClient11.1
ORACLE_HOME=/space/ats/netcool/platform/aix5/oracleInstantClient11.1
TNS_HOME=/space/ats/netcool/platform/aix5/oracleInstantClient11.1/network/admin/tnsnames.ora
```

Figure 15-53. Troubleshooting an Oracle data source (3 of 3)

## Troubleshooting a DB2 data source (1 of 2)

- DB2 requires the following environment variable:
  - LIBPATH=\$DB2HOME/lib32
  - Cognos works only with DB2 32-bit libraries
- The following error means that the Cognos DB2 data source was not able to find the database:

```
UDA-SQL-0031 Unable to access the "testDataSourceConnection" database.
UDA-SQL-0129 Invalid login information was detected by the underlying database.
[IBM][CLI Driver] SQL1013N The database alias name or database name "ITNMDB1" could not be
found. SQLSTATE=42705
```

Figure 15-54. Troubleshooting a DB2 data source (1 of 2)

## Troubleshooting a DB2 data source (2 of 2)

- Cognos requires the DB2 Client and a database that is cataloged in JazzSM
- After you install Tivoli Network Manager with DB2, run a script to catalog the Tivoli Network Manager database:

```
$NCHOME/precision/scripts/sql/db2/catalog_db2_database.sh
```

Figure 15-55. Troubleshooting a DB2 data source (2 of 2)

## Exercise: Tivoli Network Manager reports

Tivoli Network Manager reports

© Copyright IBM Corporation 2017

Figure 15-56. Exercises: Tivoli Network Manager reports

Complete the exercise for this unit.

## Review questions

1. Approximately how many predefined reports are available in Tivoli Network Manager 4.2?
2. To what locations can you save a report shortcut?
3. In what formats can you produce Tivoli Network Manager reports?

Figure 15-57. Review questions

Write your answers here:

## Unit summary

- View an asset report for IBM Tivoli Network Manager
- Create the same report in several different report formats
- Create a report snapshot
- Schedule a report snapshot

---

Tivoli Network Manager reports

© Copyright IBM Corporation 2017

*Figure 15-58. Unit summary*

## Review answers

1. Tivoli Network Manager 4.2 has approximately 60 default reports.
2. You can save a report to any directory on the server to which you have access. Typically, the report is saved to the user's home directory.
3. Tivoli Network Manager can produce reports in these formats: Portable Document Format (PDF), Hypertext Markup Language (HTML), Microsoft Excel spreadsheet, comma-separated value (CSV) file, Extensible Markup Language (XML)

Figure 15-59. Review answers

# Unit 16. Event searches

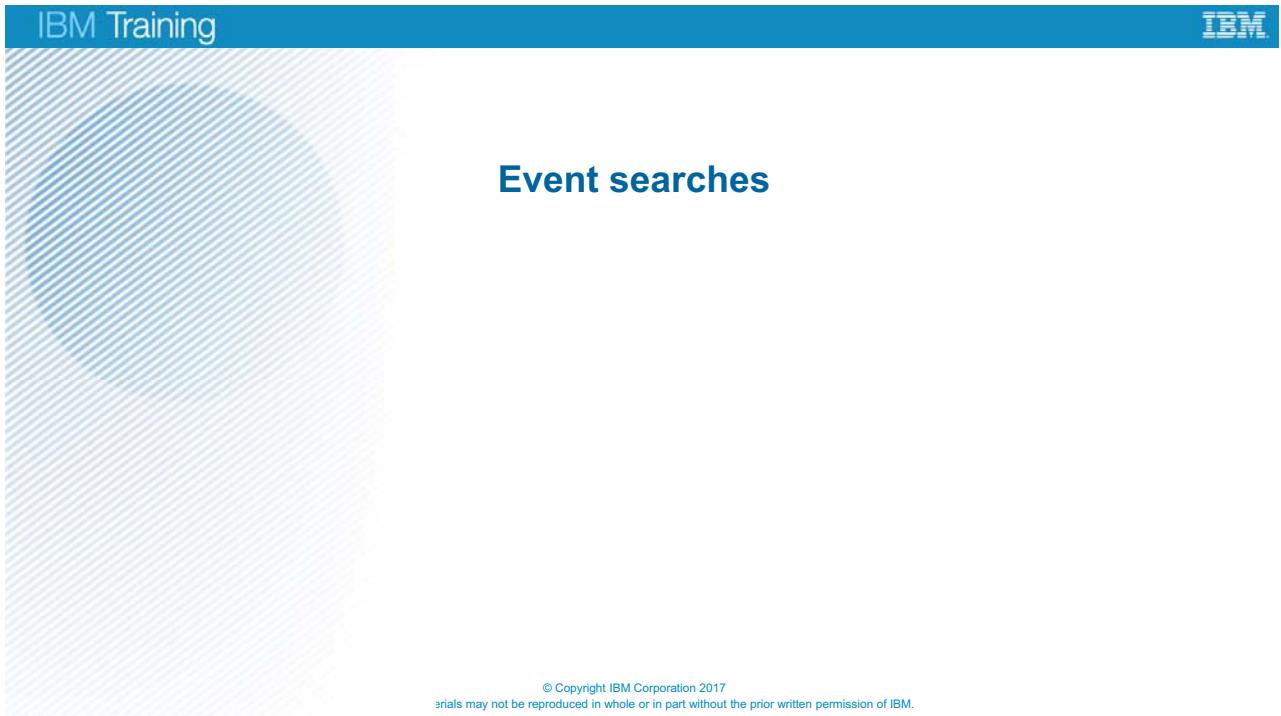


Figure 16-1. Event searches

## Estimated time

00:20

## Overview

This unit introduces the analytics engine and shows how it processes log and event data. This already-configured ability of Netcool Operations Insight quickly locates all events that have common elements, and groups the events with tags. Operators can then quickly find just the kinds of events they need to do their tasks.

## Objectives

After you complete this unit, you should be able to do these tasks:

- Describe how Log Analysis processes events
- Describe the features of event search
- Explain how to conduct an event search

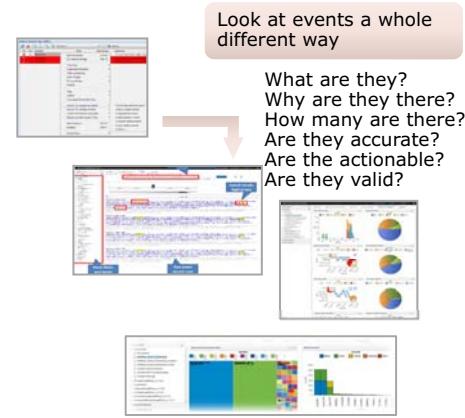
Event searches

© Copyright IBM Corporation 2017

*Figure 16-2. Objectives*

## Event Search Analytics for operational agility and efficiency

- Accelerate problem isolation, identification, and resolve for greater operations agility
  - Identify problems within seconds with insight to your near real-time and historical event data
  - Isolate problems faster by bringing relevant events data into problem investigations
  - Repair problems by providing the correct details quickly
  
- Turn your historical event data into insight for higher operational efficiency
  - Identify event hot spots that impact your workload with simple visualization: Top Nodes, Top Locations, Top Managers, Top Events with Tickets, and more
  - Easily see related events that can be candidates for suppression
  - Use visualizations to quickly isolate the problems that make more difference to your environment



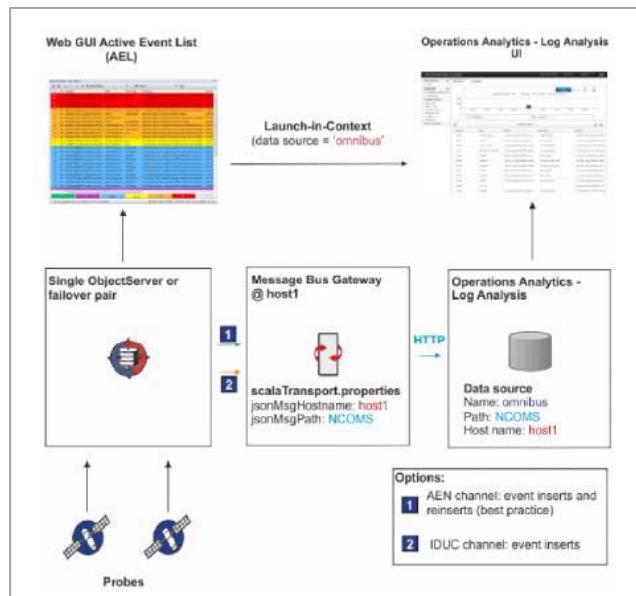
Provides rapid interactive line of sight to opportunities for improving efficiency, making Operations task simpler

© Copyright IBM Corporation 2017

*Figure 16-3. Event Search Analytics for operational agility and efficiency*

## Event search overview

- OMNIbus events in the ObjectServer are sent to Log Analysis by the Message Bus Gateway
- OMNIbus event data is stored in Log Analysis in the context of a data source
  - Default name is **omnibus**
- OMNIbus events are indexed and searchable in Log Analysis
- The search runs against a data source



Event searches

© Copyright IBM Corporation 2017

*Figure 16-4. Event search overview*

## Searching events

- Rapidly search event history with user-defined criteria
- Results show numbers of values for prebuilt index patterns

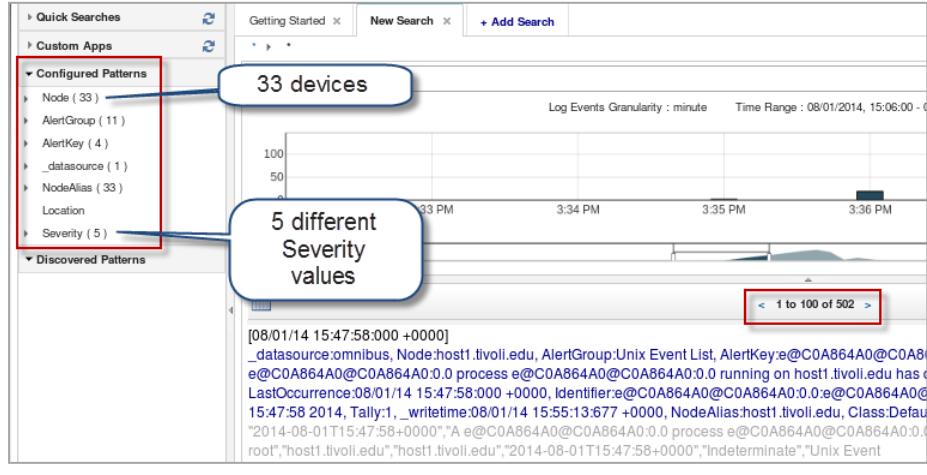
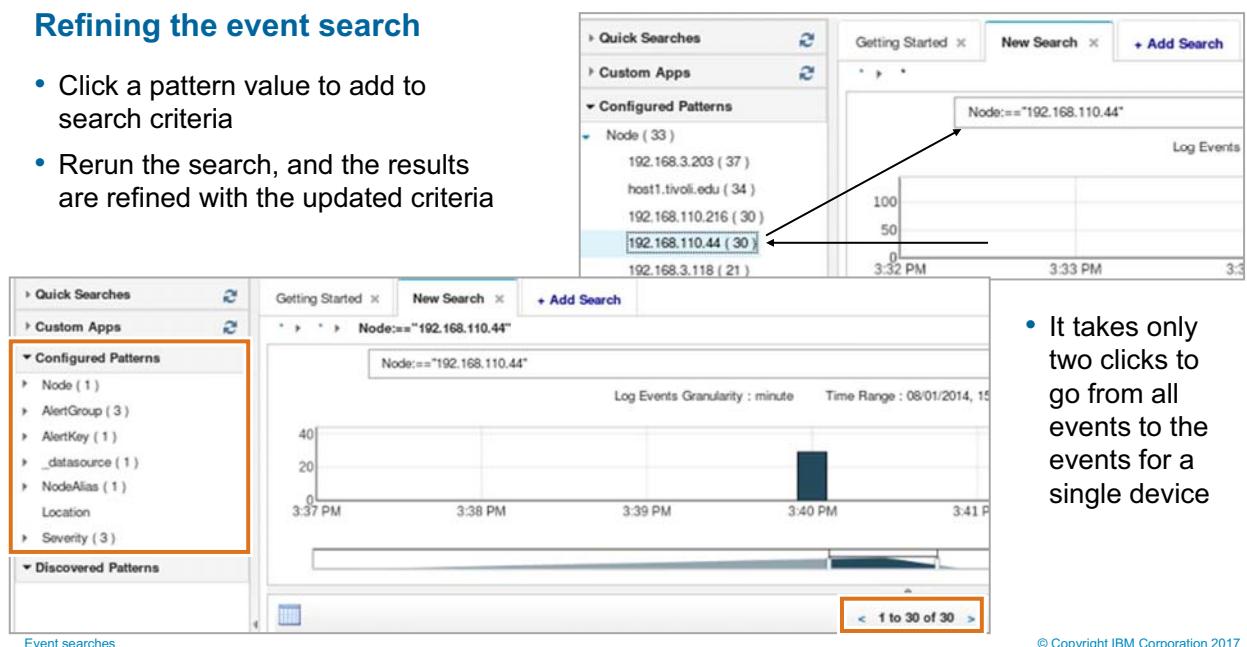


Figure 16-5. Searching events

## Refining the event search

- Click a pattern value to add to search criteria
- Rerun the search, and the results are refined with the updated criteria



- It takes only two clicks to go from all events to the events for a single device

© Copyright IBM Corporation 2017

Figure 16-6. Refining the event search

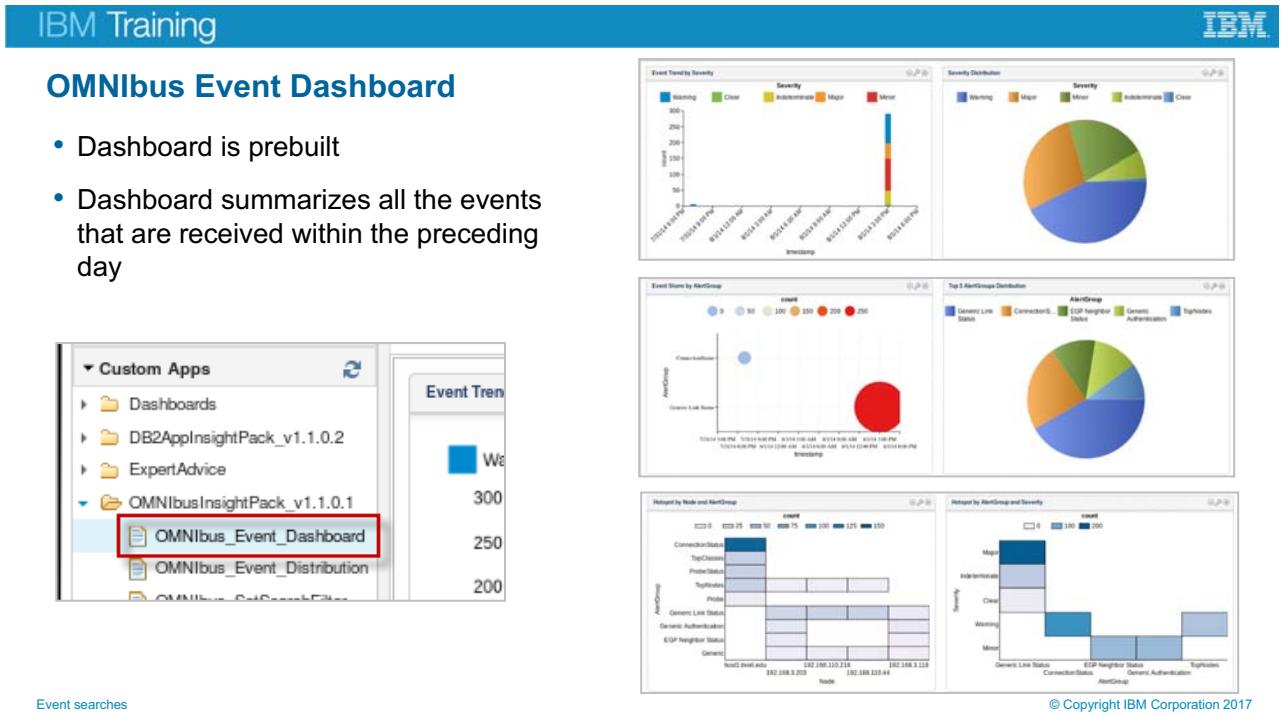
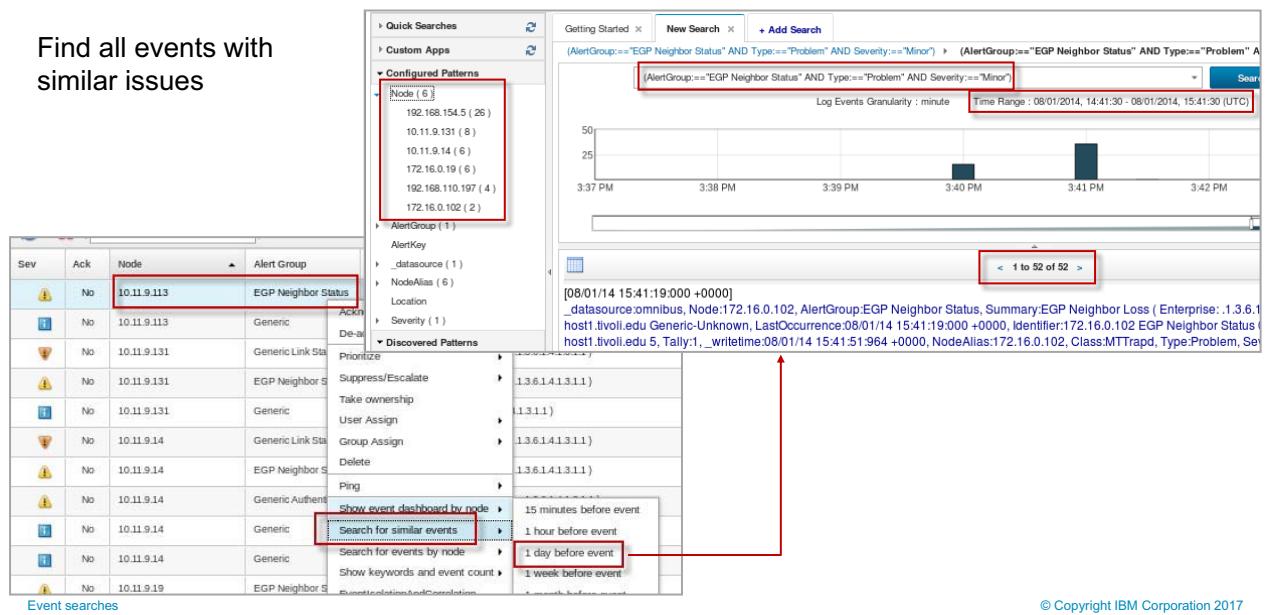


Figure 16-7. OMNIbus Event Dashboard

IBM Training

## Search for similar events

Find all events with similar issues



© Copyright IBM Corporation 2017

Figure 16-8. Search for similar events

## Search for all events on the same device

The screenshot shows the IBM Event Viewer interface. On the left, the 'Event Viewer' window displays a list of events for 'host1.tivoli.edu'. A red box highlights the 'host1.tivoli.edu' entry in the list. On the right, a 'Getting Started' search interface is shown with the query '(Node==>host1.tivoli.edu) & (Node==>host1.tivoli.edu)'. The search results show a timeline from 3:39 PM to 3:43 PM with 555 events. A red box highlights the search query in the search bar. Another red box highlights the 'Time Range' field set to '08/05/2014, 15:09:49 - 08/05/2014, 16:09:49 (UTC)'. The bottom pane shows the event details, with a red box highlighting the first event entry.

© Copyright IBM Corporation 2017

Figure 16-9. Search for all events on the same device

## Exercise: Event searches in Netcool Operations Insight

Event searches

© Copyright IBM Corporation  
2017

Figure 16-10. Exercise: Event searches in Netcool Operations Insight

Complete the exercise for this unit.

## Unit summary

After you complete this unit, you should be able to do these tasks:

- Describe how Log Analysis processes events
- Describe the features of event search
- Explain how to conduct an event search

[Event searches](#)

© Copyright IBM Corporation 2017

*Figure 16-11. Unit summary*

# Unit 17. Log Analysis basics

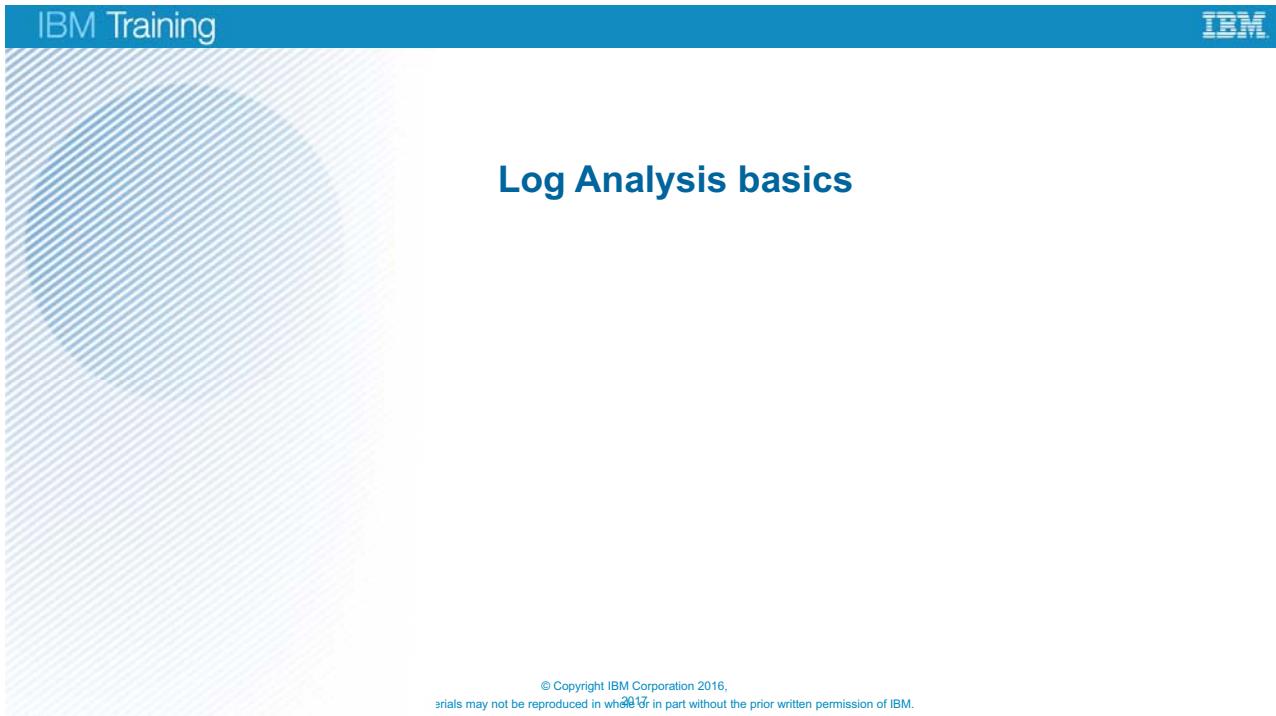


Figure 17-1. Log Analysis basics

## Estimated time

00:45

## Overview

This unit shows students how to use basic Log Analysis in Netcool Operations Insight. Completion of this unit in the Tivoli Network Manager 4.2 Operations and Administration course is optional. Log Analysis is a feature that is included in Netcool Operations Insight. Tivoli Network Manager runs as a component within Netcool Operations Insight. The analytics engine can read inventory and discovery data from the NCIM database. This ability provides the user with a powerful capacity to search the network inventory database to find all instances of devices that share a similar characteristic.

If you have a network problem that occurs only on devices with a specific iOS version, what can you do? You can run a topology search with the analytics engine to determine what other devices run that version of operating system. You can then determine what other network devices need software upgrades to prevent this problem in the future.

You can also right-click an event and have the analytics engine find every log file entry or every other instance of this event in a historical database and show it to you. You can then graph the

number of these incidences over time or analyze the frequency of the events. All of these things are possible with the log analytics engine.

## Objectives

After you complete this unit, you should be able to do the following tasks:

- Describe the features of Log Analysis
- Explain how log files are processed
- Explain how log data is searched
- Describe how to create graphs and custom dashboards
- Explain the functions of an insight pack

*Figure 17-2. Objectives*

## Collect and process log data

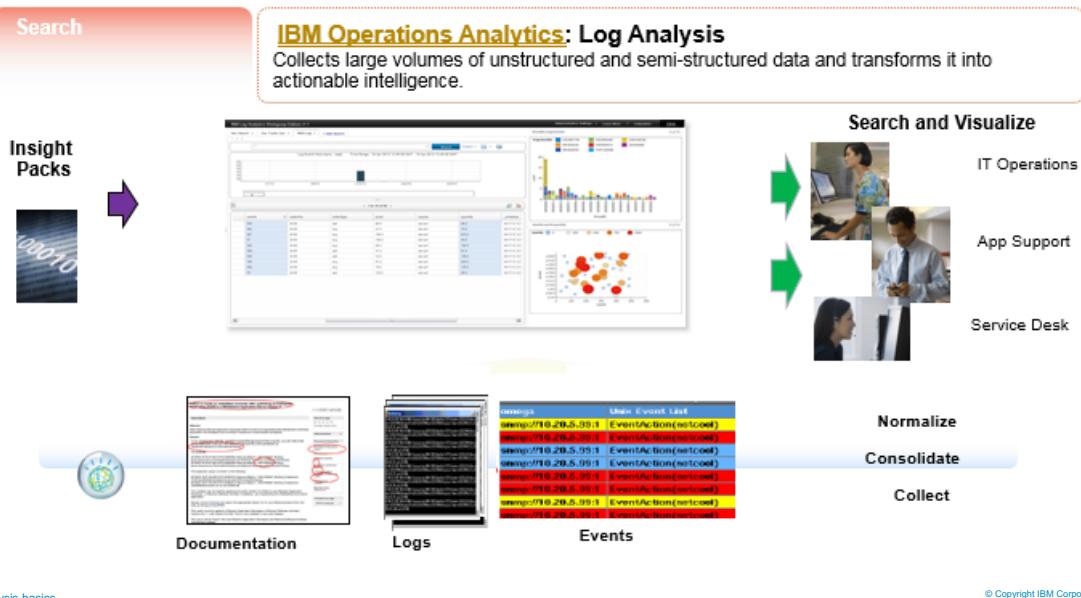
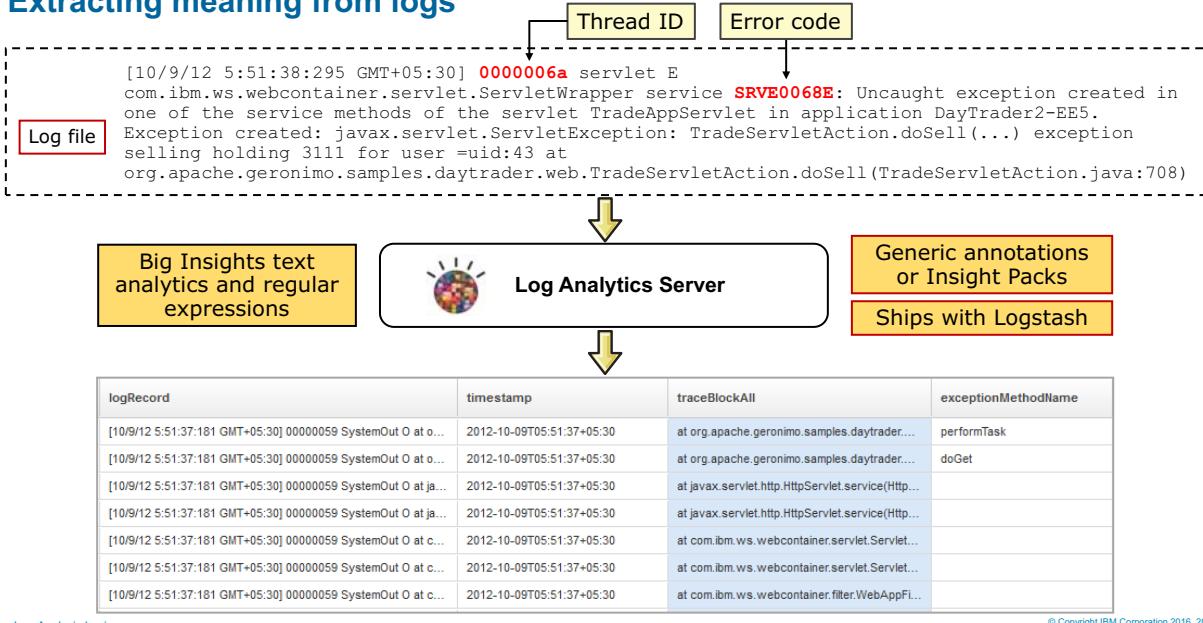


Figure 17-3. Collect and process log data

Netcool Operations Insight includes an analytics engine that is preconfigured to parse multiple formats of textual information from different types of logs. Also, you can purchase more Insight Packs to extend the abilities of the analytics engine to search and parse other types of data. Some products within the Netcool Operations Insight suite often come with an included Insight Pack. For example, the DB2 Insight Pack enables the analytics engine to parse and search through DB2 database tables.

## Extracting meaning from logs



Log Analysis basics

© Copyright IBM Corporation 2016, 2017

Figure 17-4. Extracting meaning from logs

When you view results from the analytics engine, you can look at them in two different ways. The default view shows a string of characters from a log file or database that contains the relevant fields that you selected. Reading through this string can help you determine whether the event is relevant to your situation. However, you can select an icon to view the same results in a tabular format. This format organizes the data stream into fields with field names that serve as the headers of columns in a table.

## Simple chart and dashboard creation

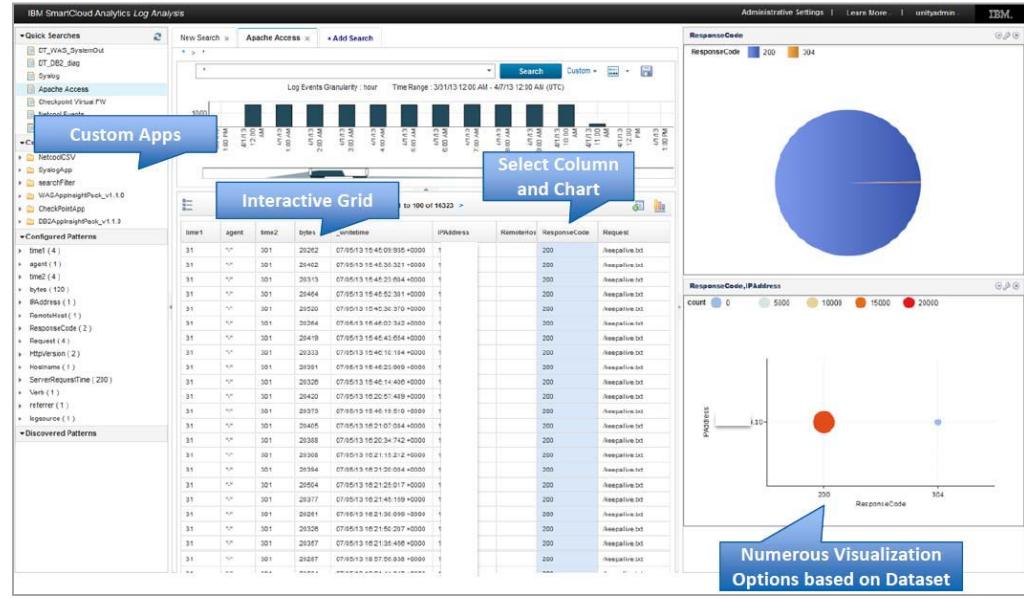


Figure 17-5. Simple chart and dashboard creation

You can create dashboards and charts inside Netcool Operations Insight.

- To create a dashboard, you start with a blank page and drag widgets onto the page.
  - a. Each widget has a different function. For example, one widget pulls events from the OMNIbus event list. Another widget can display a chart from a separate database. Other widgets can pull in weather information.
  - b. After you drag and position the widgets onto the blank page, you save the page.
  - c. When you click the icon or menu item that represents the page, it loads the charts and tables that you configured for that page.
- Creating charts is an even easier task.
  - a. You select characteristics or enter search text.
  - b. From the available results, you select a column of data and a chart style.
  - c. You click a button to create the chart, and it is immediately available. If you want the chart to be persistent, you can save a copy of the chart.

It usually takes no more than three clicks to show a chart inside the Netcool Operations Insight GUI.

## Compare messages from different log sources

| message                                                    | timestamp                   | <input checked="" type="checkbox"/> | msgclassifier | _datasource    |
|------------------------------------------------------------|-----------------------------|-------------------------------------|---------------|----------------|
| ADM5502W The escalation of "397" locks on table "ADMI..."  | 04/16/13 12:04:14:000 +0000 |                                     | ADM5502W      | db2diag.log    |
| ADM5502W The escalation of "396" locks on table "ADMI..."  | 04/16/13 12:04:32:000 +0000 |                                     | ADM5502W      | db2diag.log    |
| ADM5502W The escalation of "398" locks on table "ADMI..."  | 04/16/13 12:04:52:000 +0000 |                                     | ADM5502W      | db2diag.log    |
| ADM5502W The escalation of "410" locks on table "ADMI..."  | 04/16/13 12:05:13:000 +0000 |                                     | ADM5502W      | db2diag.log    |
| Connection not available while invoking method createOr... | 04/16/13 12:05:13:537 +0000 |                                     | J2CA0045E     | SystemOut1.log |
| Connection not available while invoking method createOr... | 04/16/13 12:05:13:537 +0000 |                                     | J2CA0045E     | SystemOut2.log |

Figure 17-6. Compare messages from different log sources

In this example, the analytics engine returns results from three different data sources. The first data source is a DB2 diagnostic log. The remaining two sources are error logs from the Jazz for Service Management dashboard GUI server. This example illustrates the ability of the analytics engine to extract data from multiple resources at the same time.

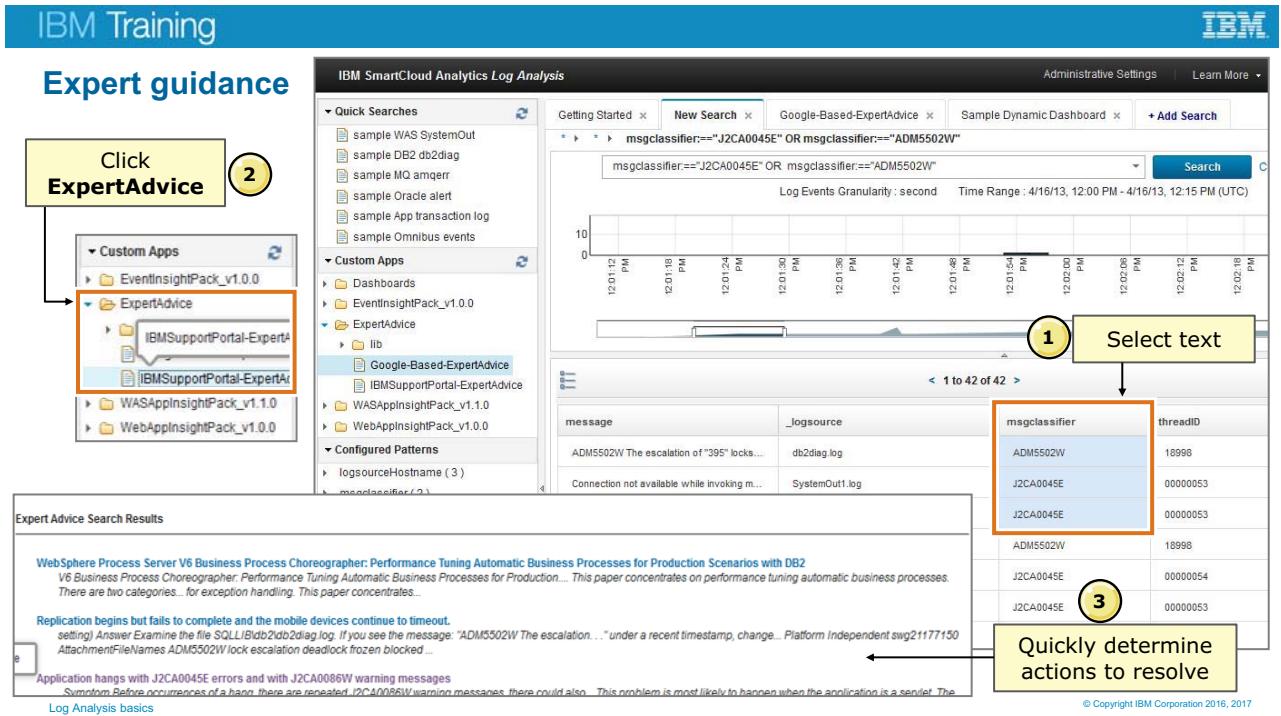


Figure 17-7. Expert guidance

Each Insight Pack includes a feature that is called Expert Advice. To use this feature, you do the following steps:

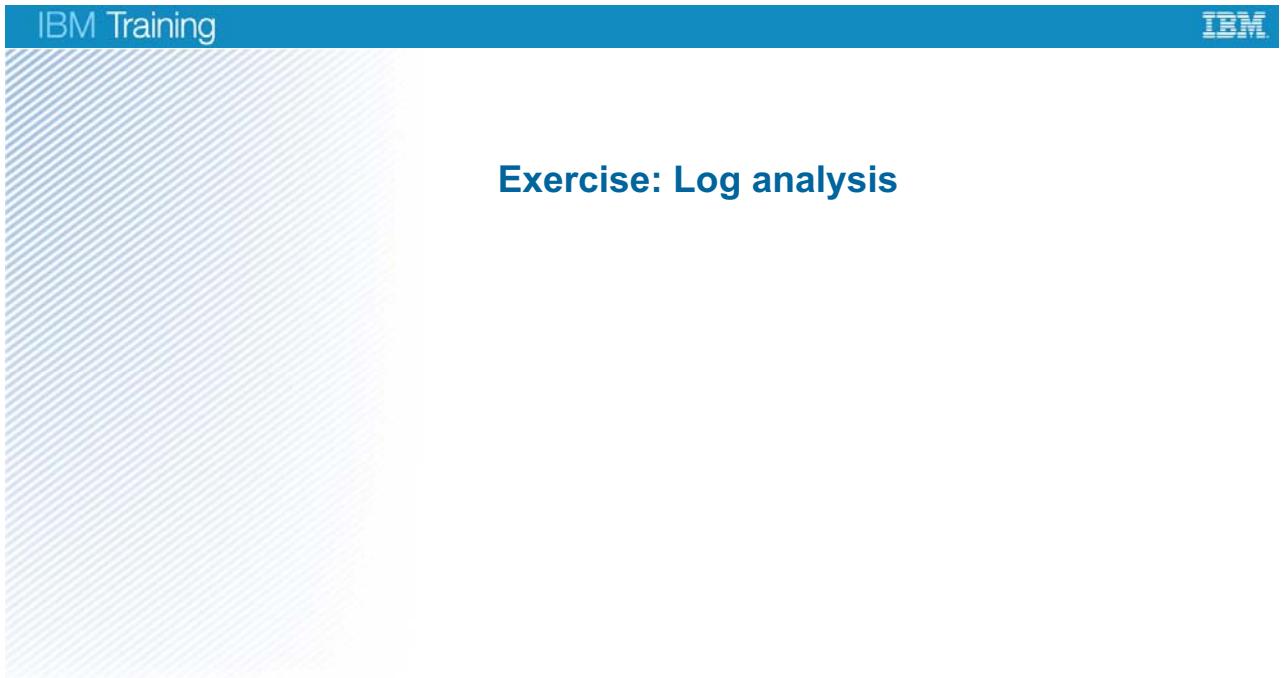
1. Select text from a column that shows log or event data.
2. In the **Custom Apps** menu tree, click the ExpertAdvice that is associated with your selected text or event data.
3. The results window shows ways that you solved similar problems in the past by using data from your trouble-ticketing system or historical event database. It can also show hyperlinks to external articles for your review. You can then use this information to select a proper course of action to resolve your problem.

## Insight Packs provide content

- Insight Packs installation instructions explain how to annotate and split log files
- Insight Packs provide custom applications and dashboards
- Insight Packs are for a particular technology or application
- Example Insight Packs
  - IBM DB2 for `db2diag.log` files
  - IBM WebSphere for **SystemOut**, **SystemErr**, and **trace**
  - Web access for Apache, IBM HTTP Server, Tomcat, and JBoss log files
  - Syslog for syslog message files
  - More

*Figure 17-8. Insight Packs provide content*

Some products include an Insight Pack when you purchase the product. Other Insight Packs are available for purchase.



*Figure 17-9. Exercise: Log analysis*

Complete the student exercise for this unit.

While this unit is regarded as optional for this course, the student exercise takes only a few minutes to complete. It provides valuable experience for those individuals who are unfamiliar with the analytics engine in Netcool Operations Insight.

## Unit summary

After you complete this unit, you should be able to do the following tasks:

- Describe the features of Log Analysis
- Explain how log files are processed
- Explain how log data is searched
- Describe how to create graphs and custom dashboards
- Explain the functions of an insight pack

*Figure 17-10. Unit summary*

# Unit 18. Seasonal events

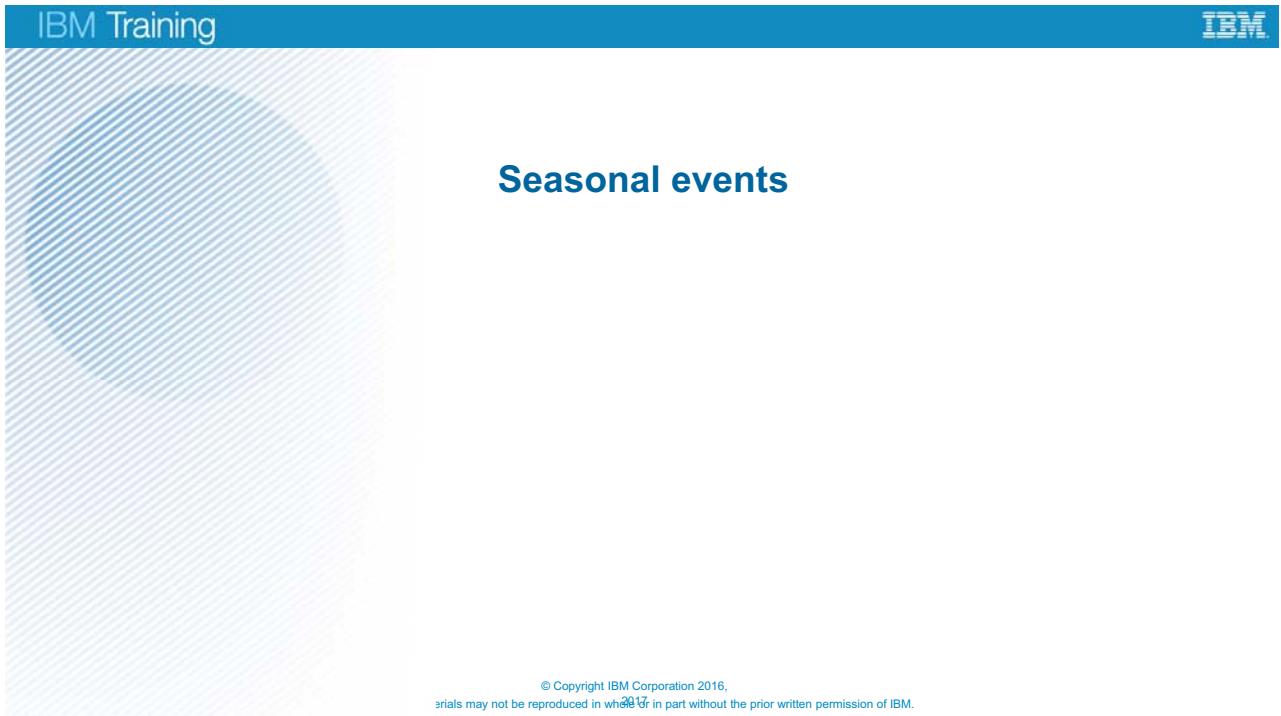


Figure 18-1. Seasonal events

## Estimated time

00:20

## Overview

Netcool Operations Insight can look for events or groups of events that tend to happen at regular intervals. These events are called seasonal events. This unit shows how to identify seasonal events so that you can begin to understand their correlation to activities and processes in your network environment.

## How you will check your progress

Complete the student exercise for this unit.

## Objectives

After you complete this unit, you should be able to do the following tasks:

- Describe event seasonality
- Explain how to conduct a seasonal events analysis
- Explain how to examine the results of the analysis
- Explain how to create a seasonal event rule

*Figure 18-2. Objectives*

## Seasonal event function

- Netcool Operations Insight analytics engine can discover events that occur in a non-random pattern over time
- The result is a summary of events that are likely to be seasonal, including a confidence score
- Events are analyzed in four different time periods:
  - Minute of hour
  - Hour of day
  - Day of week
  - Day of month

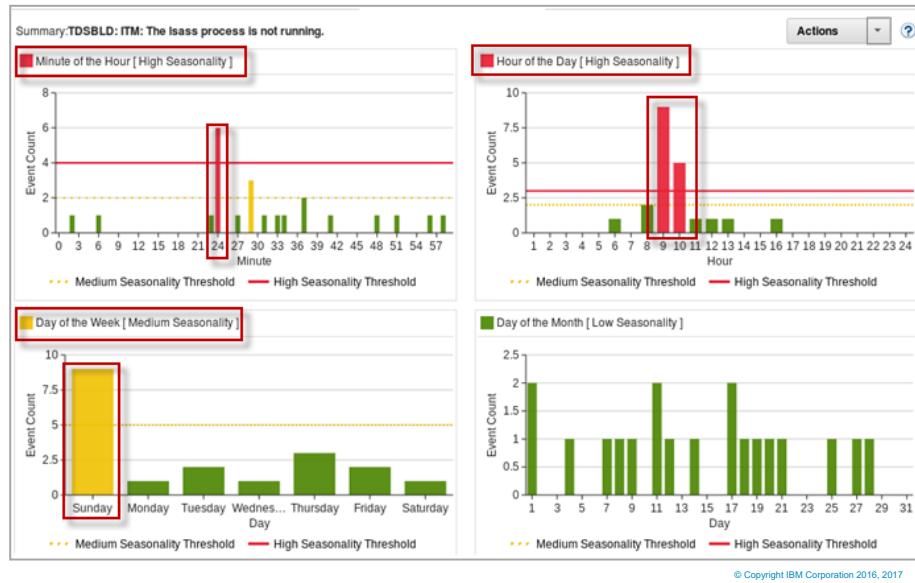


Figure 18-3. Seasonal event function

Every network is unique. But in most networks, certain events tend to happen at certain times. For example, running a large report that uses databases across the network might result in the following kinds of events:

- Increase in CPU and memory usage on certain servers
- Packets dropped, retransmission requests, and frames retransmitted because of incompatibility in the maximum transmission unit (MTU) size of devices along the data path

Recognizing that these events occur at certain times of the month, times of the day, or days of the week can provide valuable information necessary to reschedule tasks for better load balancing. It can also help support personnel to know that certain bursts of activity are normal and are not an aberration that requires their attention. The analysis engine inside Netcool Operations Insight can recognize event seasonality so that you can take appropriate action.

## How seasonality is determined

- Seasonality is determined by counting event observations in time period bins
- If the **Summary** field is the same, events are identified as unique
- The number of actual observations in each time period bin is compared with a uniform distribution of events
- The difference is a measure of probable seasonality
- An **observation** is a count of whether the event arrived in each bin

Figure 18-4. How seasonality is determined

