



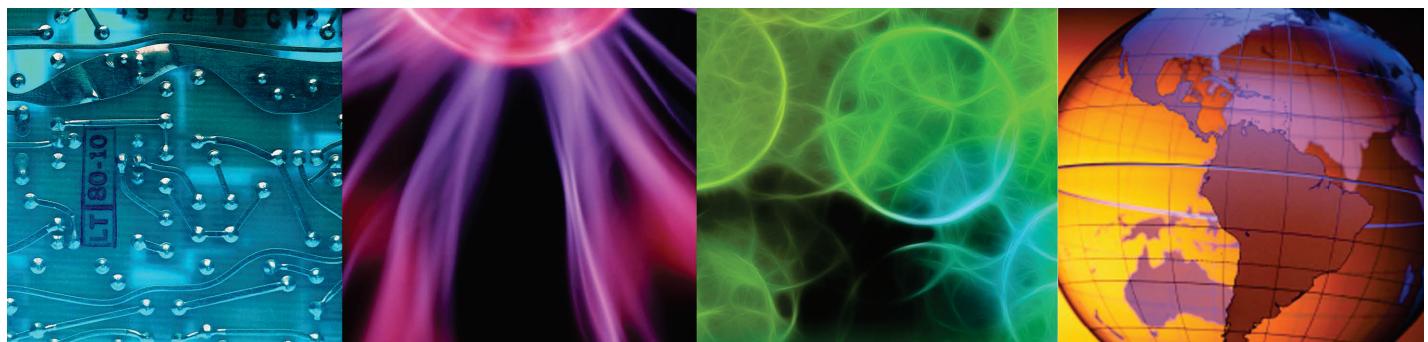
IBM Training

IBM Tivoli Netcool/OMNIbus 8.1 Installation and Configuration

Student Notebook

Course code TN025 ERC 1.0

November 2014



Cloud & Smarter Infrastructure

All files and material for this course are IBM copyright property covered by the following copyright notice.

© Copyright IBM Corp. 2014. All Rights Reserved.

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

IBM, the IBM logo, and ibm.com are 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.

Trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

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

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

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results.



Contents

About this course	xii
About the student	xiii
Learning objectives	xiv
Course agenda	xv
There are no student exercises for this unit	
1 Basic Tivoli Netcool/OMNibus components and concepts	1
Objectives	2
Lesson 1 Overview	3
Tivoli Netcool/OMNibus overview	4
Tivoli Netcool/OMNibus software components	5
ObjectServer	6
Desktop	7
Web GUI	8
Dashboard Application Services Hub	9
Probes	10
Gateways	11
Online help system	12
Summary	13
2 Installation prerequisites and planning	14
Objectives	15
Lesson 1 Preinstallation planning	16
Preinstallation planning	17
System requirements	18
Supported operating systems	19
Select product and version	20
View the results	21
Hardware requirements	22
View the result	23
Additional software requirements	24
View results	25
Mobile device support	26
Prerequisite scanner	27
Prerequisite scanner: Command line	28
Prerequisite scanner: Example of a pass	29
Prerequisite scanner: Example of a failure	30
Prerequisite scanner: More information	31
Lesson 2 Other considerations	32

Architecture considerations	33
Upgrade considerations	36
Post-upgrade steps	37
Student exercises	38
Summary	39
3 Netcool/OMNibus core installation.....	40
Objectives	41
Lesson 1 Preinstallation requirements	42
Installation overview	43
IBM Installation Manager	45
Installation Manager user modes	46
Installation options	47
Preinstallation steps	48
Lesson 2 Core installation	49
Netcool/OMNibus installer	50
Packages	51
License agreement	52
Installation Manager directory locations	53
Netcool/OMNibus installation directory	54
Netcool/OMNibus features	55
Data migration	56
Installation summary	57
Installation complete	58
Initial Configuration Wizard	59
Task options	60
ObjectServers	61
Host computers	62
Process agents	63
Primary aggregation ObjectServer	64
Backup aggregation ObjectServer and synchronizer gateway	65
Summary	66
Output files	67
Apply the configuration	68
Lesson 3 Preliminary operation	69
Tivoli Netcool/OMNibus communications	70
IDUC in firewall environments	71
Configuring the interfaces file	72
Defining the interfaces file on a server	73
Defining the interfaces file on a client	74
ObjectServer names	75
Building the ObjectServer	76
Starting a Netcool/OMNibus system	77
Student exercises	78
Summary	79
4 Installing Netcool/OMNibus Web GUI.....	80
Objectives	81
Lesson 1 Installing Netcool/OMNibus Web GUI	82

Installation	83
Installing Jazz for Service Management	85
Installation options	86
Custom installation	87
Jazz for Service Management Home Location	88
Source Locations	89
Select Operations	90
Installation Locations	91
Administrator user	92
Installation tasks	93
Review results	94
Dashboard Application Services Hub	95
Installing Web GUI	96
Install packages	97
Package group	98
Package features	99
Administrator credentials	100
Installation summary	101
Installation complete	102
Installation log file	103
Postinstallation configuration	104
Postinstallation steps	105
Configuration summary	106
Postinstallation steps	107
UNIX/Linux aliases	108
Register Java plug-in	109
Dashboard Application Services Hub access	110
Student exercises	111
Lesson 2 Web GUI user features and functions	112
Web GUI user authority	113
Event Dashboard	114
Active Event List	117
Event Viewer	118
Mobile access to event list	119
Lesson 3 Web GUI administrative features and functions	120
Typical Web GUI administration	121
Web GUI administrative authority	122
Dashboard Application Services Hub administrative authority	123
Creating groups and users	124
Creating a group	125
Creating a user	126
Assigning roles to a group	127
Creating a view	128
Creating a filter	129
Using a filter and view	130
Creating a map	131
Adding a map to a page	132
Lesson 4 Web GUI user authentication	134

Web GUI user authentication	135
ObjectServer user synchronization	136
ObjectServer group synchronization	137
Web GUI roles	138
Web GUI group roles	139
LDAP as an authentication source	140
Configuring LDAP as an authentication source	141
Removing the ObjectServer from the Virtual Member Manager realm	142
Adding LDAP to the Virtual Member Manager realm: Step 1	143
Adding LDAP to the Virtual Member Manager realm: Step 2	144
Adding LDAP to the Virtual Member Manager realm: Step 3	145
Adding LDAP to the Virtual Member Manager realm: Step 4	146
Adding LDAP to the Virtual Member Manager realm: Step 5	147
Synchronizing LDAP users with the ObjectServer	148
Student exercises	149
Summary	150
5 Basic ObjectServer administration	151
Objectives	152
Lesson 1 Netcool/OMNIbus directory structure	153
\$NCHOME directory	154
\$NCHOME/bin directory	155
\$NCHOME/etc and \$NCHOME/platform directories	156
\$NCHOME/license and \$NCHOME/properties	157
\$OMNIHOME directory	158
\$OMNIHOME/bin directory	159
\$OMNIHOME/db and \$OMNIHOME/desktop directories	160
\$OMNIHOME/etc and \$OMNIHOME/gates directories	161
\$OMNIHOME/extensions directory	162
\$OMNIHOME/install and \$OMNIHOME/java directories	163
\$OMNIHOME/log and \$OMNIHOME/platform directories	164
\$OMNIHOME/probes, \$OMNIHOME/tsm, \$OMNIHOME/utils, and \$OMNIHOME/var directories	165
Lesson 2 ObjectServer structure	166
Object Server databases	167
Database and table restrictions	168
ObjectServer structure: alerts.status	169
ObjectServer structure: alerts.details	170
ObjectServer structure: alerts.journal	171
Default ObjectServer fields	172
Data storage	176
Region storage and physical checkpoints	178
Deduplication	180
Deduplication behavior	181
Properties	182
Properties using nco_sql	184
Modifying an ObjectServer property	186
Important properties	187
ObjectServer OSLC interface overview	191

Important OSLC properties	192
Lesson 3 Administration overview	193
Basic ObjectServer administration	194
User administration	195
User configuration requirements	196
Connection wizard	198
Tivoli Netcool/OMNIbus Administrator	199
Accessing the Users tab	200
Restriction filters	201
Roles	202
Groups	203
Creating a new group	204
Adding new users	206
User administration when using LDAP	207
Creating groups	209
Creating users	210
Assigning roles	211
ObjectServer synchronization	212
ObjectServer, Web GUI, Dashboard Application Services Hub	213
Lesson 4 Modifying the ObjectServer	214
Modifying the ObjectServer	215
Adding ObjectServer resources	217
MyDataBase.MyTable	218
Student exercises	219
Summary	220
6 Tools and automations	221
Objectives	222
Lesson 1 Introduction to ObjectServer SQL	223
ObjectServer SQL	224
Accessing the ObjectServer with the command-line	225
Viewing data using SELECT	227
Fields and operators	228
IN operator and subqueries	230
Modifying data using UPDATE	231
Creating data using INSERT	232
Lesson 2 Desktop tools	234
Native desktop tools	235
Native desktop menu options	236
Web GUI tools	237
Creating a Web GUI tool	238
Command tool example	239
SQL tool example	240
CGI tool example	241
CGI command registration	242
Limiting tool access by group	243
Limiting tool access by class	244
Creating a tool prompt	245

Using a prompt in a tool	246
Adding a tool to a menu	247
Lesson 3 ObjectServer automations	249
Trigger groups	250
Triggers	251
How triggers work	252
Temporal Trigger	254
Temporal Trigger example	256
Database Trigger	257
Database Trigger example	258
Signal Trigger	259
Signal Trigger example	260
User Defined Signal	261
Event correlation	262
Automation and trigger best practices	263
Procedures	265
SQL procedure example	266
External procedure example	267
Journal entry by automations	269
Student exercises	270
Summary	271
7 Common integrations	272
Objectives	273
Lesson 1 Overview	274
Common integrations	275
Probe operation	277
Probe basics	279
Probe properties	281
Rules file	283
Testing and implementing new rules	284
A rules file example	286
Tokens and fields	287
Testing and logic in a rules files	288
switch/case statements	289
Control functions: Discard, recover, update, details	290
Extract and exists functions	291
Lookup tables	293
Multicolumn lookups and defaults	294
Properties and writable properties	295
Associative arrays	297
Additional functions	298
Events to alternate ObjectServers and tables	299
generic_clear trigger	300
Lesson 2 Common probes	301
Syslog probe	302
Syslog next generation	303
SNMP probe	304

Tivoli EIF probe	305
Netcool/OMNIbus Knowledge Library	306
MIB manager	308
Lesson 3 Optional probe features	309
Optional probe features	310
Installation	311
Configuration	312
Usage	314
Limitations	315
Example of usage of Syslog probe	316
Lesson 4 Remote probe administration	318
Features to facilitate remote administration	319
Rules file caching	320
Bidirectional communication	321
Probe registry	322
Bidirectional probe communication usage	323
Student exercises	325
Summary	326
8 ObjectServer high availability	327
Objectives	328
Lesson 1 Overview	329
Architecture overview	330
Virtual ObjectServer	331
Failover process	332
Fallback process	333
Configuring interfaces file for failover	334
How nco_xigen shows failover	335
Configuring client processes for failover	336
Configuring the Web GUI component: Option 1	337
Configuring the Web GUI component: Option 2	338
Lesson 2 Implementing high availability	339
Initial Configuration Wizard	340
Steps for implementing high availability	341
Configuration pack, nco_confpack	343
Important BackupObjectServer property	345
BackupObjectServer property	346
Configuring properties for failback	347
Lesson 3 Gateway configuration	348
Bidirectional or synchronizer gateway	349
ObjectServer gateway determination of resync direction	351
Gateway data movement techniques	352
Configuring resynchronization	354
Bidirectional ObjectServer gateway configuration	356
Bidirectional gateway configuration files	357
Automations in HA configurations	359
Automated failover and failback	360
Initial setup	361

Automated failover: Primary goes down	362
Automated fallback: Primary comes up	363
Automated failover and fallback key concepts	365
High availability and data integrity options	366
Student exercises	368
Summary	369
9 Process control	370
Objectives	371
Lesson 1 Overview	372
Process control	373
Process control setup example	374
The nco_pa.conf file	375
The nco_process section	376
Second nco_process section	377
The nco_service section	378
The nco_security and nco_routing sections	379
Configuring authentication	381
Process agent daemon	383
Running external procedure on server2	384
Process control command line utilities	387
Configuring automated start	389
Initial Configuration Wizard	390
Lesson 2 Visual Process Activity (PA)	391
Visual PA: Logging in	392
Visual PA: Adding a process	393
Visual PA: New process configuration	394
Visual PA: Starting the process	395
Student exercises	396
Summary	397
10 Event archiving	398
Objectives	399
Lesson 1 Overview	400
Historical event reporting	401
Components	402
Implementation	403
Lesson 2 Tivoli Common Reporting	404
Tivoli Common Reporting architecture	405
Installing Tivoli Common Reporting	406
Installing Jazz and Tivoli Common Reporting	407
Installing Tivoli Common Reporting into an existing Jazz deployment	408
Accessing Tivoli Common Reporting	409
Student exercises	410
Lesson 3 Gateway for JDBC	411
Gateway for JDBC audit mode of operation	412
Gateway for JDBC reporting mode of operation	413
Archive database configuration	414
REPORTER database configuration	415