The screenshot shows the 'Configure Analytics' dialog box with various fields filled out. The 'Name' field contains 'SE\_Workshop'. The 'Event identity' dropdown is set to 'IDENTIFIER'. The 'Date range' dropdown is set to 'Relative' with '60 Months' selected. The 'Start date' and 'End date' fields both show '9/30/2015'. The 'Run every' field has 'Months' selected. The 'Filter' field contains the expression 'Severity >= 3 and AlertGroup like 'SYM%''. Under 'Select the analytics type you want to run', the 'Seasonal event analytics' checkbox is checked. At the bottom, there are 'Save', 'Save & Run', and 'Cancel' buttons. To the right, the 'Configure Analytics' menu in the Insights sidebar is shown, with the 'Save & Run' button highlighted by a red box.

Figure 18-5. Creating a configuration

Looking for event seasonality is a simple task.

- From the Dashboard Application Services Hub (DASH), choose the Configure Analytics menu option. Click the icon to create a new analysis. Enter the following values:
  - Name
  - Source of event data
  - An event filter that is expressed in terms of time
- Then, check the box for **Seasonal event analytics**.
- Click **Save & Run**.

## IBM Training



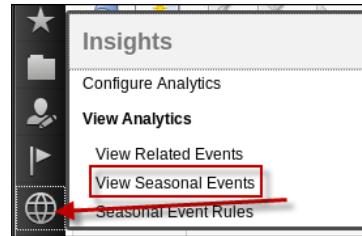
### Run a configuration

- The configuration is submitted for processing

| Name          | Event Identity | Seasonal Status | Related Event Status | Start Time          | End Time            | Seasonality Phase      | Seasonal Phase Progress | Related Event Phase    | R<br>E<br>P<br>P |
|---------------|----------------|-----------------|----------------------|---------------------|---------------------|------------------------|-------------------------|------------------------|------------------|
| Sample Config | SUMMARY        |                 |                      | Jun 16, 2015 7:35:4 | Sep 16, 2015 7:35:4 | Saved, Waiting for use | 0%                      | Saved, Waiting for use |                  |
| SE_Workshop   | IDENTIFIER     |                 |                      | Sep 30, 2010 1:19:2 | Sep 30, 2015 1:19:2 | Queued, Waiting to run | 0%                      | Not Enabled            |                  |

- After the analysis finishes, you see the following result

|               |            |  |  |                     |                     |             |  |          |
|---------------|------------|--|--|---------------------|---------------------|-------------|--|----------|
| Test          | IDENTIFIER |  |  | Jun 16, 2015 7:35:4 | Sep 16, 2015 7:35:4 | Completed   |  | 600      |
| ProblemEvents | IDENTIFIER |  |  | Jun 16, 2015 7:35:4 | Sep 16, 2015 7:35:4 | Not Enabled |  | Saved, W |
| SE_Workshop   | IDENTIFIER |  |  | Sep 30, 2010 1:19:2 | Sep 30, 2015 1:19:2 | Completed   |  | Not Enab |



© Copyright IBM Corporation 2016, 2017

Figure 18-6. Run a configuration

## Viewing seasonal events analysis results

- The analysis identifies potential seasonal events

| Configuration |  | Event Count | Node                 | Summary                           | Alert Group    |
|---------------|--|-------------|----------------------|-----------------------------------|----------------|
| ALL           |  | 154         | ducttape.tivlab.rule | TDSBLD: ITM: The lsass process    | ITM_NT_Process |
| SE_Workshop   |  | 77          | s3w2k.tivlab.aust    | TDSBLD: ITM: The services process | ITM_NT_Process |
| Workshop2     |  | 77          | fw2w2k.tivlab.aust   | TDSBLD: ITM: The lsass process    | ITM_NT_Process |
|               |  |             | fw3w2k.tivlab.aust   | TDSBLD: ITM: The lsass process    | ITM_NT_Process |

- Right-click an event and select **Show Seasonal Event Graphs**

| Node                 | Summary                           | Alert Group    | Reviewed By | Confidence Level |
|----------------------|-----------------------------------|----------------|-------------|------------------|
| ducttape.tivlab.rule | TDSBLD: ITM: The lsass process    | ITM_NT_Process |             | 100%             |
| s3w2k.tivlab.aust    | TDSBLD: ITM: The services process | ITM_NT_Process |             | 100%             |
| fw2w2k.tivlab.aust   | TDSBLD: ITM: The lsass process    | ITM_NT_Process |             | 100%             |
| fw3w2k.tivlab.aust   | TDSBLD: ITM: The lsass process    | ITM_NT_Process |             | 100%             |
| rtpbwin1.tivlab.rule | TDSBLD: ITM: The lsass process    | ITM_NT_Process |             | 100%             |
| rtpbwin3b.tivlab.ral | TDSBLD: ITM: The lsass process    | ITM_NT_Process |             | 100%             |

Seasonal events

© Copyright IBM Corporation 2016, 2017

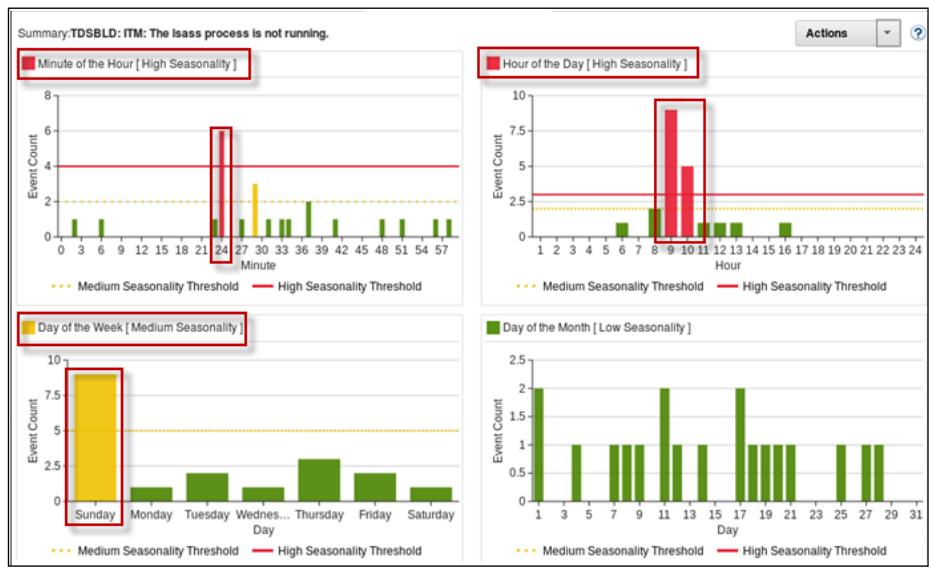
Figure 18-7. Viewing seasonal events analysis results

The analysis shows a potential grouping of seasonal events. It also shows a confidence level that indicates how likely it is that these events occur with predictable regularity. You can then create a rule that summarizes the seasonal event groupings with a message that can be sent to the Netcool/OMNIbus ObjectServer.

## Sample seasonal reports

- The analysis identifies potential seasonal time frames
- The example event analysis indicates:
  - High confidence for minute of the hour
  - High confidence for hour of the day
  - Medium confidence for day of the week

The event repeats every Sunday, at 24 minutes past the hours of 9 and 10 AM



Seasonal events

© Copyright IBM Corporation 2016, 2017

Figure 18-8. Sample seasonal reports

The automatically created graphs indicate the confidence level that the graph truly indicates seasonal events.

- In this example, the graphs with a red check box indicate a high confidence level.
- The graph with the yellow check box indicates a medium or marginal confidence.
- The graph with a green check box indicates a low confidence level or no true correlation regarding the timing of the events.

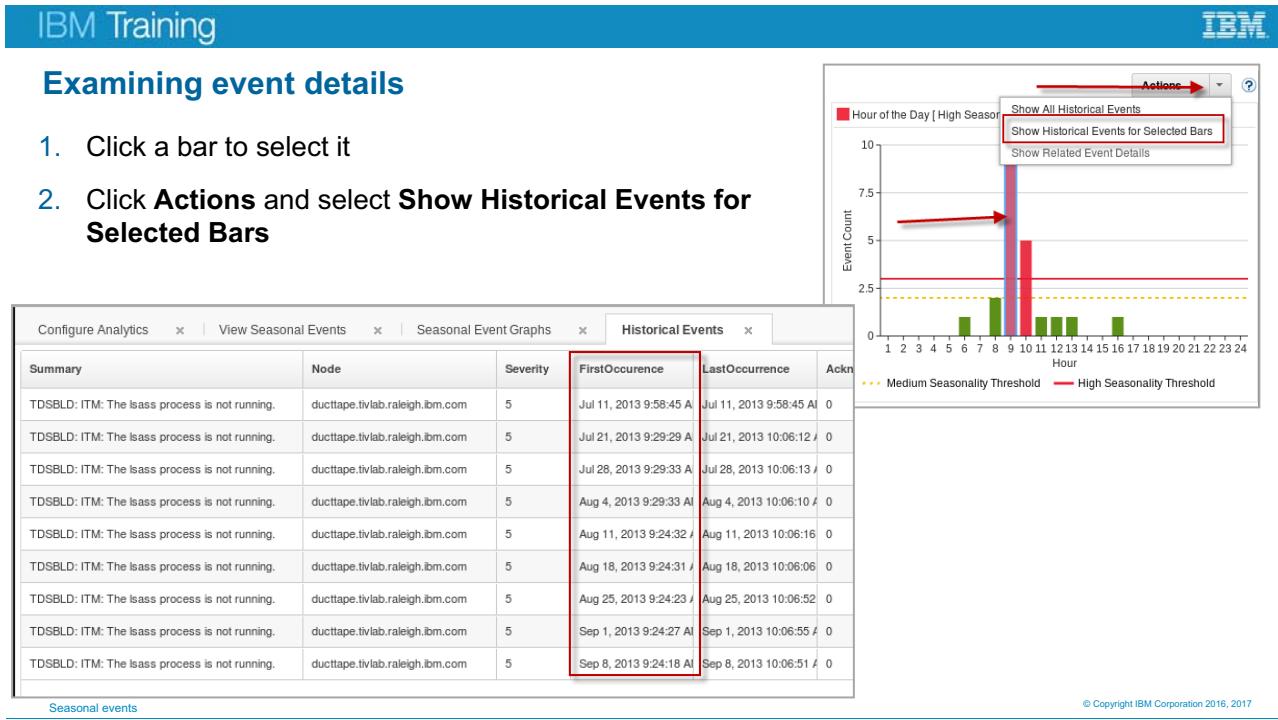


Figure 18-9. Examining event details

Clicking the graphical results opens a window that shows the historical events and their time stamps that are associated with the portion of the graph you clicked.

## Seasonal event rules

- The administrator reviews the analysis and can choose to deploy an automated action for a seasonal event
- The actions are implemented by Tivoli Netcool/Impact
- You can use seasonal event rules for these tasks:
  - Apply actions to suppress an event
  - Modify or enrich an event
  - If the event occurs when it is not expected, the event is shown in the event list
  - If the event does not occur when it is expected, that is also shown

---

Seasonal events

© Copyright IBM Corporation 2016, 2017

Figure 18-10. Seasonal event rules

Knowing that events occur with regular seasonality and a high confidence level can be helpful to your support personnel.

You can create a rule that can suppress multiple events and add a synthetic event to the Netcool/OMNIbus ObjectServer.

As soon as the rule is deployed, it can do the following actions:

- Create an event if the expected seasonal events do not occur as expected
- Create an event if the events occur at an unexpected time
- Suppress the expected events
- Suppress the expected events but insert a synthetic event that shows that the seasonal grouping occurred as expected

## Creating a seasonal event rule

1. Right-click an event and select **Create Rule**
2. Configure the time frame for the rule
3. Select the type of action
4. Click **Deploy**

| Node                 | Summary                             | Alert Group    | Reviewed By |
|----------------------|-------------------------------------|----------------|-------------|
| ducttape.tivlab.rule | TDSBLD: ITM: The Isass process i... | ITM_NT_Process |             |
| s3w2k.tivlab.aust    | TD                                  | Process        |             |
| fw2w2k.tivlab.aust   | TD                                  | Process        |             |
| fw3w2k.tivlab.aust   | TD                                  | Process        |             |

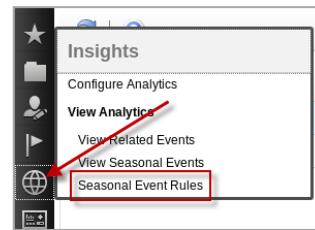
The screenshot shows the 'Create Rule' dialog box. The 'Rule Name' field is set to 'SE\_Workshop\_Rule'. Under 'Event Selection', there is a list of selected events: 'TDSBLD: ITM: The Isass process i...'. A red box highlights the 'Day of Week' dropdown, which is set to 'Wednesday [Low]'. Below this, under 'Time Condition(s)', the 'AND' radio button is selected. To the right, there is a section titled 'Actions When Event(s) Does Not Occur in Specified Time Window(s)' with a 'Create Event...' button.

Figure 18-11. Creating a seasonal event rule

This graphic illustrates that deploying a seasonal event rule is a simple four-step process.

## Seasonal event rule results

- Select **Seasonal Event Rules**
- Netcool/Impact collects statistics for every rule



| Seasonal Event Rules |            |                  |
|----------------------|------------|------------------|
| Configuration        | Rule Count | Rule Name        |
| All                  | 1          | SE_Workshop_Rule |
| SE_Workshop          | 1          |                  |

| Rule Name        | Last Run                | Deployed                | Suppressed Events | Unsuppress Events | Enriched/M Events | Generated Events on Non-occurr |
|------------------|-------------------------|-------------------------|-------------------|-------------------|-------------------|--------------------------------|
| SE_Workshop_Rule | Sep 30, 2015 3:53:09 PM | Sep 30, 2015 3:48:52 PM | 0                 | 0                 | 0                 | 1                              |

© Copyright IBM Corporation 2016, 2017

Figure 18-12. Seasonal event rule results

For any deployed rule, Tivoli Netcool/Impact maintains statistics on how many events match the rule. It also shows how many events were suppressed, how many events run suppressed, and how many events were enriched with a synthetic event.

## Event rule result

Example of an event that is created when a seasonal event does not occur when expected

| Sev | Ack | Node                            | Alert Group | Summary                                                                            |
|-----|-----|---------------------------------|-------------|------------------------------------------------------------------------------------|
|     | No  | ducttape.tivlab.raleigh.ibm.com | SE_Auto     | NON-OCCUR_1_{\"IDENTIFIER\":\"NT_BP_ProcMissing_Critical:Primary:DUCTTAPE:NT:is\"} |

Figure 18-13. Event rule result

© Copyright IBM Corporation 2016, 2017

## Exercise: Find related events

---

Seasonal events

© Copyright IBM Corporation 2017

*Figure 18-14. Exercise: Find related events*

Complete the student exercise for this unit.

While this unit is optional for the Tivoli Network Manager 4.2 course, this exercise takes only a few minutes. It provides valuable training for those individuals who have little or no experience with Netcool Operations Insight.

## Unit summary

After you complete this unit, you should be able to do the following tasks:

- Describe event seasonality
- Explain how to conduct a seasonal events analysis
- Explain how to examine the results of the analysis
- Explain how to create a seasonal event rule

Figure 18-15. Unit summary

# Unit 19. Related events

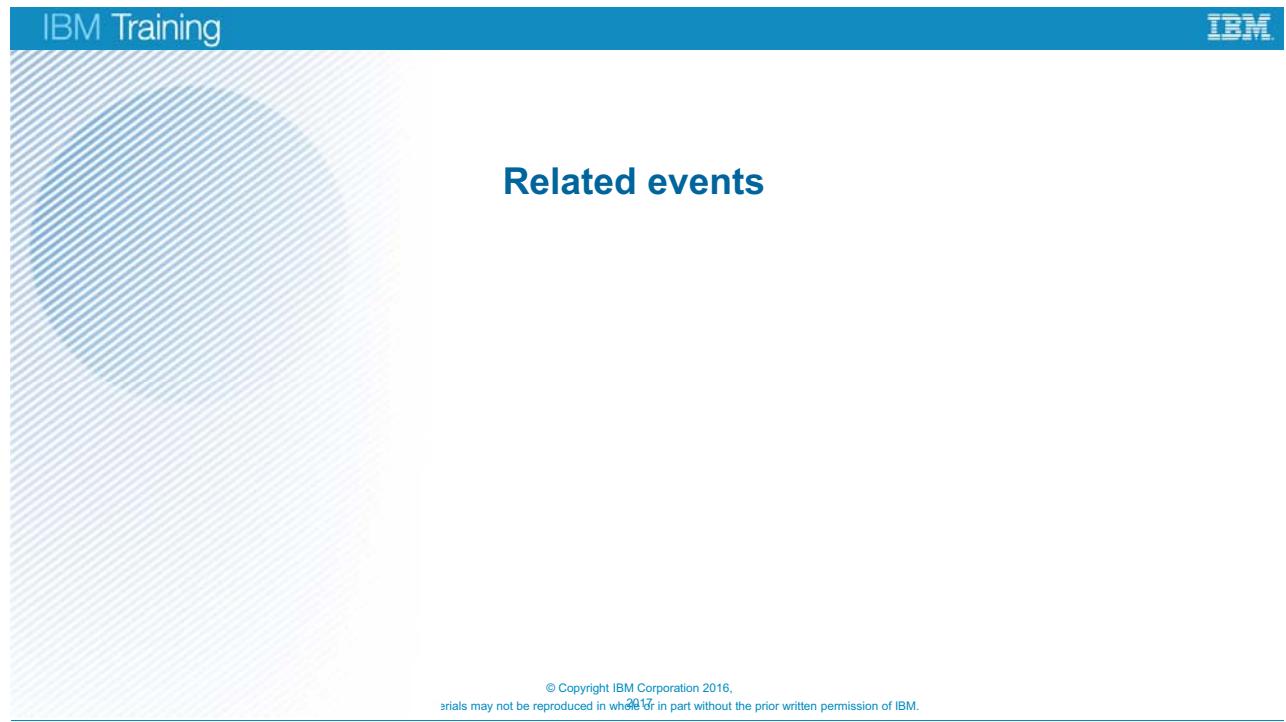


Figure 19-1. Related events

## Estimated time

00:20

## Overview

This unit shows how Netcool Operations Insight enables the user to search for all the events on a device and generate graphs or tables based on the search results.

## How you will check your progress

- Complete the student exercise for this unit.

## Objectives

After you complete this unit, you should be able to do the following tasks:

- Describe the steps in the related events workflow
- Explain how to conduct a related events analysis
- Explain how to examine the results of the analysis
- Explain how to deploy a group
- Explain how to view grouped events

*Figure 19-2. Objectives*

## Feature overview

- Event Analytics looks for these types of events:
  - Related events
  - Seasonal events
- Related events
  - Correlates unknown related events and shows them grouped as parent-child events in the Event Viewer
  - Results in reduction of actionable events that are presented to the operator
  - Done by looking for relationship between events in the historical event database
- Seasonal events
  - Discovers events that occur in a nonrandom pattern over time
  - The result is a summary of events that are likely to be seasonal, including a confidence score
  - Done by analyzing events in the historical event database

[Related events](#)

© Copyright IBM Corporation 2016, 2017

*Figure 19-3. Feature overview*

In the preceding unit, you learned that seasonal events analysis looks for an increase in the number of events in predictable time patterns. In this unit, you learned that the analytics engine looks for certain events that tend to occur together. For example, an incompatibility in the maximum transmission unit (MTU) size can result in the following related events in your network:

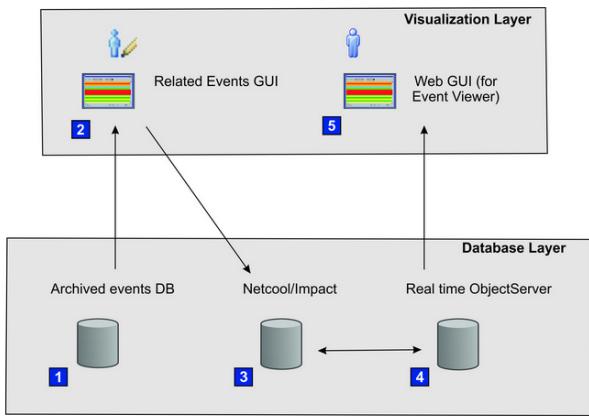
- CPU usage increase
- System memory usage increase
- Increase in dropped packets
- Retransmission requests

The analytics engine can determine that multiple occurrences of these event groupings occurred in close time proximity to one another. It can identify that the events are related and assigned a mathematical probability that indicates the level of certainty that the events are indeed related.

The administrator can choose to deploy a rule that suppresses these events and inserts a synthetic parent event. It can also designate one event as a parent event and the other events as children events. In the Event Viewer GUI, the children events are shown as indented underneath the parent event. This way of displaying related events helps support personnel focus on the most important event that is likely causing the other events to occur. In the preceding example, a new event that indicates an MTU size mismatch might become the parent event while the other listed events become children events.

## Related events workflow

1. Netcool/OMNIbus archives real-time events to a database
2. Analytics engine identifies event patterns and their frequency of recurrence and then displays them in the Related Events GUI
3. The administrator can do these tasks:
  - Deploy the identified grouping so that Netcool/Impact looks for the same event groupings in the future
  - Watch for further event groupings to verify that the pattern is valid
  - Ignore the grouping
4. Deployed groupings result in the creation of Netcool/Impact policies that monitor events in real time
5. Operators focus on a reduced number of events by working the identified parent event (with the children events that are indented underneath the parent)



Related events

© Copyright IBM Corporation 2016, 2017

*Figure 19-4. Related events workflow*

The workflow that is shown on this slide indicates that two prerequisite components are available to search for related events. First, events must be archived to a historical database. Netcool Operations Insight includes a license to use a DB2 database only for storing historical events. Second, Tivoli Netcool/Impact must be installed. Netcool/Impact is included as a part of Netcool Operations Insight.

IBM Training 

## Creating a configuration

1. Log in to Dashboard Application Services Hub
  - User requires **ncw\_analytics\_admin** role
2. Select **Configure Analytics**
3. Create an analysis
4. Enter the following values:
 

|                  |                    |
|------------------|--------------------|
| ▪ Name           | ▪ Events from last |
| ▪ Event identity | ▪ Filter           |
5. Select **Related event analytics**
6. Click **Save & Run**



**Configure Analytics**

|                                                                                                                                                              |                                                            |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <b>General</b>                                                                                                                                               | <b>Related Events</b>                                      |
| Name: <input type="text" value="RE_Workshop"/>                                                                                                               | Event identity: <input type="button" value="IDENTIFIER"/>  |
| Date range:                                                                                                                                                  | <input type="radio"/> Relative <input type="radio"/> Fixed |
| Events from last:                                                                                                                                            | <input type="text" value="60"/> Months                     |
| Start date: <input type="text" value="9/30/2015"/>                                                                                                           | End date: <input type="text" value="9/30/2015"/>           |
| Run every:                                                                                                                                                   | <input type="text"/> Months                                |
| Filter: <input type="text" value="Severity &gt;= 3 and AlertGroup like %TTT%"/>                                                                              | <input type="button" value="AlertGroup"/>                  |
| * Select the analytics type you want to run<br><input type="checkbox"/> Seasonal event analytics <input checked="" type="checkbox"/> Related event analytics |                                                            |
| <input type="button" value="Save"/> <input type="button" value="Save &amp; Run"/> <input type="button" value="Cancel"/>                                      |                                                            |

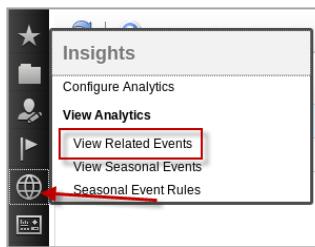
Figure 19-5. Creating a configuration