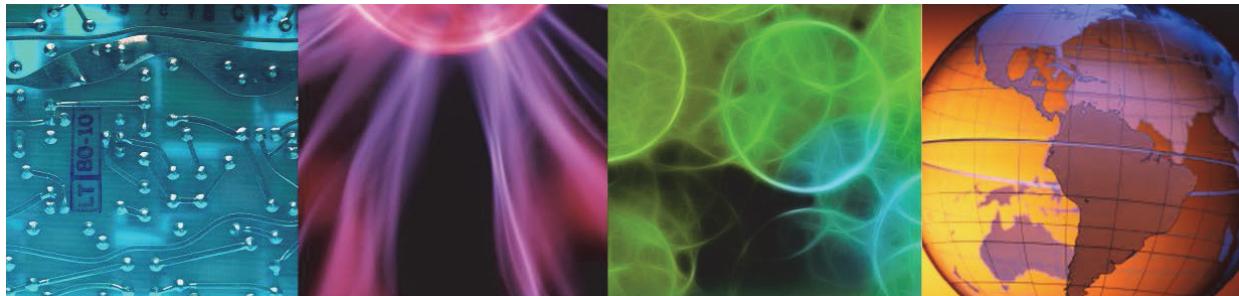
REPORTER database configuration, continued	416
AUDIT tables	417
Text conversion tables	419
Gateway installation and configuration	420
JDBC_GATE.map file	421
JDBC_GATE.props file	422
JDBC_GATE.rdwtr.tblrep.def file	423
JDBC_GATE.startup.cmd file	424
Gateway operation	425
Lesson 4 Event reports	426
Event reports	427
Import report package	428
Create data source	429
Test data source	430
Verify reports	431
Verify reports	432
Reports included with the product	433
Student exercises	435
Summary	436
Appendix A Software installation files	437



About this course



IBM Tivoli Netcool/OMNIbus V8.1 Installation and Configuration



© Copyright IBM Corporation 2014

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

This 3-day instructor-led course teaches you to perform a complete installation of IBM® Tivoli® Netcool®/OMNIbus 8.1. Through lectures and extensive hands-on exercises, you learn all the steps necessary to perform an installation, including hardware sizing, confirming software prerequisites, installation, creation of ObjectServers, installation of probes, installation of gateways, configuring high availability, and deploying historical event reporting.

Throughout the course, you practice your knowledge through extensive hands-on exercises that emphasize the skills taught in the lectures.

The lab environment for this course uses the Red Hat Linux platform.

For information about other related courses, visit the Cloud & Smarter Infrastructure education training paths website:

ibm.com/software/software/tivoli/education/

Details	
Delivery method	Classroom or instructor-led online (ILO)
Course level	ERC 1.0
	This course is an update of the following previous course: TN024: IBM Tivoli Netcool/OMNIbus 7.4 Installation and Configuration
Product and version	IBM Tivoli Netcool/OMNIbus 8.1
Duration	3 days
Skill level	Intermediate

About the student

This course is designed for anyone given the task of installing Tivoli Netcool/OMNIbus.

Before taking this course, make sure that you have the following skills:

- Practical knowledge of UNIX and typical UNIX-based tools
- Practical knowledge of network fault management
- Basic understanding of database concepts, including language constructs, database versus table versus column, triggers
- IBM Tivoli Netcool/OMNIbus V8.1 User course or equivalent experience

Learning objectives

Course objectives

In this course, you learn to perform the following tasks:

- Install Netcool/OMNibus Core component
- Create ObjectServer instances
- Install Netcool/OMNibus Web GUI
- Configure and demonstrate ObjectServer high availability
- Install Netcool/OMNibus integrations
- Create users and groups
- Create menus and tools
- Create ObjectServer triggers and trigger groups
- Configure ObjectServer event archiving
- Install and use Tivoli Common Reporting and Netcool/OMNibus reports

Course agenda

The course contains the following units:

1. [Basic Tivoli Netcool/OMNibus components and concepts](#)

This unit provides an introduction to the features, and functions of Netcool/OMNibus.

There are no student exercises for this unit.

2. [Installation prerequisites and planning](#)

In this unit, you learn about the hardware, and software requirements for the installation of Netcool/OMNibus.

The exercises in this unit validate the host configurations before installing the Netcool/OMNibus components.

3. [Netcool/OMNibus core installation](#)

This unit teaches the installation of Netcool/OMNibus core, the creation of an ObjectServer, and the steps to verify basic function.

In this unit, you learn how to install the Netcool/OMNibus core component, create an ObjectServer, and validate the installation.

4. [Installing Netcool/OMNibus Web GUI](#)

This unit covers the installation of Web GUI and a brief introduction to the features of the product.

In this unit, you install the Netcool/OMNibus Web GUI component, and validate the installation.

5. [Basic ObjectServer administration](#)

In this unit, you learn about the structure and functions of the ObjectServer. You learn about important tables and important columns. You also learn how to perform basic administration of the ObjectServer.

The exercises in this unit provide an overview of Netcool/OMNibus directory structure, an introduction to the Administrator utility, and a demonstration of ObjectServer modifications.

6. [Tools and automations](#)

This unit provides an introduction to menus, tools, and automations. This unit focuses on creating a basic tool, modifying an existing automation, and creating an automation.

The exercises in this unit continue the implementation of the solution to delete events. In this unit, you create a custom tool that seems to delete an event record. In reality, the tool modifies the Delete_Flag column, and the restriction filter prevents access to the event. This action gives the user the impression that the event is deleted. In addition to the tool, you create an ObjectServer trigger. It is the trigger that removes the event record from the ObjectServer. The trigger removes the event only if certain conditions are met.

7. Common integrations

This unit is focused on installing and configuring the components that are used for most common integrations. In this unit, you learn how to install and configure the Syslog probe, and the SNMP probe. These probes are common probes that you use in most installations. They provide the basis for integrating applications that produce Syslog messages and infrastructure components capable of generating SNMP traps.

The exercises in this unit guide you through the installation, and configuration of two of the most popular Netcool/OMNibus probes: the IBM® Tivoli® Netcool®/OMNibus SNMP Probe, and the IBM Tivoli Netcool/OMNibus Syslog Probe. In addition, you install the prebuilt collection of rules files for these probes. These files are the IBM Tivoli Netcool/OMNibus Knowledge Library.

8. ObjectServer high availability

In this unit, you learn about the high-availability configuration. You learn about the functional components, and the process to create the configuration.

The exercises in this unit guide you through the implementation of ObjectServer high availability.

9. Process control

Process control is the Netcool/OMNibus facility for automatically starting, stopping, and restarting components. You can also use this facility to run commands on a local or remote system and automate remedial actions. In this unit, you learn about the features, functions, and methods to configure process control. Process control and process activity refer to the same capability.

The exercises in this unit configure Netcool/OMNibus Process Activity to automatically start all core components: ObjectServers, probes, and gateway.

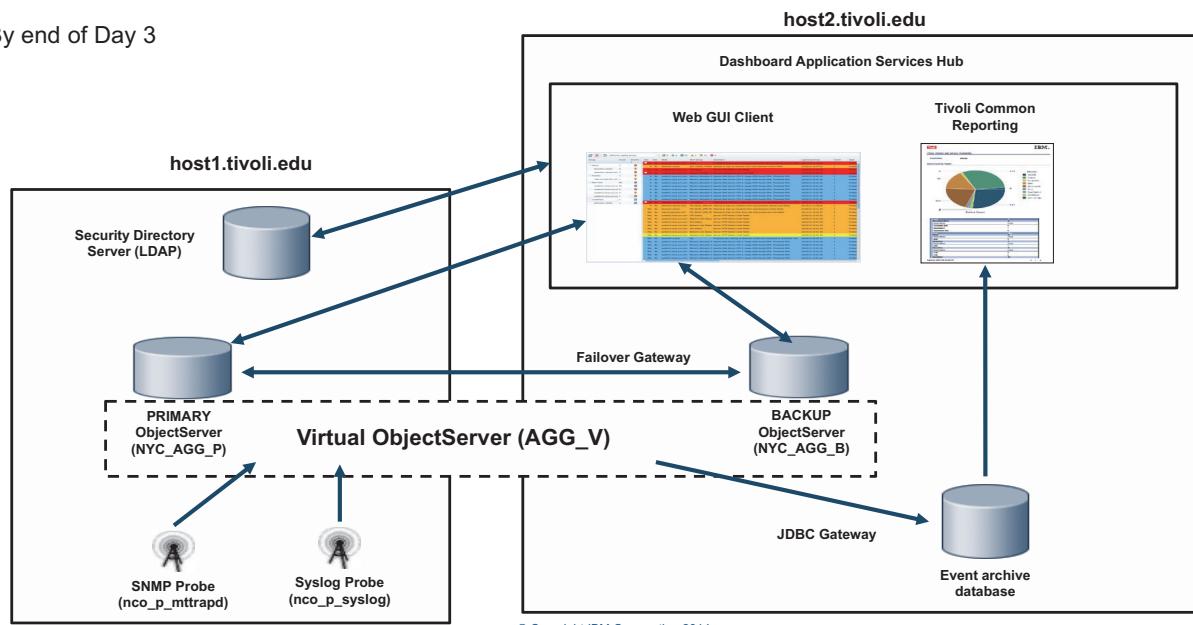
10. Event archiving

Netcool/OMNibus includes a package of prebuilt event reports. In this unit, you learn how to configure the Netcool/OMNibus solution to implement historical event retention and reporting.

The exercises in this unit teach you how to configure event archiving, and to install Tivoli Common Reporting.

Classroom architecture

By end of Day 3



3

The slide shown here illustrates the architecture that you create during the three days of this class.

[About this course](#)

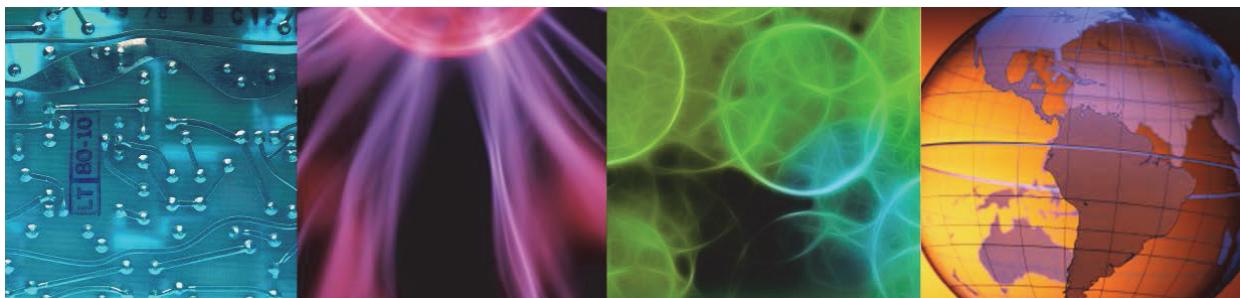
[Course agenda](#)



1 Basic Tivoli Netcool/OMNIbus components and concepts



1 Basic Tivoli Netcool/OMNIbus components and concepts



© Copyright IBM Corporation 2014
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit provides an introduction to the features, and functions of Netcool/OMNIbus.

References: SC27-6264-00 *Installation and Deployment Guide*

Objectives

In this unit, you learn to perform the following tasks:

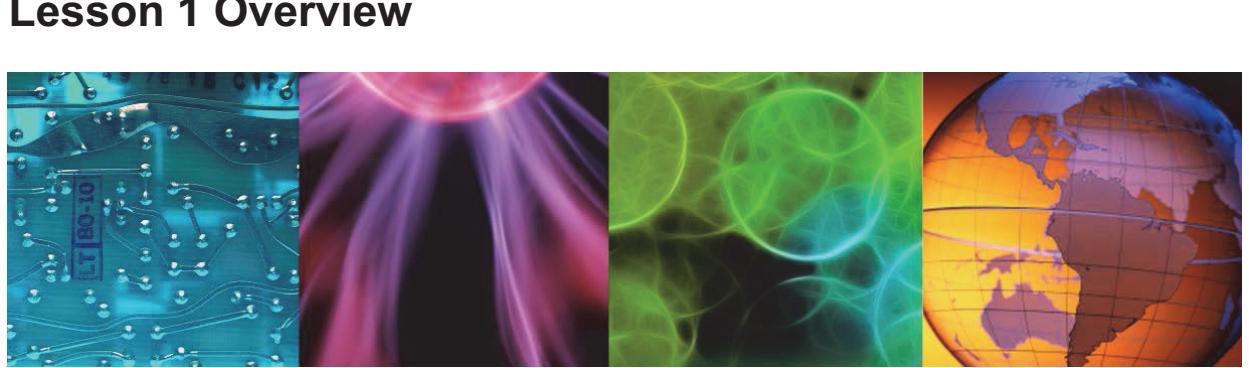
- Identify the basic software components of the Tivoli Netcool/OMNibus product suite
- Explain the logical relationships between different Tivoli Netcool/OMNibus software components

© Copyright IBM Corporation 2014

Objectives



Lesson 1 Overview



© Copyright IBM Corporation 2014

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

In this lesson, you learn the names, and functions of the Netcool®/OMNibus software components. After completing this lesson, you should be able to describe the Netcool/OMNibus software components.