The steps necessary to look for related events are identical to the steps to look for seasonal events with one exception. Instead of clicking the **Seasonal events analytics** check box, click the **Related events analytics** check box.

## Run a configuration

| Name                 | Event Identity | Season Status | Related Event Status | Start Time         | End Time           | Seasonality Phase       | Seasonal Phase Progress | Related Event Phase          | Related Event Phase Progress | Schedule |
|----------------------|----------------|---------------|----------------------|--------------------|--------------------|-------------------------|-------------------------|------------------------------|------------------------------|----------|
| Sample Configuration | SUMMARY        |               |                      | Jun 16, 2015 7:35: | Sep 16, 2015 7:35: | Saved, Waiting for user | 0%                      | Saved, Waiting for user      | 0%                           | No       |
| ProblemEvents        | IDENTIFIER     |               |                      | Jun 16, 2015 7:35: | Sep 16, 2015 7:35: | Not Enabled             | 0%                      | Saved, Waiting for user      | 0%                           | No       |
| RE_Workshop          | IDENTIFIER     |               |                      | Sep 30, 2010 6:25: | Sep 30, 2015 6:25: | Not Enabled             | 0%                      | Step 1 of 5: Retrieving data | 0%                           | No       |

- The configuration is submitted for processing
- After the analysis is done, it shows the result
- Click View Related Events**



| Seasonality Phase       | Seasonal Phase Progress | Related Event Phase     | Related Event Phase Progress | Schedule |
|-------------------------|-------------------------|-------------------------|------------------------------|----------|
| Saved, Waiting for user | 0%                      | Saved, Waiting for user | 0%                           | No       |
| Not Enabled             | 0%                      | Saved, Waiting for user | 0%                           | No       |
| Not Enabled             | 0%                      | Completed               | 100%                         | No       |

Related events

© Copyright IBM Corporation 2016, 2017

Figure 19-6. Run a configuration

The length of time that analysis takes to run depends on the amount of historical event data that you have. A progress bar indicates whether the analysis is complete or still in progress.

## Viewing related events analysis results

- The analysis identifies potential event groups
- The group names show in the lower portion of the frame
- The number of events for each group is shown after the group name

| Configuration | Strength | Groups | Events | Node                             | Sub |
|---------------|----------|--------|--------|----------------------------------|-----|
| All           |          | 87     | 501    | tabantest.tivlab.raleigh.ibm.com | TD  |
| RE_Workshop   | Strong   | 87     | 501    | tabantest.tivlab.raleigh.ibm.com | TD  |

| Group Name    | Strength | Events | Instances | Reviewed |  |
|---------------|----------|--------|-----------|----------|--|
| RE_Workshop:0 | Strong   | 7      | 7         | No       |  |
| RE_Workshop:1 | Strong   | 3      | 21        | No       |  |

Related events

© Copyright IBM Corporation 2016, 2017

Figure 19-7. Viewing related events analysis results

For each set of related events, the number of events that seem to occur together and the number of instances in which they did occur together are shown. The administrator can then choose one event as the parent event and other events as children events. The analysis shows one event as the default parent event. However, the administrator can choose any of the related events to be the parent event.

## Examining event group details

- Event groups are identified by time spans
- The analysis looks for the same groups that occur in multiple spans
- The event details show each span that was found in the event archive
- The event details also show how many events occurred in each time period
- Events that occur in more periods receive a higher confidence rating within the group

| Events                   |     | Correlation Rule | Contains | Events    | Timeline              |
|--------------------------|-----|------------------|----------|-----------|-----------------------|
| Date and Time            |     |                  |          | Offset    | Time                  |
| Sep 14, 2012 12:32:17 PM | 157 |                  |          | -00:00:18 | Sep 14, 2012 12:51:51 |
| Sep 17, 2012 4:42:06 AM  | 157 |                  |          | -00:00:18 | Sep 14, 2012 12:51:51 |
| Sep 17, 2012 6:17:17 AM  | 157 | Yes              |          | -00:00:18 | Sep 14, 2012 12:51:51 |
| Sep 17, 2012 8:22:18 AM  | 157 | Yes              |          | -00:00:18 | Sep 14, 2012 12:51:51 |
| Sep 20, 2012 3:57:33 AM  | 157 | Yes              |          | -00:00:18 | Sep 14, 2012 12:51:51 |
| Sep 23, 2012 12:47:43 AM | 157 | Yes              |          | -00:00:18 | Sep 14, 2012 12:51:51 |

Related events

© Copyright IBM Corporation 2016, 2017

Figure 19-8. Examining event group details

## Deploying a configuration

- Right-click the group name, and select **Deploy**
- After the group is deployed, Netcool/Impact does these tasks:
  - Watches for new events
  - Enriches new events with relationship information
  - Collects statistics for the group

The screenshot shows the 'View Related Events' interface with the 'Active [1]' tab selected. A context menu is open over the 'RE\_Workshop:4' group, with the 'Deploy' option highlighted. The interface includes tabs for 'Configure Analytics' and 'View Related Events', and buttons for 'New [86]', 'Watched [0]', 'Active [1]', 'Expired [0]', and 'Archived [0]'. Below the tabs is a table with columns: Configuration, Timer Fired, Timer Fired in Last Month, Last Fired, Last Occurred I, Last Occurred II, and Last Occurred III. There are two rows: 'All' and 'RE\_Workshop'. At the bottom is another table for 'Group Name'.

| Group Name    | Timer Fired | Timer Fired in Last Month | Last Fired | Last Occurred I | Last Occurred II | Last Occurred III |
|---------------|-------------|---------------------------|------------|-----------------|------------------|-------------------|
| RE_Workshop:4 | 0           | 0                         |            | 0%              | 0%               | 0%                |

© Copyright IBM Corporation 2016, 2017

Figure 19-9. Deploying a configuration

The administrator has three choices on how to respond to the analysis for related events:

- Deploy a rule that recognizes future groupings of these related events and assigns one event as a parent event while other events are children events.
- Continue to watch for more instances of these event groupings before you decide whether to deploy a rule.
- Ignore the analysis so that Netcool/Impact does not look for further instances of these events occurring together.

## Viewing grouped events

1. Open a Web GUI Event Viewer
2. Select a view with the IBM Related Events relationship
3. Locate and expand a *parent* event
4. View the grouped events

| Node                            | ParentIdentifier       | AlertGroup               | Summary                                                                                                                         |
|---------------------------------|------------------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Related Events                  |                        | Synthetic Event - Parent | PROBABLE CAUSE: : Linux_BP_SpaceUsedPct_Critical([Disk_Used_Percent]>=95 AND FS_Type<>"nfs") ON chianti.tiv.lab.raleigh.ibm.com |
| chianti.tiv.lab.raleigh.ibm.com | Workshop2:1:1443560400 |                          | : Linux_BP_SpaceUsedPct_Critical([Disk_Used_Percent]>=95 AND FS_Type<>"nfs") ON chianti.tiv.lab.raleigh.ibm.com                 |
| chianti.tiv.lab.raleigh.ibm.com | Workshop2:1:1443560400 |                          | : Linux_BP_SpaceUsedPct_Critical([Disk_Used_Percent]>=95 AND FS_Type<>"nfs") ON chianti.tiv.lab.raleigh.ibm.com                 |
| chianti.tiv.lab.raleigh.ibm.com | Workshop2:1:1443560400 |                          | : Linux_BP_SpaceUsedPct_Critical([Disk_Used_Percent]>=95 AND FS_Type<>"nfs") ON chianti.tiv.lab.raleigh.ibm.com                 |
| chianti.tiv.lab.raleigh.ibm.com | Workshop2:1:1443560400 |                          | : Linux_BP_SpaceUsedPct_Critical([Disk_Used_Percent]>=95 AND FS_Type<>"nfs") ON chianti.tiv.lab.raleigh.ibm.com                 |
| chianti.tiv.lab.raleigh.ibm.com | Workshop2:1:1443560400 |                          | : Linux_BP_SpaceUsedPct_Critical([Disk_Used_Percent]>=95 AND FS_Type<>"nfs") ON chianti.tiv.lab.raleigh.ibm.com                 |
| chianti.tiv.lab.raleigh.ibm.com | Workshop2:1:1443560400 |                          | : Linux_BP_SpaceUsedPct_Critical([Disk_Used_Percent]>=95 AND FS_Type<>"nfs") ON chianti.tiv.lab.raleigh.ibm.com                 |
| chianti.tiv.lab.raleigh.ibm.com | Workshop2:1:1443560400 |                          | : Linux_BP_SpaceUsedPct_Critical([Disk_Used_Percent]>=95 AND FS_Type<>"nfs") ON chianti.tiv.lab.raleigh.ibm.com                 |

[Related events](#)

© Copyright IBM Corporation 2016, 2017

Figure 19-10. Viewing grouped events

In this example, the synthetic parent event that indicates a probable cause of the children events is seen at the top of the Event Viewer. The associated children events are indented underneath the parent event.

**Exercises:**  
**Finding seasonal events**  
**Using Tivoli Netcool Configuration Manager**

---

[Related events](#)

© Copyright IBM Corporation 2017

*Figure 19-11. Exercises: Finding seasonal events, and Using Tivoli Netcool Configuration Manager*

Complete the student exercises for this unit.

The completion of this unit is regarded as an optional part of the Tivoli Network Manager 4.2 course. However, the student exercise for this unit takes little time. It provides valuable experience for those individuals who have little knowledge of Netcool Operations Insight.

## Unit summary

After completing this unit, you should now be able to do the following tasks:

- Describe the steps in the related events workflow
- Explain how to conduct a related events analysis
- Explain how to examine the results of the analysis
- Explain how to deploy a group
- Explain how to view grouped events

*Figure 19-12. Unit summary*

# Unit 20. Optical transport and Radio Access Networks

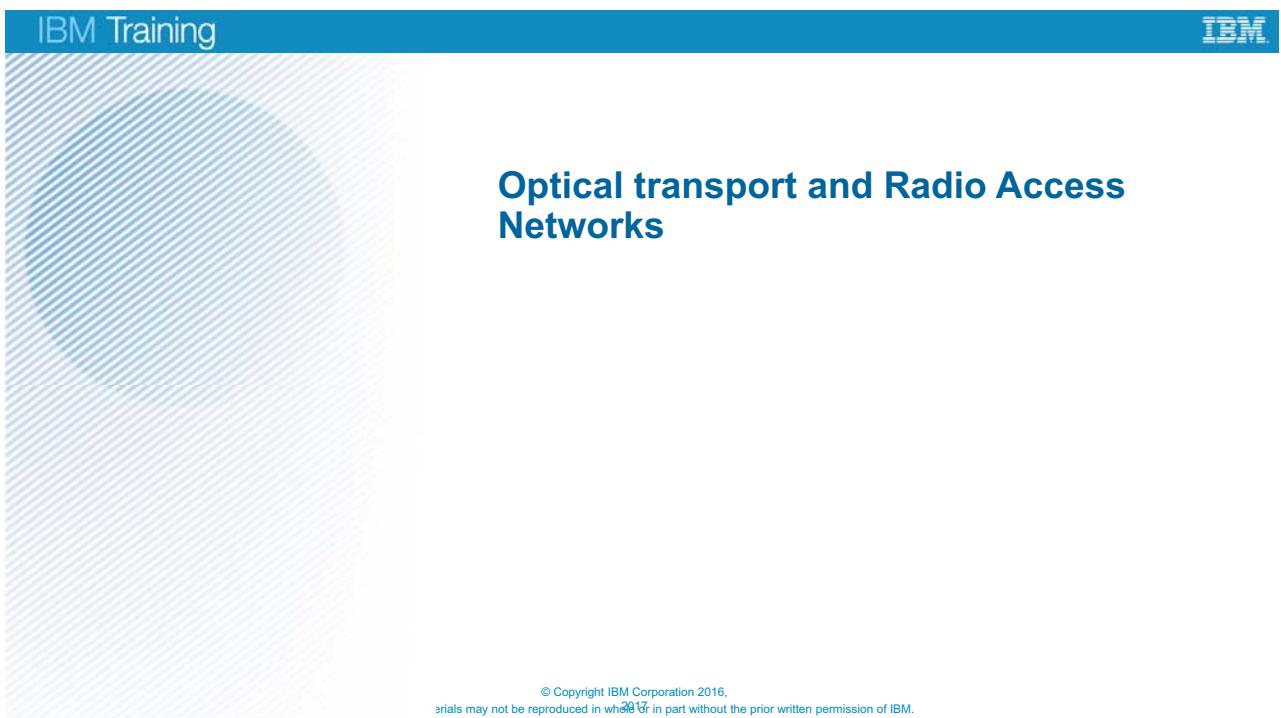


Figure 20-1. Optical transport and Radio Access Networks

## Estimated time

00:45

## Overview

This unit addresses the new functionality of IBM Tivoli Network Manager 4.1 for discovering optical and Radio Access Networks (RAN).

## How you will check your progress

- Complete lab exercises

## References

Configuring Java Collectors

<http://tinyurl.com/lwvjghk>

This unit introduces the capabilities of IBM Tivoli Network Manager 4.2 in discovering optical transport devices and devices that are part of Radio Access Networks (RAN). These capabilities are new to version 4.2.

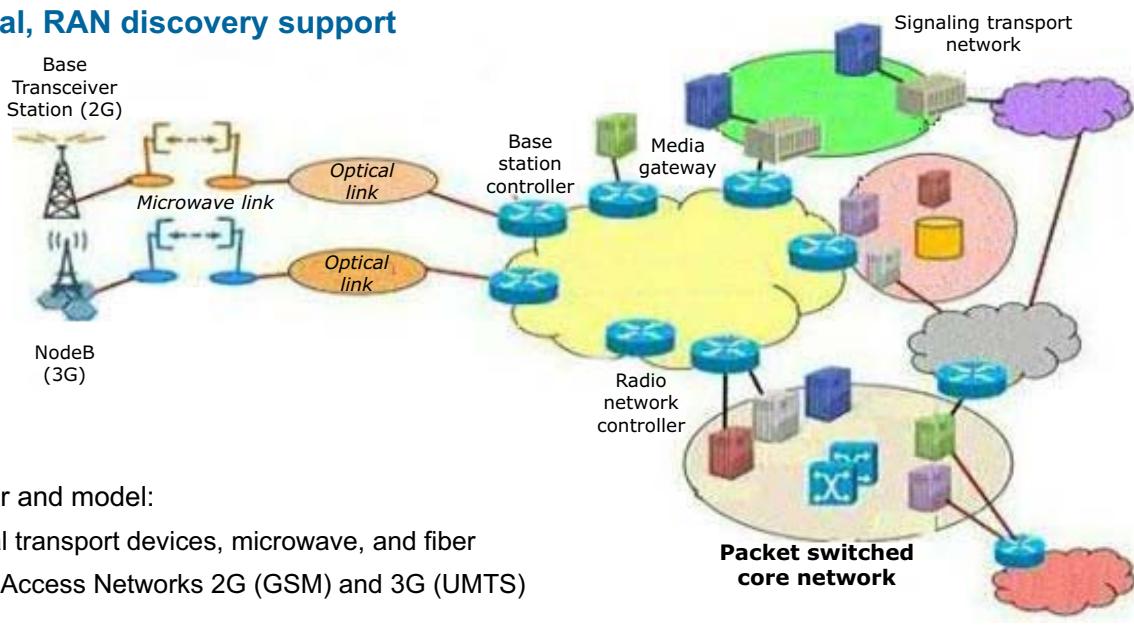
## Objectives

After you complete this unit, you should be able to do the following task:

- Describe the process for discovering optical or RAN networks

*Figure 20-2. Objectives*

## Optical, RAN discovery support



### Discover and model:

- Optical transport devices, microwave, and fiber
- Radio Access Networks 2G (GSM) and 3G (UMTS)

[Optical transport and Radio Access Networks](#)

© Copyright IBM Corporation 2016, 2017

Figure 20-3. Optical, RAN discovery support

Tivoli Network Manager uses data from element management systems (EMS) to discover and model:

- Optical transport devices, microwave, and fiber devices (with or without an IP address)
- Radio Access Network (RAN) devices that include:
  - Global System for Mobile (GSM) communications devices (also known as 2G devices)
  - Universal Mobile Telecommunications System (UMTS) devices (also known as 3G devices)

Tivoli Network Manager gathers optical and RAN data with discovery collectors. It then stores this data in the **Network Connectivity and Inventory Model** (NCIM) database. Enhancements to this database in Tivoli Network Manager Version 4.2 provide storing data even for devices that do not have IP addresses. This data is used to provide visualization of the network topology and detailed device information in the Device Structure browser.

## Components that support optical transport and RAN devices

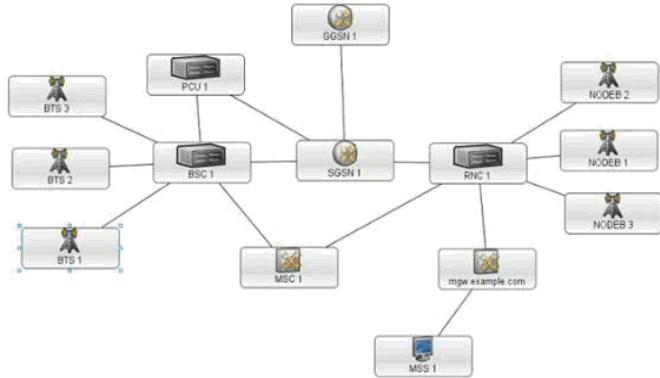
- New collectors
- Collector framework enhancements
- XML schema enhancements
- New collector agents
- Collector developer tools

Figure 20-4. Components that support optical transport and RAN devices

IBM Tivoli Network Manager 4.2 has several components that it uses to discover optical transport and RAN devices.

## Tivoli Network Manager vendor EMS support

- Tivoli Network Manager supports integration to element management systems (EMS) from multiple vendors:
  - Alcatel
  - Cisco
  - NSN
  - Tellabs
  - Huawei
- Enhanced visualization support
  - RAN-specific extensions (such as cell tower icons)
  - Bookmarks to help browse large data sets in the GUI
- The Java Collector framework and support library enables the development of Java, discovery collectors, and integration with CORBA NBI



Optical transport and Radio Access Networks

© Copyright IBM Corporation 2016, 2017

Figure 20-5. Tivoli Network Manager vendor EMS support

Tivoli Network Manager supports the following vendor element managers:

- Alcatel-Lucent 1353NM
- Alcatel-Lucent 1354RM
- Cisco CiscoWorks LMS
- NSN NetAct
- NSN DragonWave NetViewer
- Tellabs INM8000
- Huawei with CORBA TMF 814 (supports U2000 EMS)

In addition to new collectors, Tivoli Network Manager 4.2 features new Java support:

- The Java collector framework has a Java support library that can be used for the development of new Java discovery collectors.
- Collectors use Java Common Object Request Broker Architecture (CORBA) Northbound Interface (NBI) framework support to connect to a CORBA Northbound interface to provide two-way communication between the collector and the CORBA device.
- For more information on Java collectors, see <http://tinyurl.com/lwyighk>.

The **GetDeviceList()** field supports a protocol field to record the protocol that is used to discover a device. Valid values for this field include:

- Unknown: 0
- IPv4: 1
- NAT: 2

- IPv6: 3
- EMS key: 4

Many EMS systems represent entity containment with names rather than indexes. With enhancements to the following components in Tivoli Network Manager 4.2, they can use either **ENTITY-MIB** or **non-ENTITY-MIB** information to record containment relationship data:

- **GetEntities()** XML-RPC
- Perl and Java support libraries
- **CollectorInventory** agent

## Optical transport and RAN EMS integration

Devices without IP addresses can be discovered with data from their element management systems (EMS)

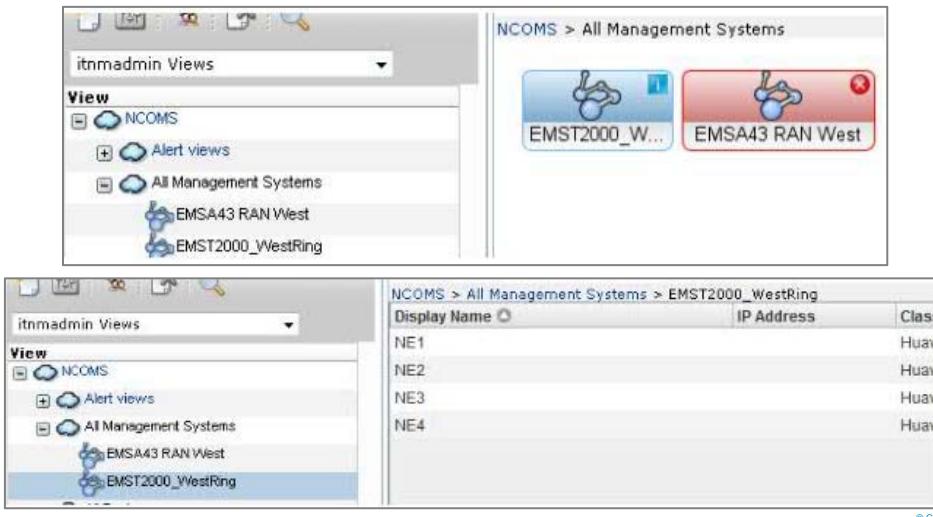
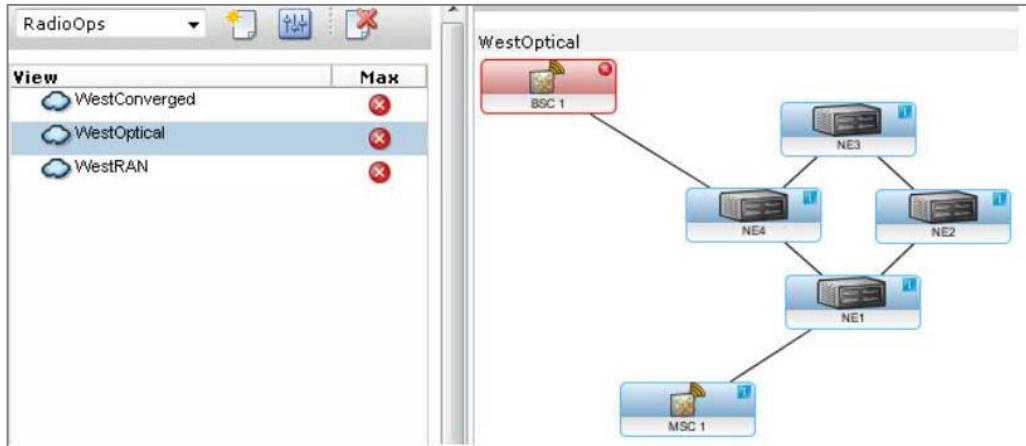


Figure 20-6. Optical transport and RAN EMS integration

Each EMS that is used as a data source for discovery is represented in the network views tree. Under the **All Management Systems** view, an icon represents each source EMS.

## Optical and RAN topology visualization

Tivoli Network Manager 4.2 shows Layer 1 devices and topology maps that are discovered through EMS



Optical transport and Radio Access Networks

© Copyright IBM Corporation 2016, 2017

Figure 20-7. Optical and RAN topology visualization

Only Layer 2 and Layer 3 network devices would be discovered in previous versions of Tivoli Network Manager. In version 4.2, Layer 1 devices can be instantiated in the network topology if they are discovered with a discovery collector.



## GSM, GPRS, UMTS, and RAN topology visualization

- Tivoli Network Manager 4.1.1 introduced the ability to visualize a 2G, 2.5G, or 3G network that is discovered with EMS information
- Tivoli Network Manager 4.2 also supports 4G LTE networks

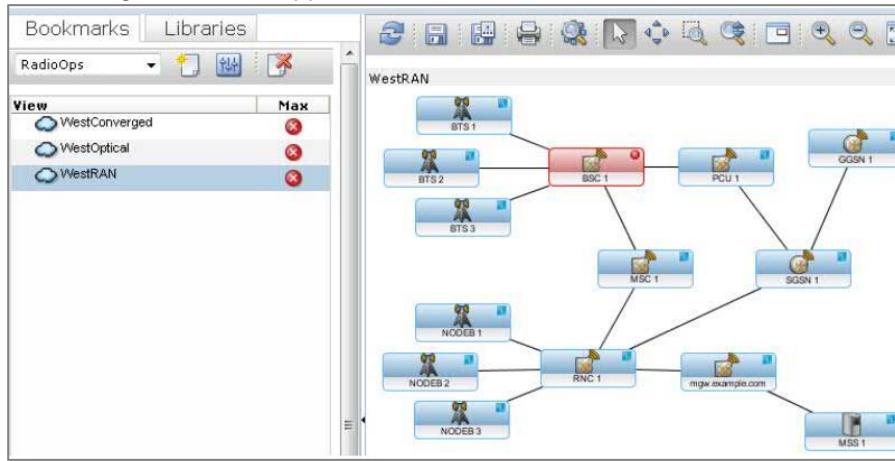


Figure 20-8. GSM, GPRS, UMTS, and RAN topology visualization

IBM Tivoli Network Manager 4.2 can discover 2G, 2.5G, 3G, and 4G LTE that are used for cellular networks.

## Multiple domain visualizations

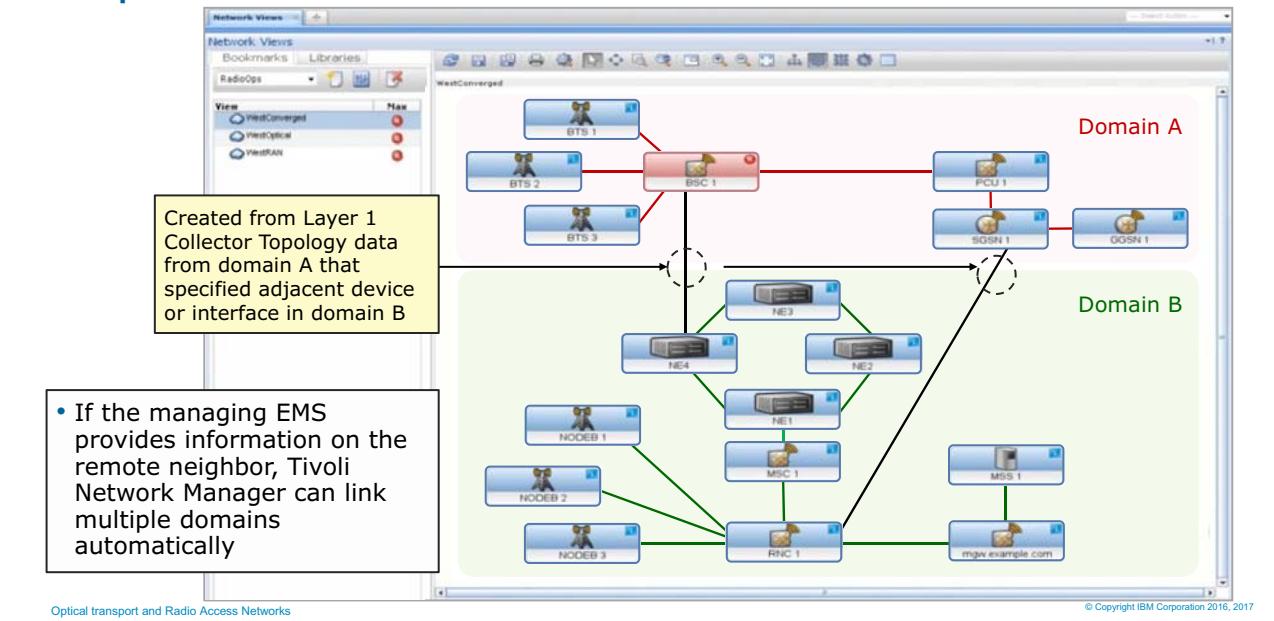


Figure 20-9. Multiple domain visualizations

Tivoli Network Manager 4.2 enables cross-domain network views that can show devices from separate discovery domains in a consolidated view.

When collectors discover optical and RAN network devices from separate element management systems, the discovered devices can be viewed in a consolidated network view. To display devices that are managed by separate EMS systems, at least one EMS must provide information about a device's remote neighbor.

## NCIM EMS integration support

- The NCIM database is extended to record the source from which an entity was discovered such as:
  - An element management system (EMS)
  - Manually added devices
  - Devices that are discovered with direct access
- Tivoli Network Manager also records the protocol that was used to discover the device
- Tivoli Network Manager and Tivoli Netcool Configuration Manager can synchronize information about EMS-managed devices to prevent the need for two discoveries

*Figure 20-10. NCIM EMS integration support*

The NCIM database has a field that records the source of discovery data. This field determines the source of discovery information for each device in your topology. For example, stacked switches can be discovered by talking to a particular element manager. Some devices are manually added to the topology. Normal finders and agents find other devices. Additionally, the type of protocol that is used to discover the device is included in the entity record.