Tivoli Netcool/OMNIbus overview

Tivoli Netcool/OMNIbus is the core alarm management database for any Tivoli Network Management solution

- A central data repository for network alarms
 - Alarm is the generic term for network faults, system faults, security faults, and performance indicators.
 - Sometimes the term alarm is used interchangeably with event. Unless indicated otherwise, in later units of this course, these two terms are synonymous
- Manager of managers
 - Tivoli Netcool/OMNIbus consolidates information from other management applications, system management systems, and element management systems
 - Other management applications include voice, data, Internet Protocol (IP), wireless, transmission, firewalls, and servers

© Copyright IBM Corporation 2014

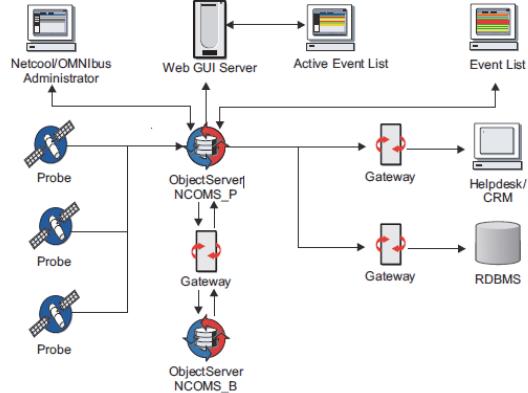
Tivoli Netcool/OMNIbus overview

Tivoli® Netcool/OMNIbus is a carrier-class service, and business assurance system. It collects and consolidates events and alarms from a wide variety of networking environments in real time. This equipment includes servers, mainframes, Windows systems, UNIX applications, circuit switches, voice switches, IP routers, SNMP devices, network management applications, and frameworks. By working with existing management systems and applications, Netcool/OMNIbus minimizes deployment time so that employees can use their existing network management skills.

Netcool/OMNIbus presents this consolidated information in a meaningful, intuitive, point-and-click format and provides information of interest to specific users through individually configurable filters and views. Through the Netcool/OMNIbus real-time monitoring, you can address problems before they cause disruptions in service. Netcool/OMNIbus provides continuity of business operations with architectures that are responsive, scalable, and highly available.

Tivoli Netcool/OMNibus software components

- ObjectServer
- Event Repository
- Desktop
 - Event List
 - Tools
- Web GUI and Dashboard Application Services Hub
 - Browser-based client
- Probes
- Gateways



© Copyright IBM Corporation 2014

Tivoli Netcool/OMNibus software components

The solution consolidates events from various management tools into a single database. Powerful event processing, and automation capabilities sort critical events from other noise, and identify event relationships across management domains. Examples of these domains include networks, storage, security, and systems.

The solution presents that information on a flexible, and powerful *single pane of glass* view of the status of your IT environment.

By understanding the relationships of problems across management domains, your operations team is able to immediately focus their efforts on solving problems rather than searching for causes.

This approach also helps to reduce operational workload while giving your operations staff a single tool rather than having to master each domain-specific management tool individually.

ObjectServer

The ObjectServer is the core fault management database of the entire Tivoli/Netcool suite

- Receives event data from probes and monitors
- Processes event data using tools and automations
- Transfers event data using gateways
- Shows event data to the user
 - Active Event List (AEL) in the Web GUI
 - Desktop Event List
- Uses Sybase OpenServer for communications

© Copyright IBM Corporation 2014

ObjectServer

The ObjectServer is the in-memory database server at the core of Netcool/OMNibus. Event information is forwarded to the ObjectServer from external programs such as probes and gateways. This information is stored and managed in database tables, and shown in the Web GUI event lists, or in the desktop event list.

A single device might generate the same error repeatedly until the problem is dealt with. The ObjectServer uses deduplication to ensure that event information that it generates from the same source is not duplicated in the event list. Repeated events are identified and stored as a single event to reduce the amount of data in the ObjectServer. The ObjectServer maintains a count, or tally, of the total number of recurrences of that event.

You can use automation to detect changes in the ObjectServer and generate automated responses to these changes. The automated response enables the ObjectServer to process alerts without requiring operator action.

Desktop

Is a thick client

Is software that is installed on user workstations

Provides controlled access to event data in the ObjectServer

Supports integration of desktop tools

Type of access is configured on a user-by-user basis

- Restrict access to specific event records
- Restrict access to certain desktop tools
- Provide read-only or read-write access to event data

© Copyright IBM Corporation 2014

Desktop

The **Desktop** is also referred to as the **Native Desktop**. The native desktop is the thick client that requires software to be installed on the user workstation.

Web GUI

Provides web-based access to event data

- Hosted inside Dashboard Application Services Hub

Provides controlled access to event data in the ObjectServer

Supports integration of desktop tools

Type of access is configured on a user-by-user basis

- Restrict access to specific event records
- Restrict access to certain desktop tools
- Provide read-only or read-write access to event data

Supports access to multiple ObjectServers

- Users can view events from multiple ObjectServers in a single event list

© Copyright IBM Corporation 2014

Web GUI

The Web GUI is a web-based application that processes events from one or more data sources, ObjectServers, and presents the event data to users in various graphical formats in a web browser. Certain data can also be shown on supported mobile devices. The Web GUI contains most features of the Netcool/OMNibus native desktop component. The Web GUI extends the event visualization and management capabilities of Netcool/OMNibus by being both highly configurable and remotely accessible through the Internet.

The Web GUI uses a client/server architecture. The Web GUI server runs inside Dashboard Application Services Hub, sometimes referred to as **DASH**. Clients connect to Dashboard Application Services Hub to access the Web GUI. You can configure the Web GUI for integrations with other Tivoli products.

You can deploy the Web GUI in a load-balanced environment with an IBM DB2® relational database. The license for Netcool/OMNibus contains an entitlement to download, install, and deploy the DB2 database in a load balancing environment.

Dashboard Application Services Hub

A web-based user interface for individual products and for integrating multiple products

A single, task-based navigation panel for multiple products

Users select actions based around the task that they want to complete, not by the product that supports that task

Single sign-on (SSO), consolidated user management, and a single point of access for different Tivoli applications

Aggregated views that span server instances, such as the Tivoli Netcool/OMNibus ObjectServer and Tivoli Enterprise Portal Server

Inter-view messaging between products to support contextual linkage between applications

The ability to create customized pages and administer access to content by user, role, or group

© Copyright IBM Corporation 2014

Dashboard Application Services Hub

Web-based products that are built on the Dashboard Application Services Hub framework share a common user interface where you can start applications and share information. Dashboard Application Services Hub helps the interaction and secure passing of data between Tivoli products through a common portal. You can link from one application to another and within the same dashboard view research different aspects of your managed enterprise.

Dashboard Application Services Hub is installed automatically with the first Tivoli product that uses the Dashboard Application Services Hub framework. Subsequent products might install updated versions of Dashboard Application Services Hub.

Probes

Lightweight software components that collect event information and send it to the ObjectServer

Enable the ObjectServer to be independent of the systems it monitors

- Large numbers of systems have available probes
- Generic and vendor-specific probes exist

Can modify and enrich event information

Are resilient

- Have store-and-forward functions
- Can be configured for automatic failover with other ObjectServers
- Can be configured in peer-to-peer mode to provide high availability

© Copyright IBM Corporation 2014

Probes

Netcool/OMNibus probes connect to an event source, detect and acquire event data, and forward the data to the ObjectServer as events. Probes use the logic that is specified in a rules file to manipulate the event elements before converting them into fields of an event in the ObjectServer alerts.status table.

Each probe is uniquely designed to acquire event data from a specific source. Probes can acquire data from any stable data source, including devices, databases, and log files. You can also configure probes to modify and add to the event data.

Gateways

Enable the exchange of events between ObjectServers and complementary vendor applications

Synchronize events between ObjectServers in a high-availability configuration

Forward event data to *help desk* applications to generate trouble tickets

Archive event data in databases

After the gateway is configured, the transfer of event data is transparent to operations

© Copyright IBM Corporation 2014

Gateways

Netcool/OMNibus Gateways enable the exchange of events between ObjectServers and complementary third-party applications, such as databases, and help desk or Customer Relationship Management (CRM) systems. You can use gateways to replicate events or to maintain a backup ObjectServer.

Application gateways enable you to integrate different business functions. For example, you can configure a gateway to send event information to a help desk system. You can also use a gateway to archive events to a database.

After a gateway is correctly installed and configured, the transfer of events is transparent to operators.

Online help system

IBM Eclipse Help System (IEHS) deploys Online Help for Tivoli Netcool/OMNibus

Online help is shown in a web browser

Two options for deployment

- Standalone: The IEHS application and online help files run on a local web server
- Information Center Mode: The IEHS application and online help files run on a remote web server

Online help is a separately installed feature

© Copyright IBM Corporation 2014

Online help system

The online help for Netcool/OMNibus is based on the IBM® Eclipse Help System (IEHS), which is a web application. Online help is shown in a web browser and is available in a stand-alone mode or information center mode. Stand-alone mode is the default.

In stand-alone mode, the IEHS application framework, and online help files that are run on a local web server. After installation, you can access the online help, usually without the need for any additional configuration. When you try to access online help, a web server automatically starts and runs locally until you manually shut it down.

In information center mode, the IEHS application framework and online help files are run on a remote web server to which users must connect to access online help. The use of a shared online help server relieves the load on local hosts or workstations.

Summary

You now should be able to perform the following tasks:

- Identify the basic software components of the Tivoli Netcool/OMNibus product suite
- Explain the logical relationships between different Tivoli Netcool/OMNibus software components

© Copyright IBM Corporation 2014

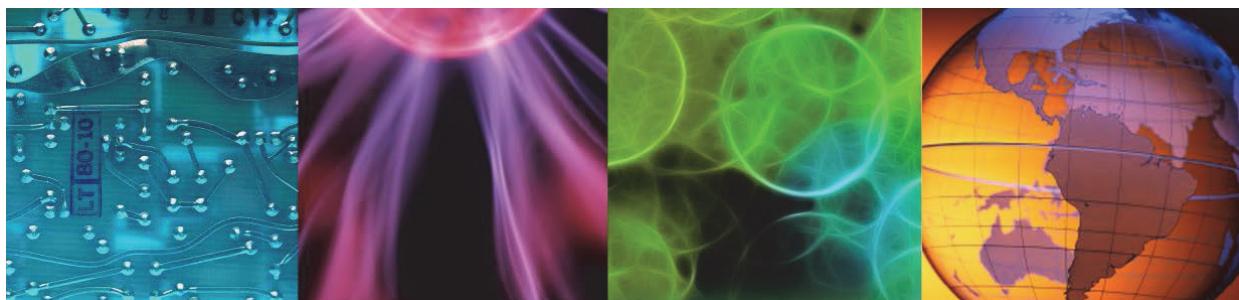
Summary



2 Installation prerequisites and planning



2 Installation prerequisites and planning



© Copyright IBM Corporation 2014

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

In this unit, you learn about the hardware, and software requirements for the installation of Netcool/OMNIbus.

References: SC27-6264-00 *Installation and Deployment Guide*

Objectives

In this unit, you learn to perform the following tasks:

- Describe the minimum disk requirements
- List the supported operating systems
- Describe other software requirements, for example, operating system patches
- Validate the host configuration with the prerequisite scanner
- Describe the issues related to various architecture configurations
- Describe the considerations for performing an upgrade

© Copyright IBM Corporation 2014

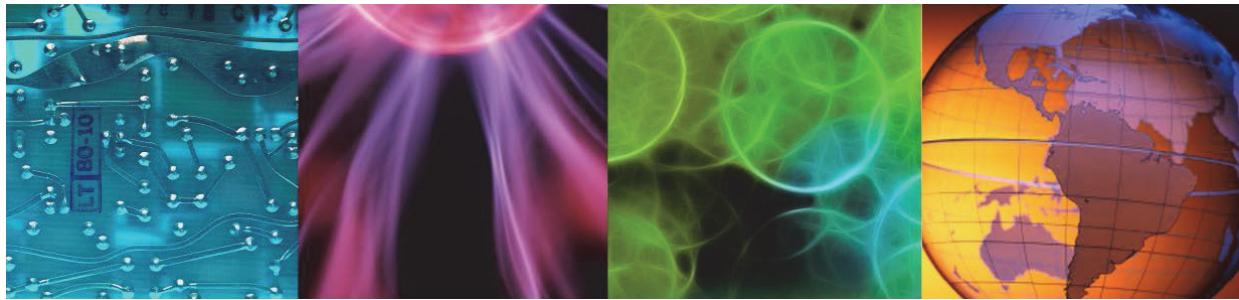
Objectives



Lesson 1 Preinstallation planning



Lesson 1 Preinstallation planning



© Copyright IBM Corporation 2014

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

In this lesson, you learn what the hardware, and software requirements are for installing Netcool®/OMNIbus. After completing this lesson, you should be able to perform the following tasks:

- Describe supported operating system
- Describe minimum disk space requirements
- Describe how to use the Prerequisite Scanner

Preinstallation planning

Read the *installation guide*

- Details every supported operating system
- Details additional software requirements
- Has step-by-step instructions for new installation
- Has step-by-step instructions for upgrades

Read the *release notes*

- Typically contains a list of *known issues* and workarounds, if any

Visit the IBM Support site

- Search for fix packs, and interim fixes

© Copyright IBM Corporation 2014

Preinstallation planning

IBM® continually releases new software, and it is always prudent to review the most current documentation for any recent changes.

System requirements

Visit the *Software Product Compatibility Reports* website

Search by product for:

- Supported operating systems
- Additional software requirements
- Supported browsers
- Hardware requirements

Software Product Compatibility Reports

Operating systems	Find out what is supported, compatible, available.
Related software	
Hypervisors	
Translations	
Detailed system requirements	
Hardware requirements	
End of service	

High-level reports about products related to *

Select a product. Get the list of the **operating systems** that it supports.

→ Create a report

Operating systems

Related software

Hypervisors

Translations



<http://pic.dhe.ibm.com/infocenter/prodguid/v1r0/clarity/index.jsp>

© Copyright IBM Corporation 2014

System requirements

The installation guide and release notes documents describe the supported operating systems. A more convenient way to verify the operating system is to use the Software Product Compatibility website, found here:

<http://pic.dhe.ibm.com/infocenter/prodguid/v1r0/clarity/index.jsp>

You can use this website to search for the list of operating systems by product name. The product information found here is not limited to Netcool/OMNibus. You can use it for all Tivoli® products.

Supported operating systems

← IBM Support Portal

Software Product Compatibility Reports

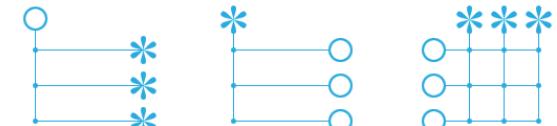
Operating systems
Prerequisites
Hypervisors
Translations
Detailed system requirements
Hardware requirements
End of service

Related links
• Feedback survey for this tool

Software Product Compatibility Reports

Find out what is supported, compatible, available.

High-level reports about products related to *



Operating systems

Select a product. Get the list of the operating systems that it supports.

Create a report **Sample report**

Prerequisites

Select an operating system. Get the list of products that run on it.

Create a report **Sample report**

Hypervisors

Select products. Select operating systems. Explore supported combinations.

Create a report **Sample report**

Translations

In-depth reports

Detailed system requirements:
Supported operating systems, prerequisites, hypervisors and hardware requirements.

Create a report **Sample report**

Hardware requirements:
Find out the required hardware for a product you are interested in.

Create a report **Sample report**

End of service:
Find out the end of service dates for the products you have in mind.

Create a report **Sample report**

© Copyright IBM Corporation 2014

Supported operating systems

You start by selecting the option for Operating systems. Then click **Create a report**.

Select product and version

The screenshot shows the 'Operating systems for a specific product' section of the IBM Support Portal. On the left, a sidebar lists categories like Operating systems, Related software, Hypervisors, Translations, Detailed system requirements, Hardware requirements, and End of service. The main area has a breadcrumb trail: IBM Support Portal > Software Product Compatibility Reports > Operating System Reports > Operating systems for a specific product. A red box highlights the 'Full or partial product name' input field where 'Netcool/OMNibus' is typed. Below it, a search results list shows 'Tivoli Netcool/OMNibus'. A red arrow points from the 'Submit' button at the bottom to the 'Submit' button in the main form. Other visible fields include 'Version' set to '8.1.0' and a checked 'Show fix packs' checkbox.

Search for a product by name

Select product

Select product version

Generate the report

Select product and version

When you select the option to search by product, you can enter the complete product name or just a portion of the name. The search function returns the list of products that match the search criteria. Next, select the product version, and click **Submit**.

View the results

Tivoli Netcool/OMNibus 8.1.0

Report filters

Available Reports: 8.1.0 initial version

Legend: Component Support: Full, Partial, None

Utilities: Regenerate Anytime, Print, Download PDF, Provide feedback

Notes: Data as of 2014-05-27 02:44:30 EDT, Disclaimers, New Design!, View new features

AIX

Operating System	Operating System Minimum	Hardware	Bitness	Product Minimum	Components	Notes	Details
AIX 5.1	Base	POWER System - Big Endian	64-Bit/32	8.1.0	Full, Partial	No	View
AIX 7.1	Base	POWER System - Big Endian	64-Bit/32	8.1.0	Full, Partial	No	View

Linux

Operating System	Operating System Minimum	Hardware	Bitness	Product Minimum	Components	Notes	Details
Red Hat Enterprise Linux (RHEL) 5 Advanced Platform	Update 7	i36-64	64-Bit/32	8.1.0	Full, Partial	No	View
Red Hat Enterprise Linux (RHEL) 5 Desktop edition	Update 7	i36-64	64-Bit/32	8.1.0	Full, Partial	No	View
Red Hat Enterprise Linux (RHEL) 5 Server	Various	i36-64	64-Bit/32	8.1.0	Full, Partial	No	View
Red Hat Enterprise Linux (RHEL) Client 6	Base	i36-64	64-Bit/32	8.1.0	Full, Partial	No	View
Red Hat Enterprise Linux (RHEL)	Base	i36-64	64-Bit/32	8.1.0	Full, Partial	No	View

© Copyright IBM Corporation 2014

View the results

The report lists all supported operating systems for the selected product. Pay close attention to the **Legend** icon. This icon indicates whether all components within Netcool/OMNibus are supported on the respective operating system or only selected components. The components are categorized as Desktop, web server, and ObjectServer. There are some versions of operating system, for example, that support the ObjectServer but not the Desktop.

This report does not include support for probes, and gateways. Operating system support for a probe or gateway is going to be directly related to the specific probe or gateway. Probes and gateways are often vendor-specific items. They provide integration with a specific data source. The probe or gateway might be on the same server as the vendor component. In that case, if the vendor application is supported only a particular server, the probe or gateway must also be on that server. In other cases, the probe is technology-specific, for example the Syslog Probe. This probe is not supported on Windows servers because it reads UNIX Syslog messages.

You should always refer to the *release notes* for the specific probe or gateway to verify which operating systems it supports.

Hardware requirements

The screenshot shows the 'Hardware requirements for a specific product' search page. The left sidebar has a 'Hardware requirements' link highlighted with a red box. The main search area has a 'Full or partial product name:' input field containing 'Netcool/OMNIBus'. Below it is a 'Search results:' section with a single result 'Tivoli Netcool/OMNIBus'. Further down are fields for 'Versions:' (set to '8.1.0') and 'Operating system families:' (with checkboxes for 'AIX' and 'Linux' both checked). At the bottom is a 'Submit' button.

Hardware requirements

Hardware requirements are also documented in the installation guide. The installation guide provides guidance that is based on the approximate size of the deployment. The installation guide provides guidance that is based on a small, medium, or large deployment. You can also use the Software Product Compatibility website to determine minimum hardware requirements.

You start by selecting **Hardware requirements**. You search for and select a product name. Next, select the product version, and click **Submit**.

View the result

Tivoli Netcool/OMNibus 8.1.0

Detailed hardware requirements

AIX	Linux	Notes	Other formats
Hardware	Components	Requirement	Applicable OS
Disk Space	Desktop ObjectServer Clients	Minimum 10 GB. High-volume installations (millions of raw events per day) might require individual assessment. For more information, see the OMNibus Sizing Guide .	All supported AIX OS
Memory	Desktop ObjectServer Clients	Minimum 4 GB RAM. Ensure adequate swap space for the selected operating system. High-volume installations (millions of raw events per day) might require individual assessment. For more information, see the OMNibus Sizing Guide .	All supported AIX OS
Server	ObjectServer Web GUI Server		

Linux	Notes	Other formats	
Hardware	Components	Requirement	Applicable OS
Disk Space	Desktop ObjectServer Clients	Minimum 10 GB. High-volume installations (millions of raw events per day) might require individual assessment. For more information, see the OMNibus Sizing Guide .	All supported Linux OS
Server			© Copyright IBM Corporation 2014

View the result

The results contain the minimum values for disk space and physical memory. A separate OMNibus Sizing Guide document can be downloaded from here:

<https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Netcool%20OMNibus/page/OMNibus%20Sizing%20Guide>

Additional software requirements

The screenshot shows a web-based search interface for related software products. The left sidebar has a navigation menu with several options: IBM Support Portal, Software Product Compatibility Reports, Operating systems, Related software (which is highlighted with a red box), Products that use a specific related software, Matrix between specific products and related software, Hypervisors, Translations, Detailed system requirements, Hardware requirements, and End of service.

The main content area is titled "Related software for a specific product". It includes a search bar for "Full or partial product name" containing "Netcool/OMNibus", a search results list showing "Tivoli Netcool/OMNibus", and a "Version:" dropdown set to "8.1.0" with a "Show fix packs" checkbox. Below these are "Filters" for "Operating system platforms", "Product components", and "Capabilities". At the bottom is a "Submit" button, which is also highlighted with a red arrow pointing to it from the sidebar.

Additional software requirements

Additional software includes things like databases, LDAP servers, web browser, and operating system patches. You start by selecting **Related software**. You search for and select a product name. Next, select the product version, and click **Submit**.

View results

Product
Tivoli Netcool/OMNibus 8.1.0

Report filters

Available Reports
8.1.0 Initial version

Legend
Component Support
Full
Partial
None

Utilities
Regenerate Anytime
Print
Download PDF
Provide feedback

Notes
Data as of 2014-08-27
02:43:30 EDT
Disclaimers
New Design!
View new features

Prerequisites Supported Software 

Show notes | Hide notes

Databases LDAP Servers Provisioning Resource Management Service Management
Event Management Problem Determination Tools Reporting and Analysis Security Management Web Browsers

Databases

Supported Software	Version	Prerequisite Minimum	Product Minimum	Components	Operating System Restrictions?	Notes	Details
DB2 Advanced Enterprise Server Edition	10.1	10.1	8.1.0	Full	No	No	
	10.5	10.5	8.1.0	Full	No	No	
DB2 Enterprise Server Edition	10.1	10.1	8.1.0	Full	No	No	
	10.5	10.5	8.1.0	Full	No	No	
DB2 Workgroup Server Edition	10.1	10.1	8.1.0	Full	No	No	

© Copyright IBM Corporation 2014

[View results](#)

Mobile device support

Some Web GUI functions can be shown on mobile devices. These functions are as follows:

- Mobile event list
 - The pages of the mobile event list, the landing page, event dashboard, event list and event details, are supported on Android V2.3 and later, and iOS 5.0 and later for iPhones.
- Mobile gauges

To show a Gauges page on a mobile device, the device must:

- Be a smartphone
- JavaScript and AJAX must be enabled on the browser
- The screen must have a minimum resolution of 320 x 240 pixels

The Gauges page has been demonstrated on:

- Smartphones that run on the Blackberry 4.6+ and iOS 4.0+ operating systems.

© Copyright IBM Corporation 2014

Mobile device support

Some Web GUI functions can be shown on mobile devices. These functions are as follows:

- Mobile event list

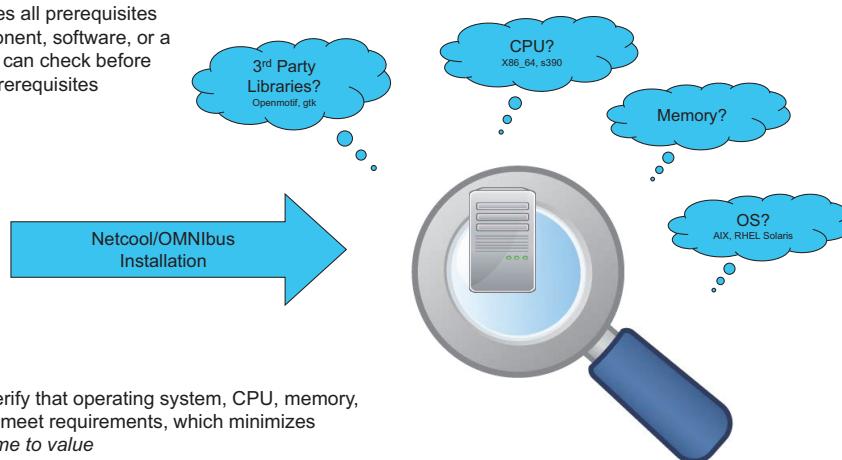
The pages of the mobile event list are the landing page, event dashboard, event list, and event details are supported on Android V2.3 and later, and iOS 5.0 and later for iPhones.

- Mobile gauges

To show a gauge page on a mobile device, the device must be a smartphone. You must enable JavaScript and Ajax on the browser, and the screen must have a minimum resolution of 320 x 240 pixels.

Prerequisite scanner

IBM Prerequisite Scanner (PRS) verifies all prerequisites before the actual deployment of component, software, or a solution takes place so that customers can check before deployment and identify any missing prerequisites



© Copyright IBM Corporation 2014

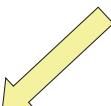
Prerequisite scanner

IBM Prerequisite Scanner is a stand-alone prerequisite checking tool that analyzes system environments before the installation or upgrade of a Tivoli product or IBM solution. The IBM Prerequisite Scanner verifies that the required hardware and software for Netcool/OMNibus, including the Web GUI component, are present on the host computer.