```
// Standard agents.sourceInfo inserts
//
// The effect of the following inserts is as
// follows:
// - if a device appears in AssocAddress.returns and has m_HaveAccess=1,
// then add a discovery source of 'Agent',
// 'SNMP'.
// - if a device appears in Details.returns and
// has m_HaveAccess=0,
// then add a discovery source of 'Agent',
// 'Unknown'.
//
// - if a device appears in
// CollectorDetails.returns and has a
// m_ExtraInfo->m_CollectorInfo block, then add
// a discovery source
// of 'Collector', 'XML-RPC'.
```

```
insert into agents.sourceInfo
(
 m_Name,
 m_Source,
 m_DiscoveryProtocol,
 m_RequiredField,
 m_RequiredValue
)
values
(
 'AssocAddress',
 'Agent',
 'SNMP',
 ['m_HaveAccess'],
 '1'
);
```

## Optical, RAN discovery: Collector agents

- New CollectorLayer1 agent

`GetLayer1Connections(), GetConnections(Microwave)`

- New CollectorRAN agent

`GetRANData(), GetConnections(RAN)`

- Perl agent API extended to support XML-RPC calls

`GetXMLRPCData($host, $port, $method, $methodArgArray)`

`GetXMLRPCEntityData($ne, $method, $methodArgArray)`

Figure 20-11. Optical, RAN discovery: Collector agents

Several new components in IBM Tivoli Network Manager 4.2 enable discovery of optical transmission and RAN devices. Many EMS systems represent entity containment by using names rather than indexes. For instantiation of these items, several components now support a non-ENTITY-MIB form of data.

- The **GetEntities()** XML-RPC call now also supports a non-ENTITY-MIB form.
- Perl and Java support libraries now support both ENTITY-MIB and non-ENTITY-MIB forms.
- The **CollectorInventory** agent now supports both ENTITY-MIB and non-ENTITY-MIB forms.
- The **GetDeviceList()** function field now supports non-IP addresses in the address field.

## Starting a Java collector

- Go to the `$ITNMHOME/collectors/javaCollectors/bin` directory

- Run the command to start the collector