Prerequisite scanner: Command line

```
./prereq_checker.sh "Product_Code [Product_Version] [,Product_CodeN [Product_VerN]]..."  
[PATH=<install path>]  
[detail]
```

PRS Product Codes match
Netcool/OMNibus installable features



PRS Code	Installer feature
NOC	All Netcool/OMNibus components
NOA	Netcool/OMNibus Administration GUI
NOD	Netcool/OMNibus Desktop components
NOP	Netcool/OMNibus Probe components (and Gateways)
NOS	Netcool/OMNibus Server components
NPA	Netcool/OMNibus Process Agent components
NOW	Netcool/OMNibus Web GUI

```
%> ./prereq_checker.sh "NOC 08010000" PATH=/opt/IBM/tivoli/netcool
```

© Copyright IBM Corporation 2014

Prerequisite scanner: Command line

The prerequisite scanner is distributed as a compressed directory. To install the scanner, you expand the file to create the directory structure.

The scanner evaluates a target system that is based on the Netcool/OMNibus components that you want to install. The scanner provides seven command-line options that you use to select the combination of components that you intend for the target system.

Prerequisite scanner: Example of a pass

```
[root@host1 prs]# ./prereq_checker.sh "NOS 08010000" detail
IBM Prerequisite Scanner
  Version: 1.2.0.13
  Build: 20140825
  OS name: Linux
  User name: root

  Machine Information
  Machine name: host1.tivoli.edu
  Serial number: VMware-56 4d 01 e8 e3 e6 fc b6-42 98 fe c4 6b 35 28 25

  Scenario: Prerequisite Scan

  NOS - Tivoli Netcool/OMNIbus Server Components [version 08010000]:
  Property          Result    Found           Expected
  =====            ======   =====
  OS Version        PASS     Red Hat Enterprise Linux Server rel... AIX V6.1
                                         AIX V7.1
                                         Solaris V10 (SPARC)
                                         Solaris V11 (SPARC)
                                         Redhat Enterprise Linux Server 5.*
                                         Redhat Enterprise Linux Server 6.*
                                         SuSE Linux Enterprise Server 11
                                         x86_64,s390,s390x
                                         4GB
                                         415MB
                                         [dir:root=/opt;non_root=USERHOME]222MB
                                         [dir:root=/var;non_root=USERHOME]3MB
                                         audit-langs-2.2-2.el6.x86_64
                                         audit-libc-2.12-1.107.el6.4.5.x86_64+
                                         glibc-2.12-1.7.el6.x86_64+
                                         libgcc-4.4.7-3.el6.x86_64
                                         libstdc++-4.4.7-3.el6.x86_64
                                         libstdc++-4.4.4-13.el6.x86_64+
                                         nss-softokn-freebl-3.12.9-11.el6.x8...
                                         pam-1.1.1-13.el6.x86_64+
                                         pam-1.1.1-4.el6.x86_64+
                                         zlib-1.2.3-29.el6.x86_64
                                         zlib-1.2.3-25.el6.x86_64

  Aggregated Properties for Scanned Products:
  Property          Result    Found           Expected
  =====            ======   =====
  /                PASS     29696.00MB
                                         640MB
                                         4.00GB
  Memory           PASS     5.42GB

Overall result: PASS (NOS 08010000: PASS)
Detailed results are also available in /tmp/prs/result.txt
```

© Copyright IBM Corporation 2014

Prerequisite scanner: Example of a pass

This example shows the results from the scanner when the system meets all of the requirements for the selected components.

Prerequisite scanner: Example of a failure

```
[root@host1 prs]$ ./prereq_checker.sh "NOC 08010000" detail
IBM Prerequisite Scanner
  Version: 1.2.0.13
  Build : 20140825
  OS name: Linux
  User name: root
.

.

NOC - Tivoli Netcool/OMNIbus Server, Desktop and Probe Components [version 08010000]:
Property          Result    Found      Expected
=====          =====    =====      =====
OS Version        PASS     Red Hat Enterprise Linux Server rel... AIX V6.1
                                         AIX V7.1
                                         Solaris V10 (SPARC)
                                         Solaris V11 (SPARC)
                                         RedHat Enterprise Linux Server 5.*
                                         RedHat Enterprise Linux Server 6.*
                                         SuSE Linux Enterprise Server 11
                                         x86_64
                                         4GB
                                         606MB
                                         [dir:root=/opt;non_root=USERHOME]302MB
                                         [dir:root=/var;non_root=USERHOME]2MB

CpuArchitecture    PASS     x86_64
Memory             PASS     5.48GB
Disk               PASS     29696.00MB
os.space.imshared  PASS     29696MB
os.space.imdata    PASS     29696MB
.

.

os.package.openmotif.x86_64  PASS     openmotif-2.3.3-6.1.el6_4.x86_64  openmotif-2.3.3-1.el6.x86_64+
os.package.pam.x86_64       PASS     pam-1.1.1-13.el6.x86_64+           pam-1.1.1-4.el6.x86_64+
os.package.zlib.x86_64      PASS     zlib-1.2.3-29.el6.x86_64+         zlib-1.2.3-25.el6.x86_64+
os.package.gtk2.x86_64      PASS     gtk2-2.18.9-4.el6.x86_64+         gtk2-2.18.9-4.el6.x86_64+
os.package.libjpeg.x86_64   FAIL     Unavailable                         libjpeg-6b-46.el6.x86_64+ [highlighted]

Aggregated Properties for Scanned Products:
Property          Result    Found      Expected
=====          =====    =====      =====
/                  PASS     29696.00MB  910MB
Memory            PASS     5.48GB    4.00GB

Overall result: FAIL (NOC 08010000: FAIL)
```

Detailed results are also available in /tmp/prs/result.txt

© Copyright IBM Corporation 2014

Prerequisite scanner: Example of a failure

This example shows the results from the scanner when the system does not meet all of the requirements for the selected components. The output indicates that a required library is missing.

Prerequisite scanner: More information

IBM officially supports the prerequisite scanner

It is **not** included with Netcool/OMNibus; download the PRS from [IBM Fix Central](#)

Netcool/OMNibus prerequisite configuration files ship with the PRS

Nothing is required from Netcool/OMNibus installation to run prerequisite checks

Supports all Netcool/OMNibus components

© Copyright IBM Corporation 2014

Prerequisite scanner: More information

The prerequisite scanner is not supplied with Netcool/OMNibus. You can locate it on the IBM Fix Central website that is found at the following URL:

<http://www-933.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm~Tivoli&product=ibm/Tivoli/Prerequisite+Scanner&release>All&platform>All&function=all>

See the following technote for information about using the prerequisite scanner with Netcool/OMNibus:

<http://www-01.ibm.com/support/docview.wss?rs=3120&uid=swg21472859>



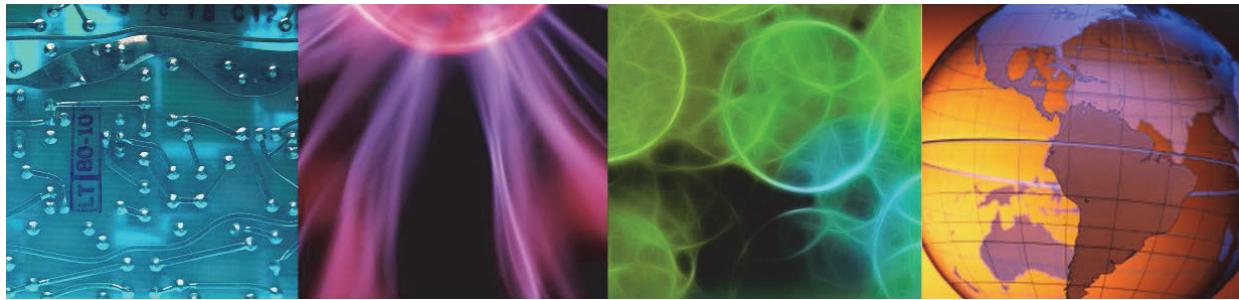
Important: Although as of this writing the technote states that the Web GUI component is not supported, a subsequent update to the scanner added Web GUI support.



Lesson 2 Other considerations



Lesson 2 Other considerations



© Copyright IBM Corporation 2014

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

In this lesson, you learn about some of the other considerations that relate to the installation of Netcool/OMNibus. After completing this lesson, you should be able to perform the following tasks:

- Describe some of the architectural considerations
- Describe considerations that are related to an upgrade
- Describe feature compatibility between versions

Architecture considerations

Component placement

- High availability
- Probe placement
- Gateway placement

Security considerations

- Is SSL encryption required
- Firewalls
- Source of user authentication: ObjectServer, LDAP

Integration with other products

- Ticketing systems
- Other Tivoli products

© Copyright IBM Corporation 2014

Architecture considerations

As part of installation planning, you must consider several issues that relate to the intended architecture.

Component placement

Netcool/OMNIbus is often deployed in a distributed configuration. Considerations for where you should deploy these components are listed in the following sections.

High availability

The primary intent of a *high availability* (HA) configuration is to prevent operational disruption because of component failures. The Netcool/OMNIbus HA is not dependent upon a specific hardware configuration, such as clustered servers. The only requirement is that TCP/IP connectivity exists between components. You can place individual components in different physical locations.

Probes

The probe communicates with the ObjectServer over TCP/IP, which ensures delivery of event data to the ObjectServer. The probe can rely upon a mechanism to collect data from the respective source that does not ensure delivery, for example SNMP traps over UDP. You can place the probes in the infrastructure close to the source.

Gateways

All gateways can connect remotely to an ObjectServer. All gateways support the ObjectServer high availability (HA) configuration where the gateway can fail back and forth between a primary and backup ObjectServer. You can place the gateway with the destination application. For example, the gateway that SmartCloud Control Desk uses can be run on that system. You can also configure it to connect remotely to the ObjectServer or ObjectServer pair in an HA configuration.

Security considerations

The following sections list some of the common considerations with regards to security.

SSL encryption

Netcool/OMNibus can communicate with client components, such as desktops and probes, over SSL-encrypted links. SSL encryption determines how the communication is configured between the ObjectServer and client component.

Firewalls

Most infrastructures contain one or more firewalls that restrict traffic flow. You must be aware of where Netcool/OMNibus components are placed relative to any firewalls in your infrastructure. You can configure the ports that various Netcool/OMNibus components use. You can customize the Netcool/OMNibus deployment to conform to specific firewall port restrictions in your environment.

User authentication

All Netcool/OMNibus users require a user name and password. The current version of Netcool/OMNibus supports LDAP as the source for user authentication.

Integration with other products

Netcool/OMNibus supports integration with products that are described in the following sections.

Ticketing systems

The integration to ticketing systems is through a Netcool/OMNibus gateway. This integration requires a user ID and password for the ticketing system. The gateway connects to the ticketing application as a valid user. Also, depending upon the ticketing system, modifications to the ticketing system might be necessary. These modifications are generally limited to standard features that are available within the product. As an example, you can configure a response option within the ticketing system. The response option generates an automated response to the gateway user when changes are made to any ticket that the gateway creates.

Other Tivoli products

Netcool/OMNibus has integrations with many Tivoli products. These integrations are typically based on a specific version or release of the Tivoli product. Review the current documentation to verify whether there are any software requirements.

Upgrade considerations

Upgrades to Tivoli Netcool/OMNibus V8.1 are supported from V7.4, V7.3.1, V7.3, V7.2.1, and V7.2.

IBM Installation Manager is the recommended option for upgrading Netcool/OMNibus.

Quick reference to upgrading:

1. Review compatibility issues with previous versions of the product
2. Check system prerequisites and obtain installation packages
3. If you are using or intend to use FIPS 140-2 encryption, refer to the FIPS 140-2 configuration checklist
4. Back up the existing system and use IBM Installation Manager to upgrade the installation
This step provides the option to automatically migrate the existing data to the new configuration
5. Review the list of files migrated and perform any manual configuration required
6. On UNIX, set environment variables, if necessary
7. Upgrade the ObjectServer schema to the V8.1 schema
8. If you upgraded from an earlier version that used SSL, migrate your certificate files and private keys

© Copyright IBM Corporation 2014

Upgrade considerations

The upgrade process consists of steps that are automated and steps that are manual. The installed version of Netcool/OMNibus determines the steps that an upgrade requires.

IBM Installation Manager performs the automated steps. A copy of the previous installation is saved. New Netcool/OMNibus components are then installed, and specific configuration files are migrated. The exact list of files is in the *installation guide*. In general, these files are limited to items like property files, desktop.elc and.elv files, and other configuration files.

Post-upgrade steps

1. Upgrade ObjectServer schema

- The automated upgrade does not alter the ObjectServer schema
- There are scripts that you must run manually to upgrade the ObjectServer
- Depending upon the previous version, you might have to run multiple scripts

2. Install probes and gateways

- Probes and gateways are installed differently starting in Tivoli Netcool/OMNibus V8.1
- You must reinstall probes and gateways used in previous versions
- The upgrade process saves the previous probe rules and properties in a separate directory
- You must manually move these files to the new probe directory

© Copyright IBM Corporation 2014

Post-upgrade steps

One of the required manual processes is to upgrade the ObjectServer configuration. Scripts are provided with the installation that you use to update the ObjectServer. These scripts must be run manually to apply the ObjectServer modifications.

Based on the existing version of Netcool/OMNibus, you might have to run multiple scripts. For example, if the previous Netcool/OMNibus version is 7.3, you run a script to upgrade the ObjectServer to Netcool/OMNibus 7.3.1 compatibility. Then you run a second script to upgrade from Netcool/OMNibus V7.3.1 to Netcool/OMNibus V7.4. And you run one more script to upgrade from Netcool/OMNibus 7.4 to 8.1. You run these multiple scripts because changes are introduced to the ObjectServer during Netcool/OMNibus upgrades. The subsequent changes are based on the previous changes. The only way to ensure that the changes are introduced correctly is to use this step-by-step process.

Another consideration of the upgrade process is related to probes and gateways. Most probes and gateways are installed separately. When upgrading Netcool/OMNibus core, individual probes and gateways are not upgraded. They must be upgraded separately with IBM Installation Manager.

Student exercises



© Copyright IBM Corporation 2014

Student exercises

Perform the exercises for this unit.

Summary

You now should be able to perform the following tasks:

- Describe the minimum disk requirements
- List the supported operating systems
- Describe other software requirements, for example, Operating System patches
- Be able to use the prerequisite scanner to validate the host configuration
- Describe the issues related to various Architecture configurations
- Describe the considerations for performing an upgrade

© Copyright IBM Corporation 2014

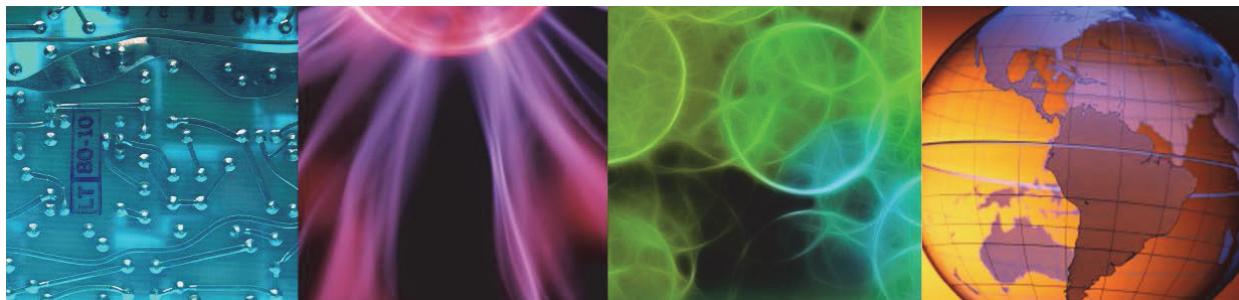
Summary



3 Netcool/OMNIbus core installation



3 Netcool/OMNIbus core installation



© Copyright IBM Corporation 2014

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

This unit teaches the installation of Netcool/OMNIbus core, the creation of an ObjectServer, and the steps to verify basic function.

References: SC27-6264-00 *Installation and Deployment Guide*

Objectives

In this unit, you learn to perform the following tasks:

- Install Tivoli Netcool/OMNibus V8.1 core product
- Update communications file
- Create and start a new ObjectServer
- Start a probe
- Use the desktop to view ObjectServer events

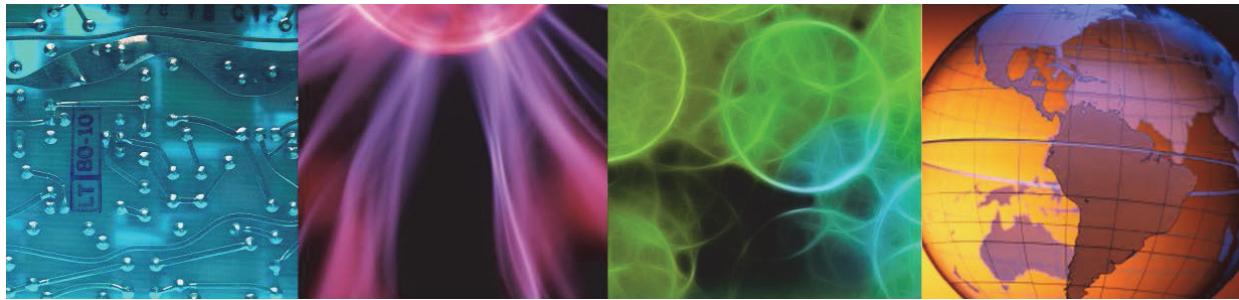
© Copyright IBM Corporation 2014

Objectives

Lesson 1 Preinstallation requirements



Lesson 1 Preinstallation requirements



© Copyright IBM Corporation 2014

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

In this lesson, you learn the steps that are required to prepare a target system before installing Netcool®/OMNibus. After completing this lesson, you should be able to perform the following tasks:

- Describe the user considerations that are related to the IBM® Installation Manager
- Describe the required environment variable settings

Installation overview

1. Design system
2. Install components on appropriate hardware
3. Configure IDUC communications address book
4. Build the ObjectServer
5. Start ObjectServer and desktop
6. Install, configure, and start probes
7. Install, configure, and start gateways
8. Install and configure other components
9. Verify that all components work together
10. Configure and start process control
11. Configure process control to start at server boot

© Copyright IBM Corporation 2014

Installation overview

The basic steps for deploying a Netcool/OMNibus solution are described in the following sections.

Design the system

This step includes the topics that are presented in the previous unit, emphasizing where you place the components. In your exercise, you create a distributed system and install various components on two servers.

Install the components

This step installs the software components.

Configure IDUC communications

You configure the address book to enable communication between components.

Build the ObjectServer

You use a utility to create the data structures that the ObjectServer uses.

Start the ObjectServer and desktop

After you create the ObjectServer, you start it. You then use the desktop to connect to the new ObjectServer and validate basic access.

Configure and start probes

After the ObjectServer is running, it is ready to receive new event data from probes. You configure a special probe that generates artificial events. This probe is convenient for testing and training. It provides a ready supply of sample event data.

Configure and start gateways

Most gateways must be installed separately. However, the ObjectServer gateways are installed with the core software. In a subsequent unit, you configure the bidirectional gateway and use it in a high-availability configuration.

Configure other components

In a production deployment, there might be components, such as Tivoli®i Netcool/Impact or Tivoli Business Service Manager, that integrate with Tivoli Netcool/OMNibus.

Verify that all components work together

After you configure the components, you verify that they work together.

Configure and start process control

Process control is the component that automatically starts and shuts down the Netcool/OMNibus solution. This step is typically the last one in a deployment. You want to verify that everything is configured correctly before enabling the automated activation.

IBM Installation Manager

IBM Software Group standard interface

Used to install all Tivoli Netcool/OMNibus components, and other IBM products

IBM Installation Manager install files are bundled with Tivoli Netcool/OMNibus

The files are also available separately

Standard interface across all operating systems

Installation options:

- Graphical (wizard)
- Textual (console)
- Silent

Three user modes:

- Administrator
- Nonadministrator
- Group

© Copyright IBM Corporation 2014

IBM Installation Manager

Many Tivoli software products now use a common installer called IBM Installation Manager. A common installer has two primary benefits:

- You use the same installer for all supported operating systems.
- The installer has a similar interface and operation across different product installations.

IBM Installation Manager maintains a database that contains a list of all installed products and features. The database can be queried to determine which specific components are currently installed.

The installer offers three modes of operation: graphical, textual, and silent:

- Graphical

The graphical mode is referred to as the wizard. This mode provides the step-by-step questions and answers that the installation requires.

- Textual

Often software is installed on a remote server over a telnet session. The textual or console mode provides the same basic features as the graphical mode but does not require console access to the server.

- Silent

The silent mode uses a response file to answer the questions during the installation and can run with no manual intervention. You can generate the response file from a previous installation in the graphical or textual mode.

Installation Manager user modes

Administrator mode:

- An administrator or root user can install one instance of Installation Manager
- Information about all of the products that are managed by this instance is stored in a single data directory
- Administrator mode requires root privileges on UNIX
- Root users can install multiple instances of Tivoli Netcool/OMNibus

Nonadministrator mode:

- Each user account can install one instance of Installation Manager
- Information about all of the products that are managed by this instance is stored in a single data directory
- Each user account can install multiple instances of Tivoli Netcool/OMNibus

Group mode:

- Users can install any number of instances of Installation Manager
- Each instance requires a different data directory
- All members of a group can use each instance
- Installation Manager automatically sets file permissions so that all members of the group can update the installation

© Copyright IBM Corporation 2014

Installation Manager user modes

IBM Installation Manager provides three user modes: Administrator, Nonadministrator' and Group. The group mode is used for installing Netcool/OMNibus components.

Installation options

Use Netcool/OMNibus installation script to install IBM Installation Manager and Netcool/OMNibus

Install IBM Installation Manager separately and then:

- Install Netcool/OMNibus using a local copy of the installation file
- or
- Configure IBM Installation Manager to download the Netcool/OMNibus installation file from Passport Advantage and install

© Copyright IBM Corporation 2014

Installation options

You have a choice when you install Netcool/OMNibus on a system where IBM Installation Manager is not currently installed. You can run a utility that installs IBM Installation Manager and then runs IBM Installation Manager to install Netcool/OMNibus. Or you can install IBM Installation Manager separately and then run IBM Installation Manager to install Netcool/OMNibus. The first option requires that you download a copy of the Netcool/OMNibus installation file from the Passport Advantage® website. This installation file contains IBM Installation Manager and Netcool/OMNibus. The second option requires that the user downloads only the installation file for IBM Installation Manager. After IBM Installation Manager is installed, you can configure it to retrieve the Netcool/OMNibus installation file during the installation.

Preinstallation steps

1. Create system users and groups

UNIX/Linux

- User for installation (if non-root)
- User for *process activity* authentication (optional)
- ncoadmin group

2. Environment variables (UNIX and Linux)

- \$NCHOME: Installation target directory
- \$OMNIHOME: \$NCHOME/omnibus (not required, but useful)
- \$PATH: Add \$OMNIHOME:\$OMNIHOME/probes (not required but useful)
- Log file rotation (optional) example:
\$NDE_LOGFILE_MAXSIZE: 102400
\$NDE_LOGFILE_ROTATION_FORMAT: %Y%m%d-%H%M
\$NDE_LOGFILE_ROTATION_TIME: 0000

3. File system permissions

Make sure the installation user has write access to \$NCHOME

© Copyright IBM Corporation 2014

Preinstallation steps

The student exercises in this unit assume that you are installing Netcool/OMNibus on a clean Linux system. You start the installation process by performing basic Linux administration tasks. Next, you configure some environment variables.

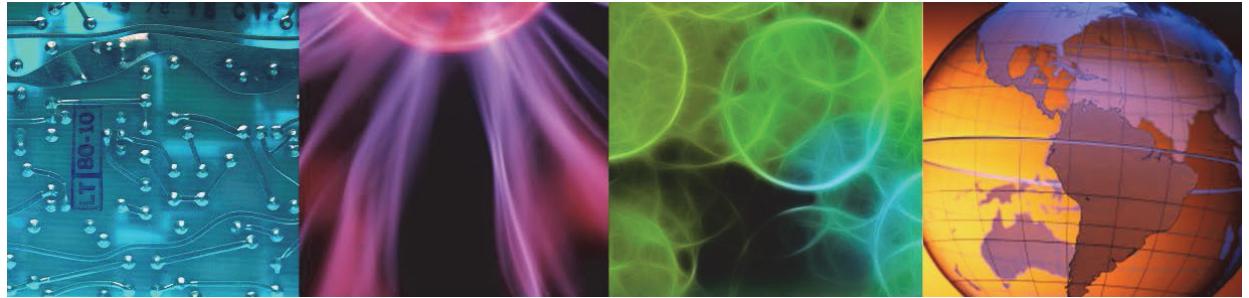
The only required environment variable is \$NCHOME, the location where Netcool/OMNibus is installed. However, you also configure more environment variables that provide some conveniences. You also set the appropriate file system permissions for the installation user that is selected.

A minimum number of steps must be done as the **root** user. Root access is no longer required after these steps are complete. You switch to the non-root user ID selected for installation and then complete the remaining exercises as that user.



Lesson 2 Core installation

Lesson 2 Core installation



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to install the Netcool/OMNIbus core components.

Netcool/OMNibus installer

Bundled with the installation files

Installs IBM Installation Manager

Starts IBM Installation Manager to complete the installation

Installation steps:

1. Expand the core installation file as follows
`unzip OMNIBus-v8.1-Core.linux64.zip`
2. Start the installation utility
`./install_gui.sh`

© Copyright IBM Corporation 2014

Netcool/OMNibus installer

The Netcool/OMNibus installer is used to install IBM Installation Manager and then run IBM Installation Manager to install Netcool/OMNibus.

Packages

The screenshot shows the 'Install Packages' interface. At the top, it says 'Select packages to install:' followed by a table titled 'Installation Packages'. The table has columns for 'Package name', 'Status', and 'Vendor'. There are two main sections: 'IBM® Installation' and 'IBM Tivoli Netcool/OMNibus'. Under 'IBM® Installation', there is one entry: 'Version 1.7.2'. Under 'IBM Tivoli Netcool/OMNibus', there are two entries: 'Version 8.1.0.0'. A callout bubble labeled 'Package name' points to the first column of the table. Another callout bubble labeled 'Version' points to the second column of the table. A third callout bubble labeled 'Location of installation files' points to the text 'Repository: /software/omnibus/OMNibusRepository' located in the 'Details' section below the table.