```
./collector.sh -jar [-Xmsminimum_memory-sizem] [-Xmxmaximum_memory-sizem] jar_file
-propsFile file_name -port port_number [-bg]
```

- For example, to start the Java CSV collector with the default .jar and property files, on port 8089, with minimum heap size of 512 M and maximum heap size of 1024 M:

```
cd $ITNMHOME/collectors/javaCollectors/bin ./collector.sh -Xms512m -Xmx1024m -jar
csv/csvcollector.jar -propsFile csv/csvcollector.properties -port 8089 -bg
```

Figure 20-12. Starting a Java collector

When you use a discovery collector, start the collector before you start the discovery process.

## Stopping a Java collector

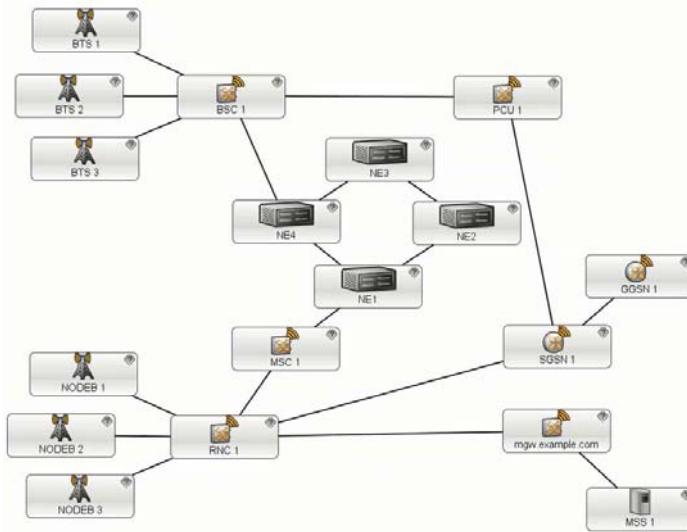
- Go to the `$ITNMHOME/collectors/javaCollectors/bin` directory
- Type the command to stop the collector that is running on a specific port:

```
shutdown_collector.sh -port port_number -host host_name
```

Figure 20-13. Stopping a Java collector

## Optical and RAN device icons

New topology types and icons in network and hop views



Optical transport and Radio Access Networks

© Copyright IBM Corporation 2016, 2017

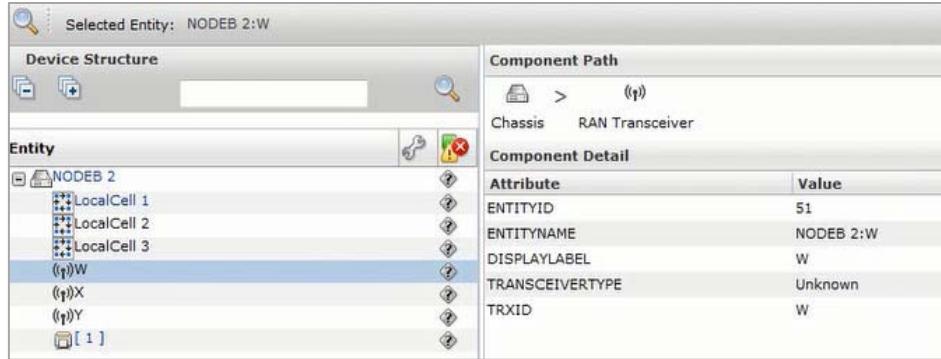
Figure 20-14. Optical and RAN device icons

Tivoli Network Manager has new icons to represent optical and RAN devices.

## Optical and RAN devices in Device Structure browser

### Structure browser view

- Optical connection termination point (CTP) data and physical termination point (PTP) data are shown in a tree structure under the cards and interfaces in which they are contained
- RAN data is seen in the Device Structure browser as seen in this graphic



Optical transport and Radio Access Networks

© Copyright IBM Corporation 2016, 2017

Figure 20-15. Optical and RAN devices in Device Structure browser

The preceding graphic shows an example Structure browser view. It shows transceivers and cells that are related to a 3G **NodeB**.

## Managed systems view

- New All Management Systems view



- Synchronization with Tivoli Netcool Configuration Manager

| Display Name | IP Address | Class Name        | Class Type    | Managed State | Maximum Severity                    |
|--------------|------------|-------------------|---------------|---------------|-------------------------------------|
| NE1          |            | HuaweiMultiplexer | NetworkDevice | Managed       | <span style="color: blue;">i</span> |
| NE2          |            | HuaweiMultiplexer | NetworkDevice | Managed       | <span style="color: blue;">i</span> |
| NE3          |            | HuaweiMultiplexer | NetworkDevice | Managed       | <span style="color: blue;">i</span> |
| NE4          |            | HuaweiMultiplexer | NetworkDevice | Managed       | <span style="color: blue;">i</span> |

Figure 20-16. Managed systems view

An **All Management Systems** view shows all element management systems from which data was gathered. Double-clicking the element management system drills down into a view that contains information on the devices that are managed by that EMS.

## Unit summary

- Describe the process for discovering optical or RAN networks

*Figure 20-17. Unit summary*

This unit has no student exercises.

# Appendix A. Databases and passwords

IBM Training

IBM

## Databases and passwords

© Copyright IBM Corporation 2016  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

### Estimated time

00:30

### Overview

This unit describes DB2 tools and commands that can be used to set values to optimize database performance for Netcool.

## Key DB2 commands (1 of 2)

- Source environment for database instance owner **db2inst1**

```
~db2inst1/sqlllib/db2profile
```

- Connect to database

```
db2 connect to databaseName user username using password
```

- List the tables of a schema

```
db2 "select char(tabschema,10) tabschema, char(tabname,33) tablename, stats_time
from syscat.tables where tabschema='NCIM' order by 1,2,3"
```

Figure A-1. Key DB2 commands (1 of 2)

## Key DB2 commands (2 of 2)

- Stop all DB2 processes

```
db2 terminate ; db2 force applications all ; db2stop force ;
```

- Clean all interprocess communications

```
ipclean -a
```

- Start database **ITNM**

```
db2start
```

```
db2 restart db ITNM ; db2 activate db ITNM ;
```

Figure A-2. Key DB2 commands (2 of 2)

Many of the DB2 executable files are found in the `$NCHOME/platform/platform-designation/db2/adm` directory. You might need to use `chmod +x` on some of these files to run the files from this directory. For ease of administration, add this directory to your `PATH` environment variable.

## Re-create a topology database instance

- As the database instance owner, run `db2move` on the source database server in a new directory to export the database (such as **ITNM**):  
`db2move ITNM export -aw tar cvf custname_dbname.tar .`
- Create a database (such as **CUST**) on the target database server `create_db2_database.sh`:  
`create_db2_database.sh CUST username [-force ]`
- Run `db2move` on the lab database server to import into the new database  
`tar xvf custname_dbname.tar`  
`db2move CUST import`

Figure A-3. Re-create a topology database instance

## General troubleshooting commands

- Gather debug for support (creates file `db2support.zip`)
  - Add to the **mustgather** utility for any database support issue

```
db2support . -d databaseName -c -s
```

- View database names

```
db2 list db directory
```

- Details of a table (such as the **entityData** table)

```
db2look -d databaseName -e -z NCIM -t ENTITYDATA -i username -w password
```

- Show configuration settings

```
db2 get dbm cfg
```

```
db2 get db cfg for databaseName
```

```
db2set -all
```

*Figure A-4. General troubleshooting commands*

When you open a support ticket, you are often given instructions on how to use the **mustgather** utility. This utility gathers information that IBM technical support requires to assist you.

## Diagnostic commands (1 of 4)

- Identify long-running commands

- List the last 10 queries by average execution time:

```
db2 "select NUM_EXECUTIONS, STMT_TEXT from sysibmadm.TOP_DYNAMIC_SQL order by
AVERAGE_EXECUTION_TIME_S desc fetch first 10 rows only" | tr -s " "
```

- List the last 10 queries by number of executions:

```
db2 "select NUM_EXECUTIONS, STMT_TEXT from sysibmadm.TOP_DYNAMIC_SQL order by NUM_EXECUTIONS
desc fetch first 10 rows only" | tr -s " "
```

- Command to know the queries that had more than X number of executions and by **SORTS\_PER\_EXECUTION** to know which queries might indicate missing or poor indexing:

```
db2 "select NUM_EXECUTIONS, SNAPSHOT_TIMESTAMP,
AVERAGE_EXECUTION_TIME_S,SORTS_PER_EXECUTION,STMT_TEXT from sysibmadm.TOP_DYNAMIC_SQL
where NUM_EXECUTIONS > 100 order by SORTS_PER_EXECUTION desc" | tr -s " "
```

*Figure A-5. Diagnostic commands (1 of 4)*

## Diagnostic commands (2 of 4)

- Create DB2 **explain** tables to know more information about a specific query

```
cd ~/sqllic/misc
db2 -tvf EXPLAIN.DDL

db2advis -d databaseName -n <schema> -s "SELECT DISTINCT tunnel.entityId TUNNELID,
COALESCE(tunnel.displayLabel,tunnel.entityName) TUNNELNAME, dm.domainMgrId
DOMAINMGRID FROM ncim.entityData tunnel INNER JOIN ncim.domainMembers dm ON
dm.entityId = tunnel.entityId INNER JOIN ncim.networkPipe outerPipe ON
outerPipe.entityId = tunnel.entityId INNER JOIN ncim.connects outerPipeConnects
ON outerPipeConnects.connectionId = outerPipe.connectionId INNER JOIN
ncim.mplsTETunnel headend ON headend.entityId = tunnel.entityId LEFT OUTER JOIN
ncpgui.pathView pv ON pv.pathId = tunnel.entityId WHERE tunnel.entityType = 36
AND pv.viewId IS NULL "
```

Figure A-6. Diagnostic commands (2 of 4)

## Diagnostic commands (3 of 4)

- Command to get diagnostic data directory:

```
db2 get dbm cfg | grep -i diag
```

- This command returns a directory such as:

```
/opt/IBM/tivoli/netcool/platform/linux2x86/db2client/sql1ib/db2dump/
```

- In this directory, you can find:

- Diagnostic information like memory usage and problem queries in `instance_name.nfy`
  - Diagnostics information like deadlocks is found in `db2diag.log`

Figure A-7. Diagnostic commands (3 of 4)

## Diagnostic commands (4 of 4)

- Run this command to know more deadlock details:

```
db2evmon -path
/opt/IBM/tivoli/netcool/platform/linux2x86/db2client/ncim/NODE0000/SQL00001/MEMBER
0000/db2event/db2detaildeadlock > db2detail_dead_log.txt
```

- Identify all long-running locks and deadlocks:

```
db2 "select l.agent_id, substr(l.appl_name, 1, 15) as appName,
substr(s.stmt_text, 1, 70) as stmt_text, l.lock_object_type, l.tabname from
sysibmadm.locks_held as l, sysibmadm.long_running_sql as s where l.agent_id =
s.agent_id and s.stmt_text is not null"
```

Figure A-8. Diagnostic commands (4 of 4)

## Performance Analyst Tool

- Useful Tivoli tool to analyze database snapshots to identify worst query offenders, and database tuning issues
- Home page and download
  - <http://ibm.biz/BdxDbg>
- Comprehensive paper on using this tool
  - <http://tinyurl.com/ntnqmt4>

Figure A-9. Performance Analyst tool

## Analyze the snapshot.txt file

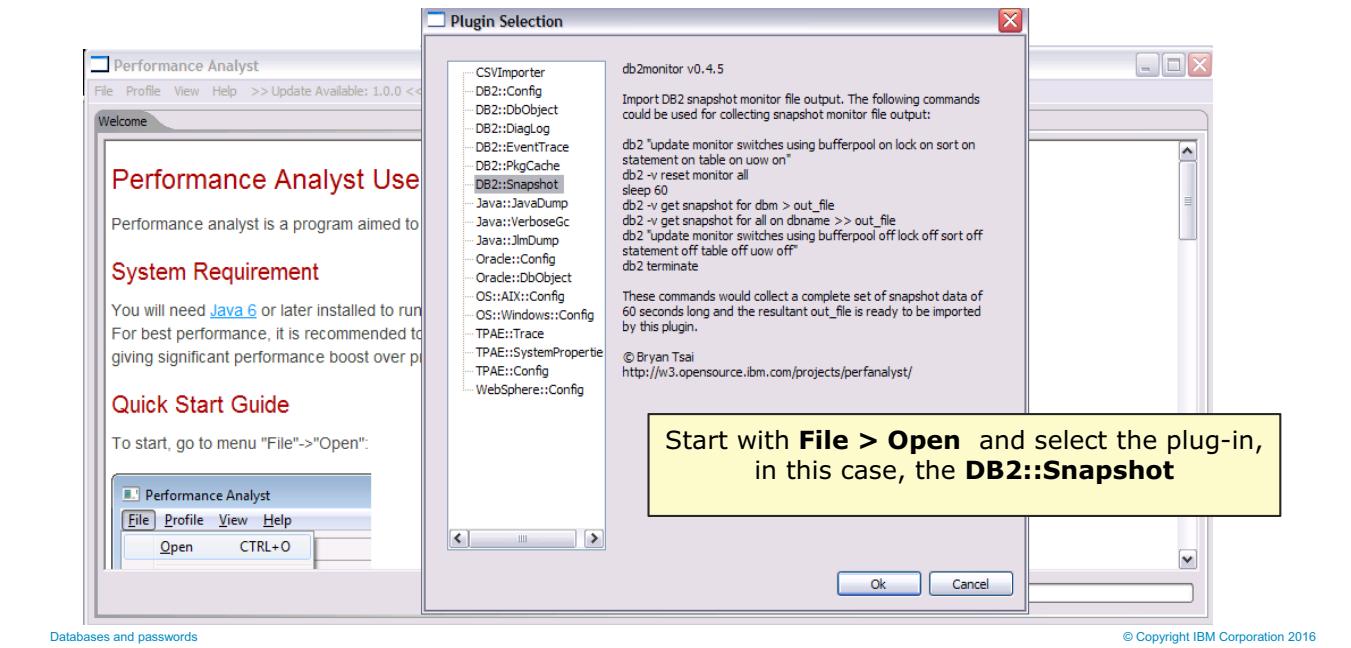
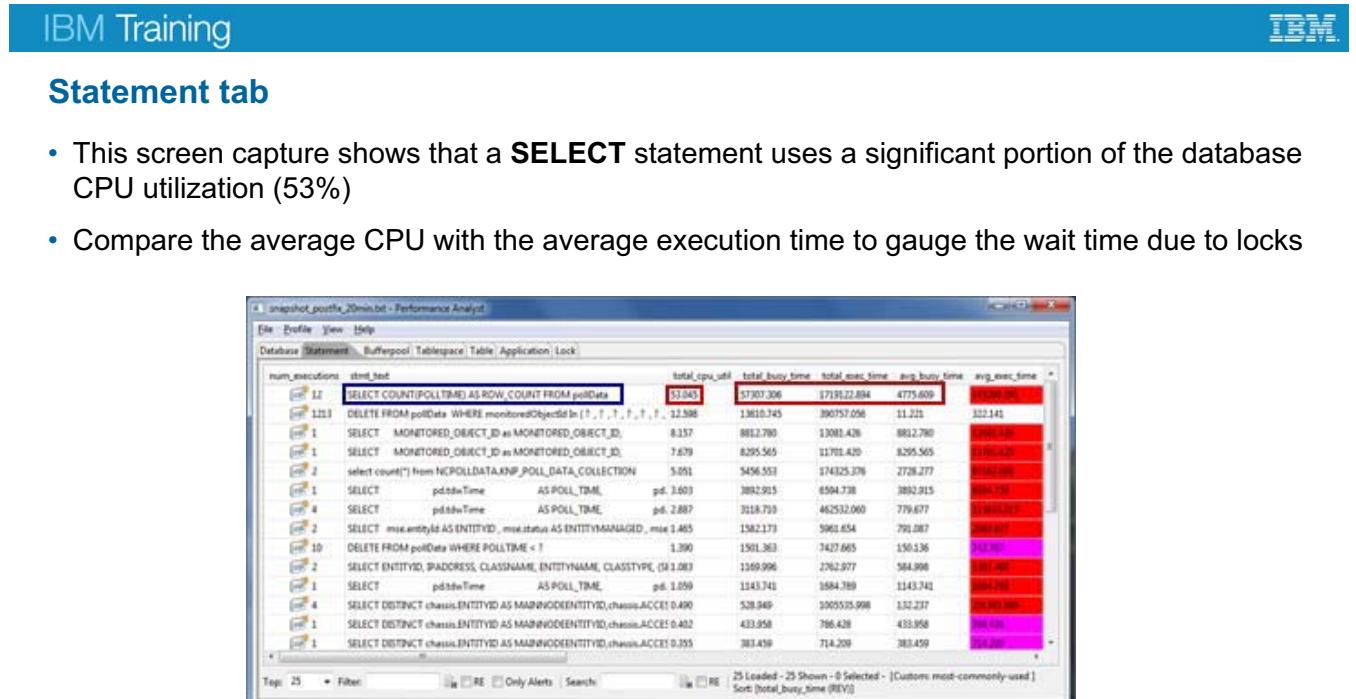


Figure A-10. Analyze the snapshot.txt file



*Figure A-11. Statement tab*

## Performance tuning with autoconfigure

- Ensure that the stats are up-to-date first:

```
db2 "reorgchk current statistics on table table_name" >reorgchk.out
```

- To see recommended values, use the following command:

```
db2 autoconfigure using mem_percent 80 apply none
```

| Current and Recommended Values for Database Manager Configuration |                   |                |                   |
|-------------------------------------------------------------------|-------------------|----------------|-------------------|
| Description                                                       | Parameter         | Current Value  | Recommended Value |
| Application support layer heap size (4KB)                         | (ASLHEAPSZ)       | 15             | 15                |
| No. of int. communication buffers(4KB)                            | (FCM_NUM_BUFFERS) | AUTOMATIC      | AUTOMATIC         |
| Enable intra-partition parallelism                                | (INTRA_PARALLEL)  | NO             | NO                |
| Maximum query degree of parallelism                               | (MAX_QUERYDEGREE) | ANY            | 1                 |
| Agent pool size                                                   | (NUM_POOLAGENTS)  | AUTOMATIC(100) | AUTOMATIC(100)    |
| Initial number of agents in pool                                  | (NUM_INITAGENTS)  | 0              | 0                 |
| Max requester I/O block size (bytes)                              | (RQRIOBLK)        | 32767          | 32767             |
| Sort heap threshold (4KB)                                         | (SHEAPTHRES)      | 0              | 0                 |

| Current and Recommended Values for Database Configuration |                   |               |                   |
|-----------------------------------------------------------|-------------------|---------------|-------------------|
| Description                                               | Parameter         | Current Value | Recommended Value |
| Default application heap (4KB)                            | (APPLHEAPSZ)      | 256           | 256               |
| Catalog cache size (4KB)                                  | (CATALOGCACHE_SZ) | 300           | 697               |

Databases and passwords

© Copyright IBM Corporation 2016

Figure A-12. Performance tuning with autoconfigure

The **apply none** syntax causes the command to return configuration information without setting any values. If you remove **apply none**, values are set.

## Transaction log size (1 of 2)

- If the transaction log size is insufficient, you might see errors during periods of many uncommitted row changes such as:

The transaction log for the database is full. SQLSTATE=57011

- By default, DB2 sets the logging to be circular (**logarchmeth1=OFF, logarchmeth2=OFF**)

- You can increase log file size or the number of secondary log files

```
db2 update db cfg for databaseName using LOGBUFSZ 2150
db2 update db cfg for databaseName using LOGFILSIZ 5000
db2 update db cfg for databaseName using LOGPRIMARY 10
db2 update db cfg for databaseName using LOGSECOND 20
```

Figure A-13. Transaction log size (1 of 2)

For more information, see:

[https://www.ibm.com/support/knowledgecenter/en/SSVJJU\\_6.4.0/com.ibm.IBMDS.doc\\_6.4/c\\_tg\\_tuning\\_db2\\_transact\\_log\\_size.html](https://www.ibm.com/support/knowledgecenter/en/SSVJJU_6.4.0/com.ibm.IBMDS.doc_6.4/c_tg_tuning_db2_transact_log_size.html)

## Transaction log size (2 of 2)

- If the transaction log files are switching too frequently, performance can be significantly affected
  - Try to get only a few switches per hour
- Monitor the current active log ID over time:

```
db2 "Select MEMBER, CUR_COMMIT_DISK_LOG_READS, CURRENT_ACTIVE_LOG from
table(mon_get_transaction_log(-1)) as t order by member asc"
```

Figure A-14. Transaction log size (2 of 2)

## Maximum number of connections

- By default, Tivoli Network Manager sets the number of application connections that are permitted to 100
  - This value is sufficient for up to five domains
- Increasing the number of domains beyond 100 can generate database connection fail errors (**SQLSTATE=57030**)
- Set the number of connections to **AUTOMATIC**
  - If it is a shared database, you can set the value to 200 for the NCIM database

```
db2 update db cfg for NCIM using MAXAPPLS AUTOMATIC
```
- Restart the database instance

Figure A-15. Maximum number of connections

## DB2 database tuning (1 of 2)

- The following DB2 database configuration settings can result in optimized performance:

```
db2 update db cfg for databaseName using LOGBUFSZ 2150
db2 update db cfg for databaseName using LOGFILSIZ 5000
db2 update db cfg for databaseName using LOGPRIMARY 10
db2 update db cfg for databaseName using LOGSECOND 20
db2 update db cfg for databaseName using LOCKTIMEOUT 300
```

Figure A-16. DB2 database tuning (1 of 2)

## DB2 database tuning (2 of 2)

```
db2 update db cfg for databaseName using NUM_IOCLEANERS AUTOMATIC (default)
db2 update db cfg for databaseName using MAXFILOP 61440 (UNIX / Linux 64 bit)
db2 update db cfg for databaseName using SOFTMAX 1000
db2 update db cfg for databaseName using STMTHEAP 20000
db2 update db cfg for databaseName using STMT_CONC LITERALS
```

Figure A-17. DB2 database tuning (2 of 2)

The following default settings for DB2 are already set for optimal performance:

```
db2 update db cfg for databaseName using NUM_IOSERVERS AUTOMATIC
db2 update db cfg for databaseName using DFT_QUERYOPT 5
db2 update db cfg for databaseName using LOCKLIST AUTOMATIC
db2 update db cfg for databaseName using AUTO_REVAL DEFERRED
db2 update db cfg for databaseName using DEC_TO_CHAR_FMT NEW
```

## Automatic tuning

- Unless constrained for resources, use the auto tuning

```
db2 update db cfg for databaseName using DATABASE_MEMORY AUTOMATIC
db2 update db cfg for databaseName using PCKCACHESZ AUTOMATIC
db2 update db cfg for databaseName using DBHEAP AUTOMATIC
db2 update db cfg for databaseName using STAT_HEAP_SZ AUTOMATIC
```

Figure A-18. Automatic tuning

## Registry settings

- Consider these settings to avoid excessive locking, especially when storing poller data

```
db2set DB2_SKIPINSERTED=ON
```

```
db2set DB2_SKIPDELETED=ON
```

- For optimal performance

```
db2set DB2_USE_ALTERNATE_PAGE_CLEANING=ON
```

Figure A-19. Registry settings

## Database manager settings

- For optimal performance of the database manager, use the following settings:

```
db2 update dbm cfg using AGENT_STACK_SZ 1024 (UNIX / Linux default)
```

```
db2 update dbm cfg using AGENT_STACK_SZ 1000 (Windows)
```

```
db2 update dbm cfg using QRIOBLK 65535
```

```
db2 update dbm cfg using MON_HEAP_SZ AUTOMATIC (default)
```

Figure A-20. Database manager settings

## Changing passwords

© Copyright IBM Corporation 2016  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

*Figure A-21. Changing passwords*

## Changing DB2 database passwords

- DB2 uses operating system access accounts that often have security standards that mandate periodic password changes
  - Password changes can result in database connection failures when the password fails
  - The read-write database account name in use can be found in `$NCHOME/etc/precision/DbLogins.domainName.cfg`
- When the password is changed on the server, the Tivoli Network Manager administrator must reset the changed password in:
  - Tivoli Network Manager
  - Tivoli Common Reporting (verify whether the operation is using Tivoli Data Warehouse or local database for `ncpolldata` database)

Figure A-22. Changing DB2 database passwords

## Changing passwords in configuration files

- Encrypt the new password using **ncp\_crypt**

```
ncp_crypt -password netcool
'@44:O2aJesKCAX6Af0aOKeTBMRWk2ru8soEOe9PFEv6smwc=@'
```

- Copy and paste the encrypted string (without the single quote marks) into the following files for each domain that uses this database account name:
  - \$NCHOME/etc/precision/DbLogins.domainName.cfg**
  - \$NCHOME/etc/precision/MibDbLogin.domainName.cfg**
- Restart the ncp processes

```
itnm_stop ncp
itnm_start ncp
```

Figure A-23. Changing passwords in configuration files

## Tivoli Network Manager GUI database access

- Change the password using the GUI:
  - Select from the menu **Administration > Network > Database Access Configuration**
  - Set the new password in both panels and click **Save** for each panel
  - Tivoli Data Warehouse is not affected by operating system password changes
  - Change the password in Tivoli Data Warehouse for the **ncpolldata** database only if it changes
- Where you can use **ncp\_crypt** encrypted passwords:
  - The **ncp\_crypt** utility can encrypt and decrypt passwords only for **DbLogins** and **MibDbLogin**
  - The encrypted password cannot be copied to the two GUI files

Figure A-24. Tivoli Network Manager GUI database access

## Tivoli Common Reporting BIRT and Cognos reports

- If you are using local **ncpolldata** database:

- Use the **configTCR.sh** script to reset the connection details to the local data source

```
$NCHOME/precision/products/tnm/bin/configTCR.sh
```

- Run to modify all the data sources

```
./configTCR -p tipadmin_pw -d new_db_pw
```

- If using Tivoli Data Warehouse, you must use the Tivoli Common Reporting **trcmd** utility to set the password for each datasource separately

Figure A-25. Tivoli Common Reporting BIRT and Cognos reports

## Tivoli Common Reporting BIRT reports

- Use the Tivoli Common Reporting CLI script

```
/opt/IBM/tivoli/tipv2Components/TCRComponent/bin/trcmd.sh
```

- Run the command for each datasource

- For example, if the new password is **netcool**:

```
./trcmd.sh -modify -dataSources -reports -user tipadmin -password tipadmin_pw
-dataSource name=NCIM -setDatasource odaPassword=netcool
```

```
./trcmd.sh -modify -dataSources -reports -user tipadmin -password tipadmin_pw
-dataSource name=PARAMETERS -setDatasource odaPassword=netcool
```

- If Tivoli Data Warehouse database password changed:

```
./trcmd.sh -modify -dataSources -reports -user tipadmin -password tipadmin_pw
-dataSource name=NCPOLLDATA -setDatasource odaPassword=netcool
```

Figure A-26. Tivoli Common Reporting BIRT reports

## Tivoli Common Reporting Cognos with Tivoli Data Warehouse

If you cannot use the `configTCR.sh` script because you are using Tivoli Data Warehouse, you have two options to change the password:

- Use the Cognos GUI to change the password on the **Signon** page for each datasource.
- Use the **trcmd.sh** utility for each datasource
  - The database details must match the Tivoli Data Warehouse database for NCPOLldata and PARAMETERS
  - For example, if the database account is **db2inst** and new password **netcool**:

```
./trcmd.sh -dataSource -add NCIM -dbType DB2 -dbName NCIM -user tipadmin
-pw tipadmin_password
-openSessionSql "SET CURRENT SCHEMA = NCIM" -dbLogin db2inst -dbPassword
netcool -force
```

*Figure A-27. Tivoli Common Reporting Cognos with Tivoli Data Warehouse*

Remember that you need to change the password for each data source.