Installation Packages	Status	Vendor
IBM® Installation Version 1.7.2	Will be installed	IBM
IBM Tivoli Netcool/OMNibus Version 8.1.0.0	Will be installed	IBM

Show all versions Check for Other

Details

IBM Tivoli Netcool/OMNibus 8.1.0.0
IBM Tivoli Netcool/OMNibus [More info...](#)
• Repository: /software/omnibus/OMNibusRepository

Packages

Software components are referred to as packages by IBM Installation Manager. Netcool/OMNibus core is considered a package. A software package is contained in a repository. When you use the Netcool/OMNibus installer utility, the installation files are contained in a local repository. In this case, the repository is a directory on disk. IBM Installation Manager lists the product name and version that is found in the repository location.

You have the option of defining the Passport Advantage website as a repository. In this case, IBM Installation Manager connects remotely with a user ID and password that you configure. In this case, IBM Installation Manager lists all software packages to which you are entitled. You then select one or more products for installation.

License agreement

Install Packages
Read the following license agreements carefully.

Install Licenses Location Features Summary

IBM Installation Manager
IBM Tivoli Netcool/OMNibus

International Program License Agreement

Part 1 - General Terms

BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, CLICKING ON AN "ACCEPT" BUTTON, OTHERWISE USING THE PROGRAM, LICENSEE AGREES TO THE TERMS OF THIS AGREEMENT. IF ACCEPTING THESE TERMS ON BEHALF OF LICENSEE, YOU REPRESENT AND WARRANT THAT YOU HAVE THE FULL AUTHORITY TO BIND LICENSEE TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS, DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, CLICK ON AN "ACCEPT" BUTTON, OR USE THE PROGRAM; AND

* PROMPTLY RETURN THE UNUSED MEDIA, DOCUMENTATION, AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM IT WAS OBTAINED FOR A REFUND OF THE AMOUNT PAID. IF THE PROGRAM IS DOWNLOADED, DESTROY ALL COPIES OF THE PROGRAM.

1. Definitions

"Authorized Use" - the specified level at which Licensee is authorized to execute or run the Program. That level may be measured by number of users, millions of service units ("MSUs"), Processed Units ("PVUs"), or other level of use specified by IBM.

"IBM" - International Business Machines Corporation or one of its subsidiaries.

"License Information" ("LI") - a document that provides information and any additional terms and conditions applicable to the use of the Program. The Program's LI is available at www.ibm.com/software/sla. The LI can also be found in the Program's documentation.

I accept the terms in the license agreements
 I do not accept the terms in the license agreements

© Copyright IBM Corporation 2014

License agreement

You must accept the license terms to continue with the installation.

Installation Manager directory locations

Install Packages
Select a location for the shared resources directory and a location for Installation Manager.

Install Licenses Location Features Summary

When you install packages, files are stored in two locations:

1) The shared resources directory - resources that can be shared by multiple packages.
2) The installation directory - any resources that are unique to the package that you are installing.

Important: You can only select the shared resources directory the first time you install a package with the IBM Installation Manager. For best results select the drive with the most available space because it must have adequate space for the shared resources of future packages.

Shared Resources Directory: /home/netcool/IBM/IBMIMShared

Once installed, IBM Installation Manager will be used to install, update, modify, manage and uninstall your packages.

Installation Manager Directory: /home/netcool/IBM/InstallationManager/eclipse

Disk Space Information

Volume	Available Space
/	17.57 GB

© Copyright IBM Corporation 2014

Installation Manager directory locations

The screen capture is based on an installation that uses the Netcool/OMNibus installer. The installer is used to install IBM Installation Manager. The user has the option of selecting the location where IBM Installation Manager is installed. The default location is the home directory of the user who is installing Netcool/OMNibus.

Netcool/OMNibus installation directory

Install Packages
A package group is a location that contains one or more packages. Some compatible packages can be installed into a group and will share a common user interface. Select an existing package group, or create a new one.

Install Licenses Location Features Summary

Use the existing package group
 Create a new package group

Package Group Name	Installation Directory
IBM Tivoli Netcool OMNibus	/opt/IBM/tivoli/netcool

Package Group Name: IBM Tivoli Netcool OMNibus
Installation Directory: /opt/IBM/tivoli/netcool
Architecture Selection: 32-bit 64-bit

Details
Shared Resources Directory: /home/netcool/IBM/IBMIMShared

Disk Space Information
Volume Available Space
/ 17.57 GB

© Copyright IBM Corporation 2014

Netcool/OMNibus installation directory

You can select the location where Netcool/OMNibus is installed. The default is **/opt/IBM/tivoli/netcool**.

Netcool/OMNibus features

The screenshot shows the 'Install Packages' interface with the 'Features' tab selected. The tree view under 'IBM Tivoli Netcool/OMNibus 8.1.0.0' includes:

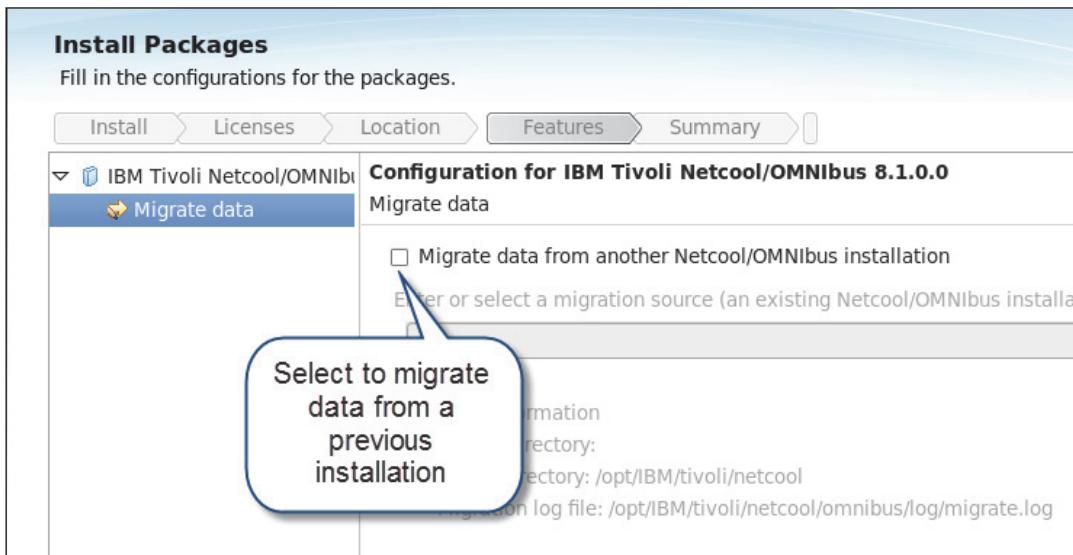
- Administrator components
 - Administrator GUI
 - Administrator tools
 - TEC migration
- Operator components
 - Operator GUI
- Server components
 - ObjectServer
 - ObjectServer gateways
 - Bridge server
 - Proxy server
- Process control components
 - Process agent

© Copyright IBM Corporation 2014

Netcool/OMNibus features

Some software products offer the ability to install various functions as independent product features. Netcool/OMNibus has a number of features that you can install independently or together. Unless disk space is restricted, it is suggested that all features be installed. Installing all features eliminates any potential issues with subsequent installations that might require a specific feature. For example, probes are installed separately. To install a probe, you must first install some Netcool/OMNibus features. If you neglect to install those features, you cannot install the probe.

Data migration



Data migration

IBM Installation Manager is used to upgrade from a previous version of Netcool/OMNIBus. When upgrading, you select the option to migrate data. When that option is selected, you enter the path for the old Netcool/OMNIBus installation.

Installation summary

Install Packages
Review the summary information.

Install > Licenses > Location > Features > **Summary** >

Target Location

Package Group Name: IBM Tivoli Netcool OMNibus
Installation Directory: /opt/IBM/tivoli/netcool
Shared Resources Directory: /home/netcool/IBM/IBMIMShared

Packages

Packages

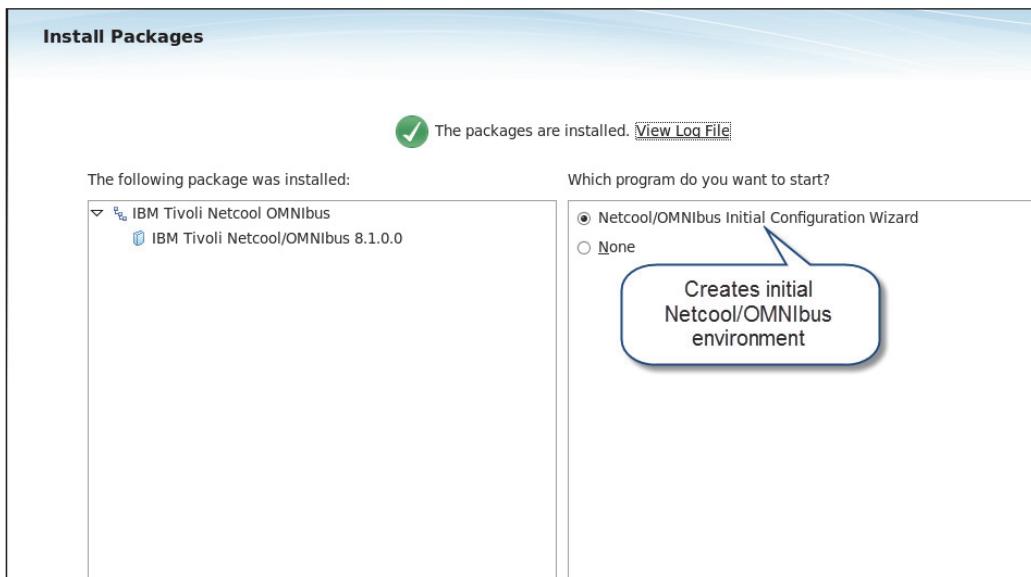
- IBM® Installation Manager 1.7.2
- IBM Tivoli Netcool/OMNibus 8.1.0.0
 - Administrator components
 - Operator components
 - Server components
 - Process control components
 - Probe and gateway components
 - Extensions

© Copyright IBM Corporation 2014

Installation summary

IBM Installation Manager displays a summary of the installation options. A typical Netcool/OMNibus core installation runs approximately 10 minutes.

Installation complete



© Copyright IBM Corporation 2014

Installation complete

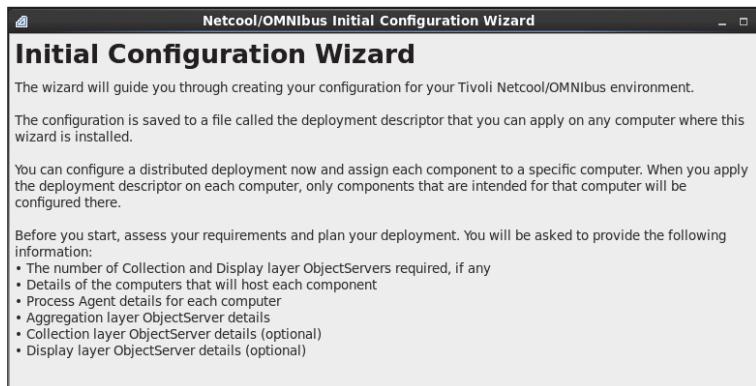
When the installation is complete, a status page opens. If the installation fails for some reason, you click **View Log Files** and review the installation log for the specific issue. If you run the prerequisite scanner and the results pass, the installation is not likely to fail.

Leave the option selected to run the Netcool/OMNIBus Initial Configuration Wizard.

Initial Configuration Wizard

Creates Netcool/OMNibus environment

- Primary ObjectServer
- Backup ObjectServer
- Interfaces file
- Gateway configuration
- Process Activity configuration



© Copyright IBM Corporation 2014

Initial Configuration Wizard

The Initial Configuration Wizard is a separate utility that you can optionally use to create the initial Netcool/OMNibus environment. The primary benefit to using this utility is the potential elimination of initial configuration issues that are due to human error. The utility eliminates the need to perform many manual steps.

Another benefit of using the utility is that the initial configuration is created with the suggested best-practice settings.

Task options

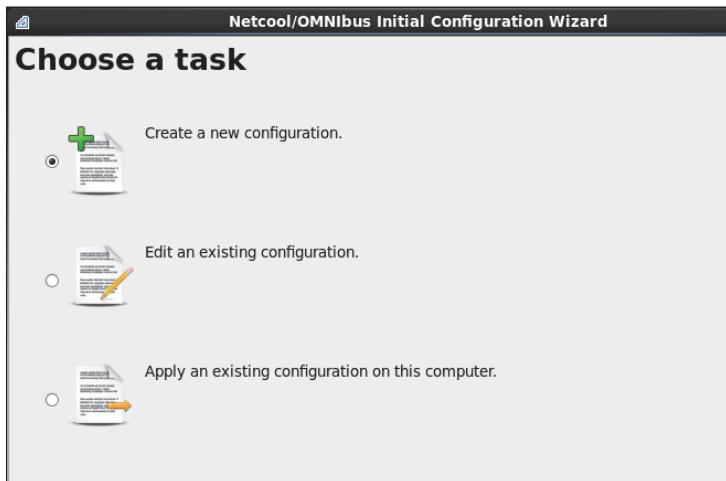
Wizard creates an XML:

- The XML file contains a list of components to create
- Wizard uses the XML file to create the components on the local system

Create new creates a new file

Edit modifies an existing file

Apply uses the file to create the components



© Copyright IBM Corporation 2014

Task options

You can use the utility to create a new configuration, update an existing configuration, or to deploy a configuration.