```
./trcmd.sh -dataSource -add NCIM -dbType DB2 -dbName NCIM -user tipadmin
-pw tipadmin_password -openSessionSql "SET CURRENT SCHEMA = NCIM" -dbLogin
db2inst -dbPassword netcool -force

./trcmd.sh -dataSource -add NCMONITOR -dbType DB2 -dbName NCIM -user tipadmin
-pw tipadmin_password
 -openSessionSql "SET CURRENT SCHEMA = NCMONITOR"
 -dbLogin db2inst -dbPassword netcool -force

./trcmd.sh -dataSource -add NCPGUI -dbType DB2 -dbName NCIM -user tipadmin
-pw tipadmin_password
 -openSessionSql "SET CURRENT SCHEMA = NCPGUI"
 -dbLogin db2inst -dbPassword netcool -force

./trcmd.sh -dataSource -add IBM_TRAM -dbType DB2 -dbName NCIM -user tipadmin
-pw tipadmin_password
 -openSessionSql "SET CURRENT SCHEMA = IBM_TRAM"
 -dbLogin db2inst -dbPassword netcool -force

./trcmd.sh -dataSource -add NCPOLldata -dbType DB2 -dbName NCIM -user tipadmin
-pw tipadmin_password
 -openSessionSql "SET CURRENT SCHEMA = NCPOLldata"
```

```
-dbLogin db2inst -dbPassword netcool -force

. ./trcmd.sh -dataSource -add PARAMETERS -dbType DB2 -dbName NCIM -user tipadmin
-password tipadmin_password

-openSessionSql "SET CURRENT SCHEMA = NCPOLLDATA"
-dbLogin db2inst -dbPassword netcool -force
```

## Use Cognos Administration to change Signon password

1. Select  
**Reporting >  
 Common Reporting**

2. Click **Launch >  
 Administration**

3. Click  
**Configuration**

The figure consists of two screenshots of the IBM Cognos Administration interface. The top screenshot shows the 'Work with reports' screen with the 'Administration' option highlighted in the 'Launch' menu. The bottom screenshot shows the 'Administration' screen with the 'Configuration' tab selected. Both screenshots include step numbers and descriptive text from the accompanying text blocks.

*Figure A-28. Use Cognos administration to change Signon password*

## Cognos Signon page (1 of 3)

4. Repeat the following steps for each data source:

**NCIM,  
NCPOLLDATA,  
PARAMETERS**

- Click the data source name
- Check the box next to the data source name, click **More**, click **Set Properties**

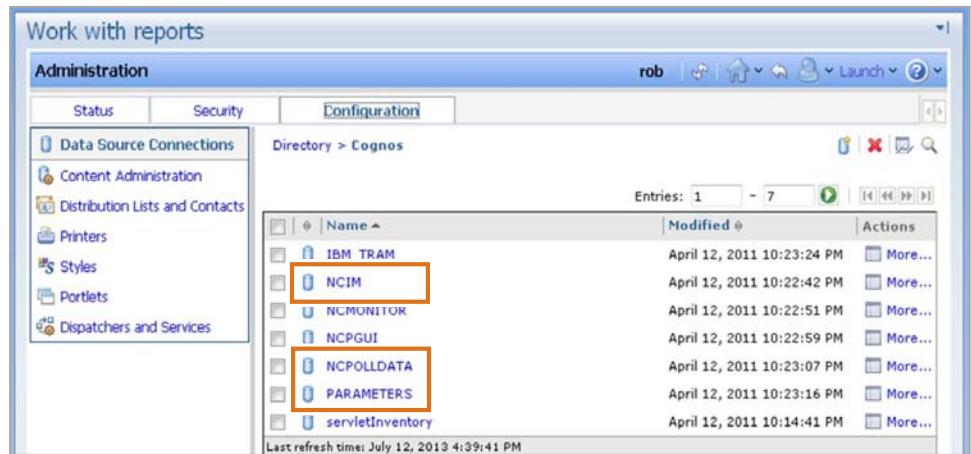


Figure A-29. Cognos Signon page (1 of 3)

## Cognos Signon page (2 of 3)

Work with reports

Administration

Status Security Configuration

Data Source Connections Content Administration Distribution Lists and Contacts Printers Styles Portlets Dispatchers and Services

Directory > Cognos > NCIM

Entries: 1 - 1 | More... | Actions

| Name | Modified                   | Actions                          |
|------|----------------------------|----------------------------------|
| NCIM | April 12, 2011 10:22:42 PM | <input type="checkbox"/> More... |

Last refresh time: July 12, 2013 4:41:55 PM

• Click the data source name

Status Security Configuration

Data Source Connections Content Administration Distribution Lists and Contacts Printers Styles Portlets Dispatchers and Services

Directory > Cognos > NCIM > NCIM

Entries: 1 - 1 | More... | Actions

| Name | Modified                  | Actions                                     |
|------|---------------------------|---------------------------------------------|
| ncim | April 2, 2012 10:43:58 AM | <input checked="" type="checkbox"/> More... |

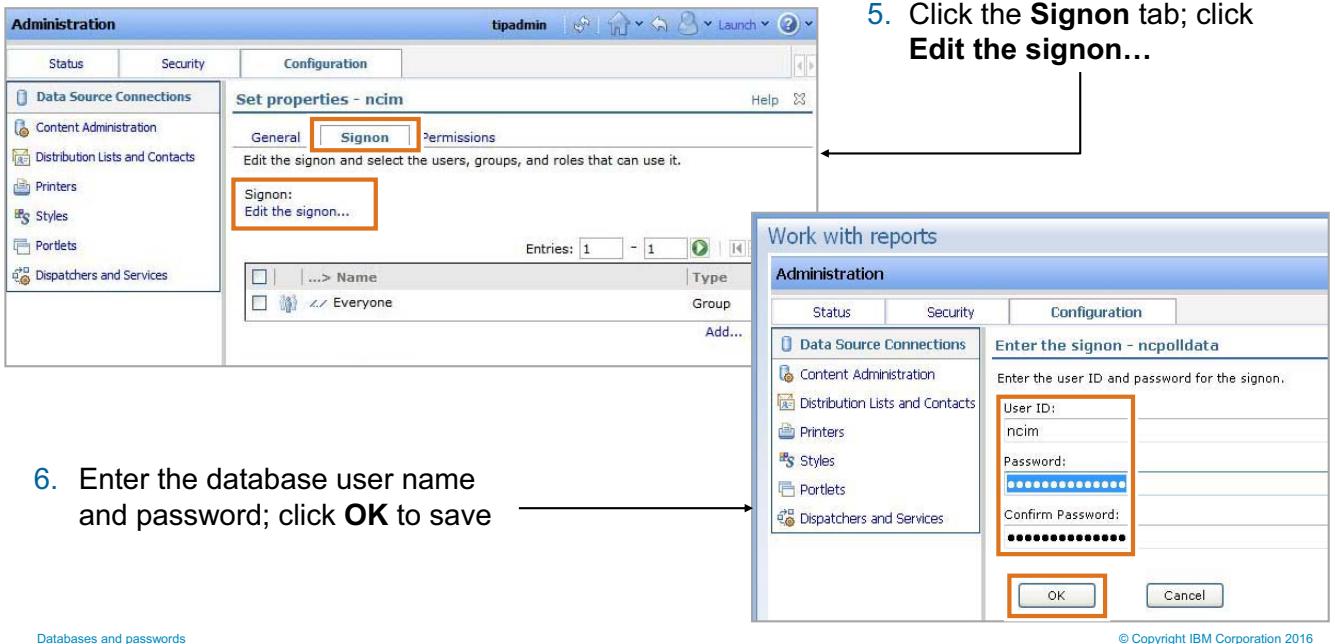
Last refresh time: July 12, 2013 4:48:36 PM

Databases and passwords © Copyright IBM Corporation 2016

- Check the box next to the data source name, click **More**, click **Set Properties**

Figure A-30. Cognos Signon page (2 of 3)

### Cognos Signon page (3 of 3)



## Helpful IBM Knowledge Center links

- Change database password for the Tivoli Network Manager core
  - <http://tinyurl.com/ybl2nkzg>
- Change database password for GUI
  - <http://tinyurl.com/qcm8rk7>
- Change database password for Cognos reports
  - <http://tinyurl.com/l3q6ha7>
- Change database password for BIRT reports
  - <http://tinyurl.com/o6ps7u2>
- DB2 10.1 documentation in IBM Knowledge Center
  - <http://tinyurl.com/l5jv8tb>

---

Databases and passwords

© Copyright IBM Corporation 2016

Figure A-32. Helpful IBM Knowledge Center links

## Useful DB2 references

- DB2 10.1 documentation in IBM Knowledge Center
  - <http://tinyurl.com/l5jv8tb>
- DB2 best practices Tuning and Monitoring database
  - <http://tinyurl.com/lidxlarw>
- Top 10 Db2 Performance tips
  - <http://tinyurl.com/lcpmnun>
- Review of 10 key DB02 Redbooks:
  - <http://tinyurl.com/lmqcbvy>

# Appendix B. Configuring DB2 high availability

IBM Training



## Configuring DB2 high availability

© Copyright IBM Corporation 2016  
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

### Estimated time

00:45

### Overview

This appendix explains some elements of setting up high availability database replication (HADR) for DB2.

## DB2 instances

- A DB2 instance essentially acts as an independent database server, accessible via a specific host and port combination
  - Each DB2 installation can have multiple instances
  - Each instance can host multiple named databases
  - An instance is required to host the NCIM database
- There is a one-to-one relationship between an instance and an instance owner
  - The `$DB2INSTANCE` variable identifies which instance is accessed via the command line
  - Log on as the instance owner, or source that owner's environment to access an instance

Figure B-1. DB2 instances

## DB2 instance configuration (1 of 2)

When setting up two DB2 instances to work as a HADR pair for use with Tivoli Network Manager:

- The same user name and password must be able to access both instances
  - The Tivoli Network Manager configuration accepts a single user name and password
  - There is already a requirement to have the same encryption key on the primary and backup servers in `$NCHOME/etc/security/keys/`
  - DB2 users are UNIX users, and so are subject to the same controls, such as password expiry
- Instance configuration (with `dbset`) is independent of database configuration (with `db2 update db cfg for dbName`)

Figure B-2. DB2 instance configuration (1 of 2)

## DB2 instance configuration (2 of 2)

- On the DB2 server, verify that the server is configured for TCP/IP access

- To display the configured communication protocol, use:

```
db2set -all DB2COMM
```

- If this command returns no results, set it with:

```
db2set DB2COMM=tcpip
```

- Restart the DB manager if `db2start` displays the error:

SQL5043N Support for one or more communications protocols failed to start successfully. However, core database manager functionality started successfully

- Run the following command, and supply the port number that clients use to access the server:

```
db2 update dbm cfg using svcename portNumber
```

- After restarting, `db2start` should no longer display the error

Figure B-3. DB2 instance configuration (2 of 2)

## Creating the primary DB2 database

1. A DB2 instance must be available
  - Creating DB2 users: <http://tinyurl.com/k99uxnv>
  - Creating a DB2 instance: <http://tinyurl.com/k8xkvya>
2. Create the NCIM database:
  - Use `create_db2_database.sh` on the database server
3. Populate the NCIM database:
  - Use `populate_db2_database.sh` on the database server or `create_all_schemas.pl` from the Tivoli Network Manager server

Figure B-4. Creating the primary DB2 database

## Key DB2 concepts: Services

- A “service” refers to a configured port

- <http://tinyurl.com/mm7x7xx>

- View configured services with:

```
db2 get dbm cfg | grep SVC
```

- Add a service with:

```
db2 update dbm cfg using svcename <serviceName>
```

- This command refers to the port number that is used to access the server from clients
  - The *serviceName* can be a port number, or it can be a name that is listed in **/etc/services**
  - Although listing services in **/etc/services** is optional, it is advised in the DB2 docs
  - It should be consistently listed on both the server and the client

Figure B-5. Key DB2 concepts: Services

## Troubleshooting: Problems accessing DB2 (1 of 3)

- Tivoli Network Manager is running with the **setuid** bit, rather than running as root
- Client-side `db2diag` might show

SQL5005C The operation failed because the database manager failed to access either the database manager configuration file or the database configuration file.

- Check the local `/etc/ld.so.conf` file for entries with “sqllib”
- That file is used to grant access to privileged libraries
- Tivoli Network Manager uses the client libraries under  
\$NCHOME/platform/linux2x86/db2client/sqllib
- If `sqllib` directories from other DB2 installations are listed, it can cause problems

Figure B-6. Troubleshooting: Problems accessing DB2 (1 of 3)

## Troubleshooting: Problems accessing DB2 (2 of 3)

- Verify that the DB2 instance is running on the expected port by running this command on the DB2 server: `netstat -an | grep <db2ClientConnectionPort>`
  - This command can result in a port that is marked as **listen**
- Verify that that port can be reached from the client by running this command on the client server: `telnet <db2ServerHostName> <db2clientConnectionPort>`
  - If this command returns **Connection refused**, the port cannot be accessed
  - If it hangs saying **Connected to db2ServerHostName**, the port can be reached

Figure B-7. Troubleshooting: Problems accessing DB2 (2 of 3)

## Key DB2 concepts: Logging and configuration

- DB2 provides various diagnostic tools for processing log files and configuration, such as db2diag
  - <http://tinyurl.com/p5pdf34>
- Data is present on each of the DB2 servers but also on the Tivoli Network Manager server
  - When troubleshooting, the most useful messages might be on the client Tivoli Network Manager server rather than the DB2 NCIM server
- The same is true of configuration, such as:

```
db2 list db directory
db2 list node directory show detail
db2 get dbm cfg
```

Figure B-8. Key DB2 concepts: Logging and configuration

## Configuring a DB2 server for HADR: Logs

- The following initial commands are required on the primary and standby DB2 servers

```
db2 update db cfg for dbName using logindexbuild on
```

```
db2 update db cfg for dbName using logarchmeth1 disk:archiveLogDir
```

- The `archiveLogDir` must exist with access permissions for the database instance owner

- Verify with:

```
db2 get db cfg for dbName | grep LOG
```

Figure B-9. Configuring a DB2 server for HADR: Logs

## Troubleshooting: Problems accessing DB2 (3 of 3)

- After setting logarchmeth1 for the database, any attempt to connect gives the error  
SQL1116N A connection to or activation of database "dbName" cannot be made because of BACKUP PENDING.
- After setting the **logarchmeth1**, a backup of the database must be created before new connections are allowed (see next slides)

Figure B-10. Troubleshooting: Problems accessing DB2 (3 of 3)

## Creating the standby DB2 database (1 of 3)

1. A DB2 instance must be available
  - Creating DB2 users: <http://tinyurl.com/k99uxnv>
  - Creating a DB2 instance: <http://tinyurl.com/k8xkvya>
2. With the automatic client reroute mechanism, the standby database is defined from an initial snapshot of the primary
  - <http://tinyurl.com/l2pkkao>

Figure B-11. Creating the standby DB2 database (1 of 3)

## Creating the standby DB2 database (2 of 3)

- On the primary DB2 server:
  - Stop all processes that are connected to the primary DB2
  - Take an offline backup of the primary NCIM database on the primary DB2 server with:

```
db2 deactivate db <dbName>
db2 backup db <dbName> to <backupDir>
```
  - Successful execution gives a time stamp: “Backup successful. The timestamp for this backup image is : 20130828152105”
- This backup command creates a file in *backupDir* with the database name, user, and time stamp in the file name
  - Copy that snapshot file to the standby DB2 server

Figure B-12. Creating the standby DB2 database (2 of 3)

## Creating the standby DB2 database (3 of 3)

- On the standby DB2 server:
  - Set the communications protocol and service name
  - Ensure that the standby user has the required privileges:  
`db2set DB2_RESTORE_GRANT_ADMIN_AUTHORITIES=ON`  
▪ <http://www.ibm.com/support/docview.wss?uid=swg21568865>
- Restore the standby database from the snapshot of the primary  
`db2 restore db <dbName> from <dirContainingPrimarySnapshot> taken at <timestampl>  
replace history file`
- The standby database has the same name and database configuration as the primary

Figure B-13. Creating the standby DB2 database (3 of 3)

## Troubleshooting: Creating the standby DB

- The backup of the primary database fails with error:

SQL1035N The database is in use

- Ensure that no connections exist to the database, then run:

db2 deactivate database **dbName**

- The restore command fails with error:

SQL1051N The path "<dirFromPrimary>" does not exist or is not valid.

- Any attempt to connect to the standby gives the error:

SQL1117N A connection to or activation of database "dbName" cannot be made because of ROLL-FORWARD PENDING

- Do **not** issue a **rollforward** command
- <http://tinyurl.com/nxndclu>

Figure B-14. Troubleshooting: Creating the standby DB

## Key DB2 concepts: HADR

- DB2 HADR replicates data from a primary DB2 server to a standby DB2 server
- The server that is primary can be altered by executing the command:

```
db2 takeover hadr on db dbName
```

Figure B-15. Key DB2 concepts: HADR

## Configuring HADR hosts and services

- In all configuration, hosts should be consistently referred to
  - Avoid mixing IPs with host names
  - Avoid mixing shortened names with fully qualified names
- The client should list both DB2 servers in /etc/hosts
  - <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0011976.html>
- In addition to the port used for client connections to the DB2 instance, each database requires a port or service for HADR communication
  - Cannot be the same port as used for client connections or that port value + 1
- In a NAT environment it might be necessary to set:

```
db2set DB2_HADR_NO_IP_CHECK=on
```

- <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0054092.html>

Figure B-16. Configuring HADR hosts and services

## Configuring HADR

- The following commands need to be run as the instance owner on both the primary and standby DB2 servers

```
db2 update db cfg for dbName using hadr_local_host localhost
db2 update db cfg for dbName using hadr_local_svc
localPortOrServiceForHadrComms
db2 update db cfg for dbName using hadr_remote_host remoteHost
db2 update db cfg for dbName using hadr_remote_svc
remotePortOrServiceForHadrComms
db2 update db cfg for dbName using hadr_syncmode NEARSYNC
db2 update db cfg for dbName using hadr_timeout 120
db2 update db cfg for dbName using hadr_peer_window 120
```

Figure B-17. Configuring HADR

## Key DB2 concepts: ACR

- Automatic client reroute reroutes connections to the primary database of a HADR pair
- The primary database can be changed with a `db2 takeover hadr` command
  - Clients automatically connect to the new primary
- To configure, run this command on both the primary and standby DB2 servers:

```
db2 update alternate server for database dbName using hostname remoteHost port
remotePortNumberForClientConnect
```

Figure B-18. Key DB2 concepts: ACR

## Enabling HADR

- With the configuration in place, HADR can now be started on both DB2 servers

- The standby server MUST be started first

- On the standby DB2 server:

```
db2 start hadr on db <dbName> as standby
```

- On the primary DB2 server:

```
db2 start hadr on db <dbName> as primary
```

- Note:** Connections cannot be made to the standby server when it is in standby

Figure B-19. Enabling HADR

## Troubleshooting: HADR fails to start

- If HADR fails to start, check the value of the log variables:  
`db2 get db cfg for dbName | grep LOG`
- These variables are set for the DB itself, not for the instance
- Unless it was manually altered since the restore, the value on the standby is the value that was set on the primary at the time the backup was made

Figure B-20. Troubleshooting: HADR fails to start

## Troubleshooting: Validate HADR configuration

- Run the same command on both the primary and standby DB2 servers

```
db2 get db cfg for dbName | grep HADR
```

- Cross-reference the configured values to check for consistency

- The “Alternate server” host name and client connection port should be displayed when running this command on the primary and standby DB2 servers

```
db2 list db directory
```

Figure B-21. Troubleshooting: Validate HADR configuration

## Key DB2 concepts: Catalogs and nodes

- A client can store a reference to a database as a “node”, providing connection details

```
db2 catalog tcpip node nodeName remote hostname server service
```

- <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.qb.client.doc/doc/t0005621.html>

- A client can provide an alias for connection to a database with that node name

```
db2 catalog db <dbName> as <dbAlias> at node <nodeName>
```

- <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.dbobj.doc/doc/t0004952.html>

Figure B-22. Key DB2 concepts: Catalogs and nodes

## Configuring a client catalog

- The Tivoli Network Manager server should catalog nodes for both the primary and the standby DB2 servers:

```
db2 catalog tcpip node primaryNodeName remote primaryHostname server primaryService
db2 catalog tcpip node standbyNodeName remote standbyHostname server standbyService
```

- The Tivoli Network Manager server should catalog the primary database:

```
db2 catalog db dbName as dbAlias at node primaryNodeName
```

- Configured nodes can be viewed with:

```
db2 list node directory show detail
```

- Configured databases can be viewed with:

```
db2 list db directory
```

Figure B-23. Configuring a client catalog

## Troubleshooting: Client catalog

- When configuring a node, it returns the error:

SQL1019N The node name "*nodeName*" specified in the command is not valid.

- Although *nodeName* is arbitrary, it has limitations

- For example, it cannot be the same as the DB2 instance user

[http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.messages.sql.doc/doc/msg\\_I01019n.html](http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.messages.sql.doc/doc/msg_I01019n.html)

- A cataloged entry can be removed with

db2 uncatalog db *dbName*

db2 uncatalog node *nodeName*

Figure B-24. Troubleshooting: Client catalog

## Tivoli Network Manager back-end configuration (1 of 2)

- The Tivoli Network Manager back end connects to the NCIM database with the **DbLogins** and **MibDbLogins** files in \$NCHOME/etc/precision/
  - If domain-specific versions exist, these files are used
  - Otherwise, the default version is used
- To implement changes, edit the files manually or use ncp\_config, for example:

```
ncp_config -domain <domainName> -schema DbLoginSchema.cfg -query "update config.dbserver set m_DbName='<dbName>', m_PortNum=0 where m_DbId IN ('NCIM', 'NCMONITOR', 'POLLDATA', 'NCPGUI');"
```
- The DB2 environment is automatically picked up from the \$NCHOME/precision/.db2sql1lib file

Figure B-25. Tivoli Network Manager back-end configuration (1 of 2)

## Tivoli Network Manager back-end configuration (2 of 2)

| Field in config file | Value                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| m_Dbld               | <ul style="list-style-type: none"> <li>Do not change</li> <li>DB2 hosts "NCIM", "NCMONITOR", "POLLDATA", "NCPGUI", "MIB"</li> </ul>                                                         |
| m_Server             | Always "db2"                                                                                                                                                                                |
| m_DbName             | The locally cataloged alias for the primary NCIM DB2 server                                                                                                                                 |
| m_Schema             | Do not change unless creating multiple NCIM instances in the same database (not recommended)                                                                                                |
| m_HostName           | Primary NCIM host name                                                                                                                                                                      |
| m_UserName           | DB2 user with access to primary and standby NCIM                                                                                                                                            |
| m_Password           | <ul style="list-style-type: none"> <li>If encrypting passwords and manually updating the file, use ncp_crypt and paste in the result</li> <li>Or update the file with ncp_config</li> </ul> |
| m_PortNum            | Set to 0 to force the local catalog to be used                                                                                                                                              |
| m_EncryptedPwd       | Usually 1, indicating encrypted passwords                                                                                                                                                   |

Configuring DB2 high availability

© Copyright IBM Corporation 2016

Figure B-26. Tivoli Network Manager back-end configuration (2 of 2)

## Troubleshooting: Tivoli Network Manager back-end configuration (1 of 2)

- If the back end cannot connect to DB2, try to run `db2` from the Tivoli Network Manager server
- On the DB2 command line:
  - connect to `dbName` user `db2User`
    - This command validates that the database is cataloged
    - This command checks for access through firewalls
  - `ncp_db_access.pl` to validate the configuration
    - This script uses the configuration details in **DbLogins** and **MibDbLogins**
  - `ncp_oql -service ncim` to validate the C++ libraries
    - The `ncp_db_access.pl` script uses Perl libraries for connection
    - Additionally, `ncp_oql` requires correct library paths and permissions

Figure B-27. Troubleshooting: Tivoli Network Manager back-end configuration (1 of 2)

## Troubleshooting: Tivoli Network Manager back-end configuration (2 of 2)

- Always read the returned error message
- Known errors from Tivoli Network Manager processes

Cannot connect to database : 08001 (Error -1013): [IBM][CLI Driver] SQL1013N The database alias name or database name "nodeName" could not be found.

- Check that the \$NCHOME/precision/.db2sql1ib file is sourcing the expected profile

Figure B-28. Troubleshooting: Tivoli Network Manager back-end configuration (2 of 2)

## Tivoli Integrated Portal configuration

- To connect to DB2 HADR, the same manual change is required to two Tivoli Integrated Portal property files in \$NCHOME/precision/profiles/TIPProfile/etc/tnm
  - Add the following line to tnm.properties:
 

```
tnm.database.jdbc.url=jdbc:db2://<primaryDb2Host>:<primaryDb2Port>/<dbName>;clientRerouteAlternateServerName=<standbyDb2Host>;clientRerouteAlternatePortNumber=<standbyDb2Port>;
```
  - Add the following line to ncpolldata.properties:
 

```
ncpolldata.database.jdbc.url=jdbc:db2://<primaryDb2Host>:<primaryDb2Port>/<dbName>;clientRerouteAlternateServerName=<standbyDb2Host>;clientRerouteAlternatePortNumber=<standbyDb2Port>;
```
- Restart Tivoli Integrated Portal after changes with
  - itnm\_stop tip / itnm\_start tip
- Tivoli Integrated Portal connects with the database name and ports, instead of through the catalog

*Figure B-29. Tivoli Integrated Portal configuration*

## Failing over NCIM

- To make the standby DB2 server primary, run this command on the standby DB2 server:  
`db2 takeover hadr on db dbName`
  - Connections to the former primary NCIM are disconnected, and reestablished as needed on the new primary NCIM
  - Transition is seamless to Tivoli Network Manager
  - Works only if primary and backup are both up and are peering
- NCIM failover with DB2 HADR is independent of Tivoli Network Manager failover
- DB2 HADR mechanisms replicate changes that are made to the database from the current primary NCIM to the standby
  - Poll policies can be configured on either NCIM
  - Network views can be created on either NCIM
  - Manual topology changes can be made on either NCIM

Figure B-30. Failing over NCIM

## Key DB2 concept: Clustering with Tivoli System Automation for Multiplatform

- Tivoli System Automation for Multiplatform provides clustering for DB2
  - It can be used to share data storage across DB2 servers
  - It can be used to configure a virtual IP address to access DB2 servers
  - It can be used to trigger takeover scripts upon failure
- Tivoli System Automation for Multiplatform is bundled with DB2 10.1

Figure B-31. Key DB2 concept: Clustering with Tivoli System Automation for Multiplatform

## Failing over NCIM without manual intervention

- An ACR limitation is that failure of the primary DB2 server does not automatically cause the standby server to become active
- If the primary DB2 server fails, this command **must** be manually executed on the standby DB2 server
  - db2 takeover hadr on db <dbName> by force
- This takeover can be automated with Tivoli System Automation for Multiplatform
  - Configured with db2haicu
  - Check configuration with db2pd -ha
- With Tivoli System Automation for Multiplatform, it is not necessary to:
  - Configure a virtual IP with Tivoli System Automation for Multiplatform if all clients support ACR (Tivoli Network Manager back end and Tivoli Integrated Portal do if configured, as explained in these slides)
  - Share storage, as HADR handle replication of data

Figure B-32. Failing over NCOM without manual intervention

## Troubleshooting: HADR state and connected processes

- To determine the current HADR state of a DB2 server, run:

```
db2pd -db dbName -hadr
db2 get db cfg for dbName | grep HADR
```

- To determine what processes are currently connected to a specific DB2 server, run:

```
db2 list applications
```

- Individual applications (such as **ncp\_poller**) can have multiple connections
- Processes reconnect only when a connection is used
- All processes should be connected to the DB2 server that is PRIMARY
- A `db2 takeover hadr` command might be required if neither server shows connections

Figure B-33. Troubleshooting: HADR state and connected processes

## Troubleshooting: Connections to alternative server

- Tivoli Network Manager back-end logs contain messages such as:

```
[IBM] [CLI Driver] [DB2/LINUXX8664] SQL1773N The statement or command requires functionality that is not supported on a read-enabled HADR standby database. Reason code = "1"
```

- DB2 has some parameters that allow read-access to the standby server

```
db2set DB2_HADR_ROS=ON
```

```
db2set DB2_STANDBY_ISO=UR
```

- These commands should not be set as they can cause problems with disconnecting from the standby server and reconnecting to the primary

Figure B-34. Troubleshooting: Connections to alternative server

## Failing over Tivoli Network Manager

The same limitations exist as for non-replicating failover in earlier versions of Tivoli Network Manager:

- Discovery runs only on the primary server
- Tivoli Integrated Portal connects to the back-end **ncp\_config** process through the **domainMgr** table
  - The **domainHost** and **domainPort** field are **not** automatically updated to identify the backup Tivoli Network Manager
- Webtools do not work when connected to the backup Tivoli Network Manager

Figure B-35. Failing over Tivoli Network Manager

## Topic summary

- NCIM failover is independent of Tivoli Network Manager failover
- DB2 HADR provides replication of data from the primary NCIM to standby/backup NCIM
- DB2 ACR provides rerouting of client connections to the appropriate primary NCIM server
- Tivoli System Automation for Multiplatform automatically promotes a standby DB2 server to primary when the primary fails

*Figure B-36. Topic summary*

## Reference material (1 of 2)

- Redbook with clear examples and diagrams for configuration steps:  
<http://www.redbooks.ibm.com/redbooks/pdfs/sg247363.pdf>
  - Step-by-step HADR configuration in section 6.2
  - Troubleshooting in section 6.4
  - Clustering with Tivoli System Automation for Multiplatform in Chapter 7
  - ACR explanation in Chapter 10

Figure B-37. Reference material (1 of 2)

## Reference material (2 of 2)

- DB2 10.1 docs on HADR
  - <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.ha.doc/doc/t0051350.html>
- DB2 10.1 docs on configuring HADR with ACR
  - Includes example command-line calls
  - <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.ha.doc/doc/t0011725.html>
- Concise existing presentation on NCIM HA, including DB2 HADR configuration tips
  - <http://tinyurl.com/y7kqox2w>

Configuring DB2 high availability

© Copyright IBM Corporation 2016

Figure B-38. Reference material (2 of 2)

# Appendix C. List of abbreviations

|              |                                              |
|--------------|----------------------------------------------|
| <b>ABR</b>   | area border router                           |
| <b>ACL</b>   | access control list                          |
| <b>ACR</b>   | automatic client reroute                     |
| <b>AEL</b>   | Active Event List                            |
| <b>AEN</b>   | Accelerated Event Notification               |
| <b>AIX</b>   | Advanced IBM UNIX                            |
| <b>AMD</b>   | Advanced Micro Devices                       |
| <b>AOC</b>   | Active Object Class                          |
| <b>ARP</b>   | Address Resolution Protocol                  |
| <b>AS</b>    | autonomous systems                           |
| <b>ASBR</b>  | autonomous system border router              |
| <b>ATM</b>   | asynchronous transfer mode                   |
| <b>BDR</b>   | backup designated router                     |
| <b>BGP</b>   | Border Gateway Protocol                      |
| <b>BI</b>    | Business Intelligence                        |
| <b>BIRT</b>  | Business Intelligence and Reporting Tools    |
| <b>BOM</b>   | business object model                        |
| <b>BRI</b>   | Basic Rate Interface                         |
| <b>CAM</b>   | content-addressable memory                   |
| <b>CCMDB</b> | Change and Configuration Management Database |
| <b>CDP</b>   | Cisco Discovery Protocol                     |
| <b>CDP</b>   | Continuous Data Protection                   |
| <b>CE</b>    | customer edge                                |
| <b>CE</b>    | customer equipment                           |
| <b>CMTS</b>  | Cable Modem Termination System               |
| <b>CORBA</b> | Common Object Request Broker Architecture    |
| <b>CPU</b>   | central processing unit                      |
| <b>CSV</b>   | comma-separated values                       |
| <b>CTP</b>   | connection termination point                 |
| <b>DASH</b>  | Dashboard Application Services Hub           |

|              |                                                    |
|--------------|----------------------------------------------------|
| <b>DB</b>    | database                                           |
| <b>DLA</b>   | Discovery Library Adapter                          |
| <b>DNCIM</b> | Discovery Network Connectivity and Inventory Model |
| <b>DNS</b>   | domain name server                                 |
| <b>DNS</b>   | Domain Name Server                                 |
| <b>DoS</b>   | denial-of-service                                  |
| <b>DR</b>    | designated router                                  |
| <b>DSA</b>   | data source adapter                                |
| <b>EIGRP</b> | Enhanced Interior Gateway Routing Protocol         |
| <b>EMS</b>   | element management system                          |
| <b>EMS</b>   | Entity Management System                           |
| <b>ESE</b>   | Enterprise Server Edition                          |
| <b>ESR</b>   | extended support release                           |
| <b>FIPS</b>  | Federal Information Processing Standard            |
| <b>GB</b>    | gigabyte                                           |
| <b>GNU</b>   | GNU's Not UNIX                                     |
| <b>GPRS</b>  | General Packet Radio Service                       |
| <b>GSM</b>   | Global System for Mobile                           |
| <b>GUI</b>   | graphical user interface                           |
| <b>HADR</b>  | high availability disaster recovery                |
| <b>HQ</b>    | headquarters                                       |
| <b>HSRP</b>  | Hot Standby Routing Protocol                       |
| <b>HTML</b>  | Hypertext Markup Language                          |
| <b>HTTP</b>  | Hypertext Transfer Protocol                        |
| <b>HTTPS</b> | Hypertext Transfer Protocol Secure                 |
| <b>HVAC</b>  | heating, ventilation, and air conditioning         |
| <b>IANA</b>  | Internet Assigned Numbers Authority                |
| <b>IBM</b>   | International Business Machines Corporation        |
| <b>ICMP</b>  | Internet Control Message Protocol                  |
| <b>IDS</b>   | Informix Dynamic Server                            |
| <b>IDUC</b>  | Insert, Delete, Update, or Control                 |
| <b>IGMP</b>  | Internet Group Management Protocol                 |
| <b>IGRP</b>  | Interior Gateway Routing Protocol                  |
| <b>ILO</b>   | instructor-led online                              |

|              |                                               |
|--------------|-----------------------------------------------|
| <b>ILT</b>   | instructor-led training                       |
| <b>IP</b>    | Internet Protocol                             |
| <b>IPL</b>   | Impact Policy Language                        |
| <b>IPM</b>   | Internet Protocol Multicast                   |
| <b>ISDN</b>  | Integrated Services Digital Network           |
| <b>ISP</b>   | internet service provider                     |
| <b>IT</b>    | information technology                        |
| <b>ITIL</b>  | Information Technology Infrastructure Library |
| <b>JDBC</b>  | Java Database Connectivity                    |
| <b>JRE</b>   | Java runtime environment                      |
| <b>KPI</b>   | key performance indicator                     |
| <b>LAN</b>   | local area network                            |
| <b>LMS</b>   | LAN Management Solution                       |
| <b>LSA</b>   | link state advertisement                      |
| <b>LTE</b>   | long-term evolution                           |
| <b>MAC</b>   | Media Access Control                          |
| <b>MDT</b>   | multicast distribution trees                  |
| <b>MIB</b>   | Management Information Base                   |
| <b>MPLS</b>  | Multiprotocol Label Switching                 |
| <b>MQTT</b>  | MQ (Message Queue) Telemetry Transport        |
| <b>MSP</b>   | managed service provider                      |
| <b>MTOSI</b> | Multi-Technology Operations System Interface  |
| <b>MTTR</b>  | mean time to repair                           |
| <b>MTU</b>   | maximum transmission unit                     |
| <b>NAT</b>   | network address translation                   |
| <b>NBI</b>   | Northbound Interface                          |
| <b>NCIM</b>  | Network Connectivity and Inventory Model      |
| <b>NcKL</b>  | Netcool Knowledge Library                     |
| <b>NM</b>    | Network Management                            |
| <b>NOC</b>   | network operations center                     |
| <b>NSA</b>   | National Security Administration              |
| <b>NSA</b>   | network session accounting                    |
| <b>NSN</b>   | Nokia Solutions and Networks                  |
| <b>NTP</b>   | Network Time Protocol                         |

|                |                                           |
|----------------|-------------------------------------------|
| <b>OID</b>     | Object Identifier                         |
| <b>OQL</b>     | Object Query Language                     |
| <b>OSI</b>     | open systems interconnection              |
| <b>OSLC</b>    | Open Services for Lifecycle Collaboration |
| <b>OSPF</b>    | Open Shortest Path First                  |
| <b>PC</b>      | personal computer                         |
| <b>PDF</b>     | Portable Document Format                  |
| <b>PE</b>      | premise equipment                         |
| <b>PE</b>      | provider edge                             |
| <b>PIM</b>     | protocol-independent multicast            |
| <b>POC</b>     | proof of concept                          |
| <b>PTP</b>     | physical termination point                |
| <b>RAID</b>    | Redundant Array of Independent Disks      |
| <b>RAM</b>     | random access memory                      |
| <b>RAN</b>     | Radio Access Network                      |
| <b>RARP</b>    | Reverse Address Resolution Protocol       |
| <b>RCA</b>     | root cause analysis                       |
| <b>RDBMS</b>   | relational database management system     |
| <b>RIP</b>     | Routing Information Protocol              |
| <b>RM</b>      | Ring Manager                              |
| <b>RP</b>      | rendezvous point                          |
| <b>RPC</b>     | Remote Procedure Call                     |
| <b>RSA</b>     | Rivest-Shamir-Adleman                     |
| <b>RSMB</b>    | Really Small Message Broker               |
| <b>RT</b>      | route targets                             |
| <b>SAE</b>     | service-affecting event                   |
| <b>SAM</b>     | Service Aware Manager                     |
| <b>SDK</b>     | software development kit                  |
| <b>SELinux</b> | Security-Enhanced Linux                   |
| <b>SLA</b>     | service level agreement                   |
| <b>SLES</b>    | SuSE Linux Enterprise Server              |
| <b>SNMP</b>    | Simple Network Management Protocol        |
| <b>SQL</b>     | Structured Query Language                 |
| <b>SSH</b>     | Secure Shell                              |

|             |                                            |
|-------------|--------------------------------------------|
| <b>SSM</b>  | System Service Monitor                     |
| <b>SVG</b>  | Scalable Vector Graphics                   |
| <b>TCP</b>  | Transmission Control Protocol              |
| <b>TE</b>   | Traffic Engineering                        |
| <b>TMF</b>  | Transaction Manager Facility               |
| <b>UCD</b>  | User Centered Design                       |
| <b>UMTS</b> | Universal Mobile Telecommunications System |
| <b>UNIX</b> | Uniplexed Information and Computing System |
| <b>UOW</b>  | unit of work                               |
| <b>UPS</b>  | uninterruptible power supply               |
| <b>URI</b>  | Uniform Resource Identifier                |
| <b>URL</b>  | Uniform Resource Locator                   |
| <b>VLAN</b> | virtual local area network                 |
| <b>VPLS</b> | Virtual Private Label Switching            |
| <b>VPLS</b> | Virtual Private LAN Service                |
| <b>VPN</b>  | Virtual Private Network                    |
| <b>VRF</b>  | virtual routing and forwarding             |
| <b>WAN</b>  | wide area network                          |
| <b>WSE</b>  | Workgroup Server Edition                   |
| <b>XML</b>  | Extensible Markup Language                 |
| <b>XSS</b>  | cross-site scripting                       |



IBM Training

**IBM**  
®

© Copyright International Business Machines Corporation 2017.