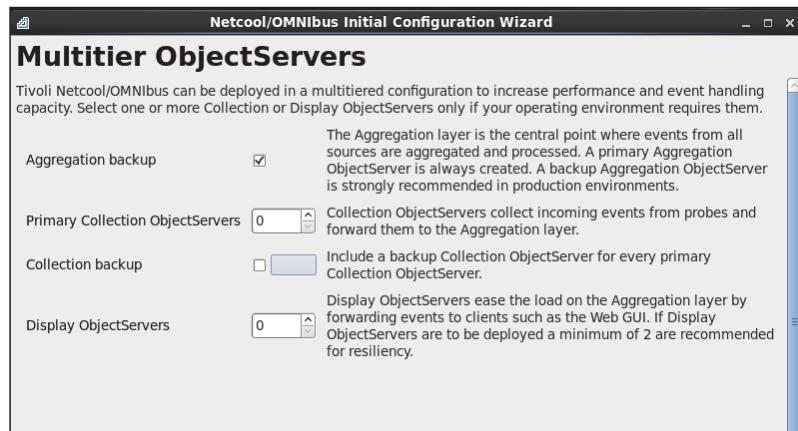
ObjectServers

Default:

- Create primary aggregation

Optional:

- Create backup aggregation
- Create collection
- Create display



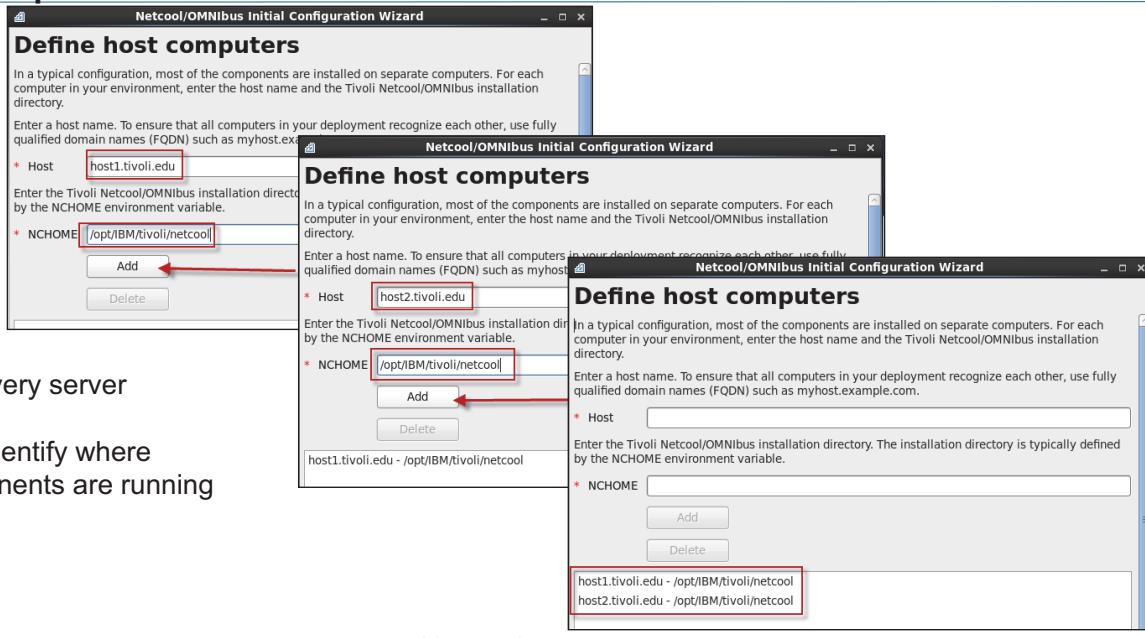
© Copyright IBM Corporation 2014

ObjectServers

The first requirement for the utility is to identify the type of configuration that is created. The type of configuration is identified based on the numbers of servers. The utility can create a multitier configuration or a single tier configuration. You can use the utility also to create high-availability ObjectServers for the single or multitier options.

A single-tier configuration is based on the Aggregation ObjectServer. The single-tier deployment is essentially one layer of the multitier deployment. You create a single-tier deployment in your student exercise.

Host computers



Identify every server

Used to identify where components are running

Host computers

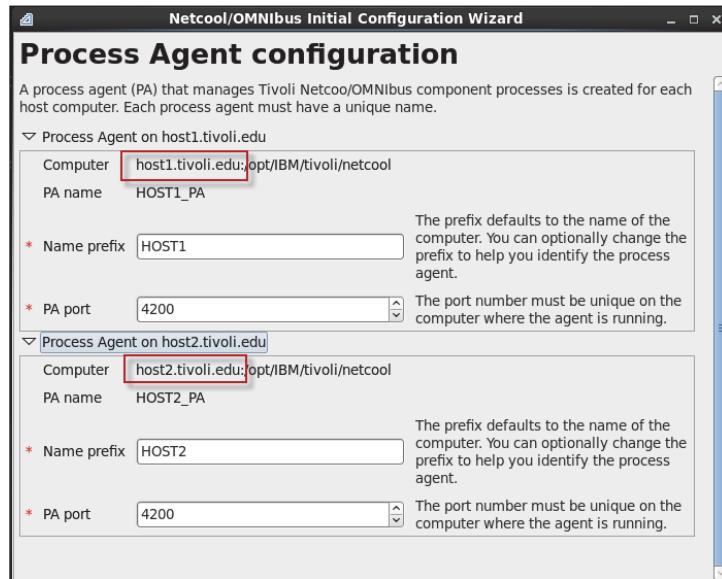
The next step in the utility is to identify the servers that contain Netcool/OMNibus components. The option exists to identify a server that is based on the IP address or with a short host name. The recommendation is to identify each server with a fully qualified host name.

The deployment in the student exercises is based on two servers.

Process agents

Automatically creates a definition for every server

Assumes that a process agent is installed on each server



© Copyright IBM Corporation 2014

Process agents

The utility creates the Process Activity configuration that is used to automatically start the Netcool/OMNibus components on their respective servers.

Primary aggregation ObjectServer

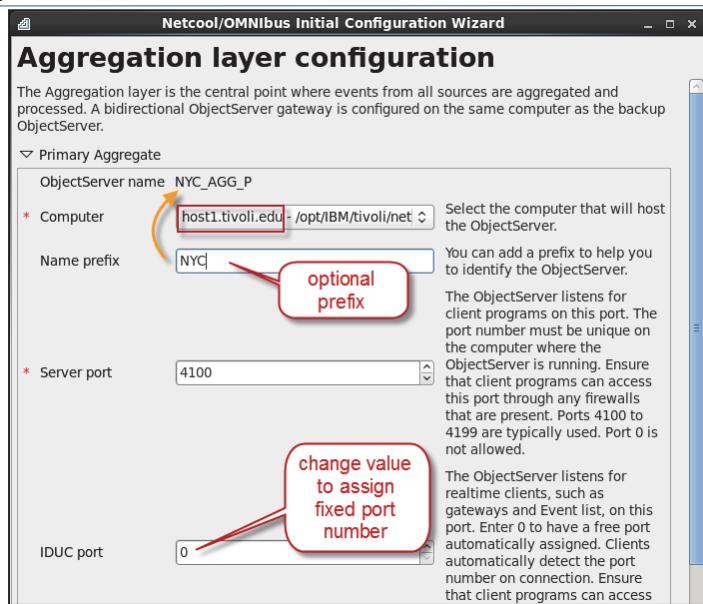
Base name is AGG_P and cannot be changed

Optionally add text as a prefix to the name

Define the listening port

Optionally configure IDUC port

ObjectServer picks a port at random if not assigned



© Copyright IBM Corporation 2014

Primary aggregation ObjectServer

To use the wizard, you must accept the default convention for names that is used by the wizard. This is a requirement because component names show in various configuration files. However, one option that is available is the ability to apply a user-specified prefix to the ObjectServer names.

By default, the primary Aggregation layer ObjectServer is named AGG_P. You can include a prefix such as NYC, which causes the name to change to NYC_AGG_P. The maximum length for the ObjectServer name is 29 characters; so the prefix can be longer than this example.

The other requirement for the ObjectServer configuration is the name of the server that hosts the ObjectServer and the port numbers that are used by the ObjectServer.

Backup aggregation ObjectServer and synchronizer gateway

Base name is AGG_B and cannot be changed

Optionally add text as a prefix to the name

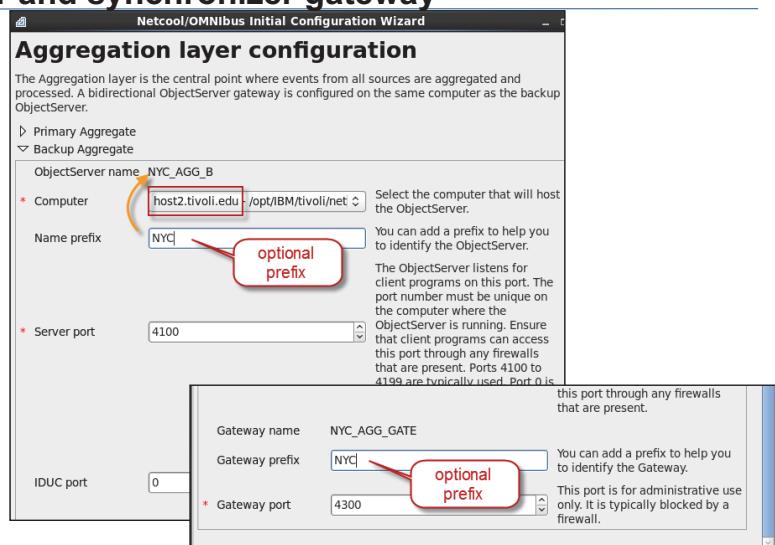
Define the listening port

Optionally configure IDUC port

ObjectServer picks a port at random if not assigned

Gateway base name is AGG_GATE and cannot be changed

Optionally add text as a prefix to the gateway name



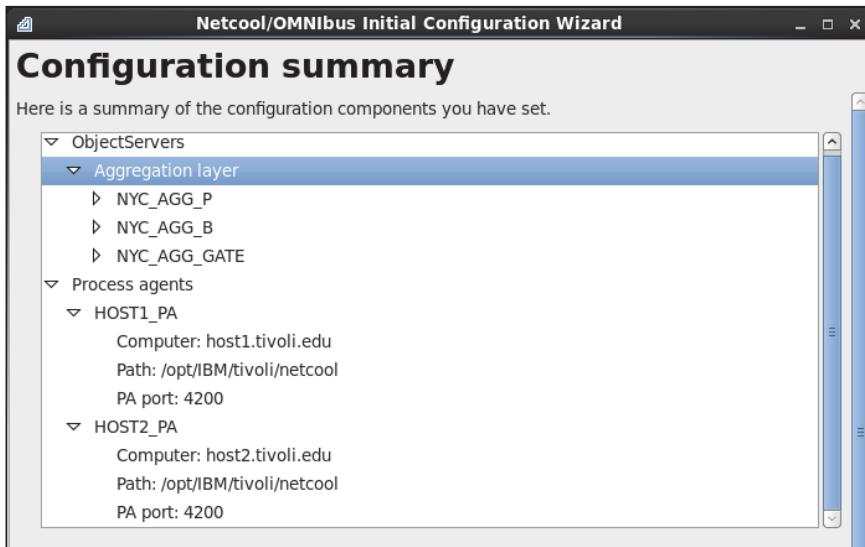
© Copyright IBM Corporation 2014

Backup aggregation ObjectServer and synchronizer gateway

The default name for the backup Aggregation ObjectServer is AGG_B. You can optionally apply a prefix to this name also. In most production environments, the backup ObjectServer is not hosted on the same server as the primary ObjectServer. You select another server to host the backup ObjectServer.

You also provide data that is used to configure the ObjectServer gateway. The gateway is automatically configured to be hosted on the same server as the backup ObjectServer. Running the gateway with the backup ObjectServer is the suggested location.

Summary



© Copyright IBM Corporation 2014

Summary

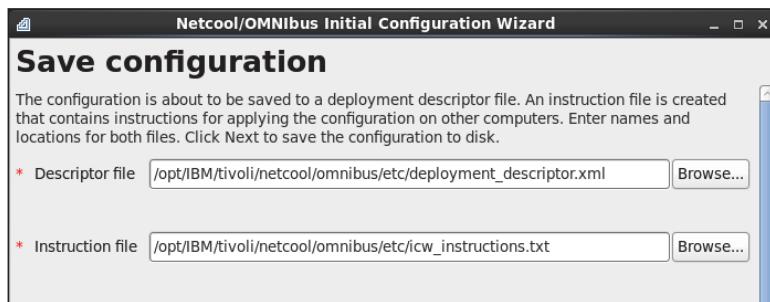
The wizard displays a summary of the components that are configured. For the aggregation layer with ObjectServer high availability, the wizard defines these items:

- Primary ObjectServer
- Backup ObjectServer
- ObjectServer Gateway
- Process Agent for the server that hosts the primary ObjectServer
- Process Agent for the server that hosts the primary ObjectServer

Output files

Descriptor file defines what components are created

Instruction file contains textual instructions for how to apply the configuration to local and remote systems

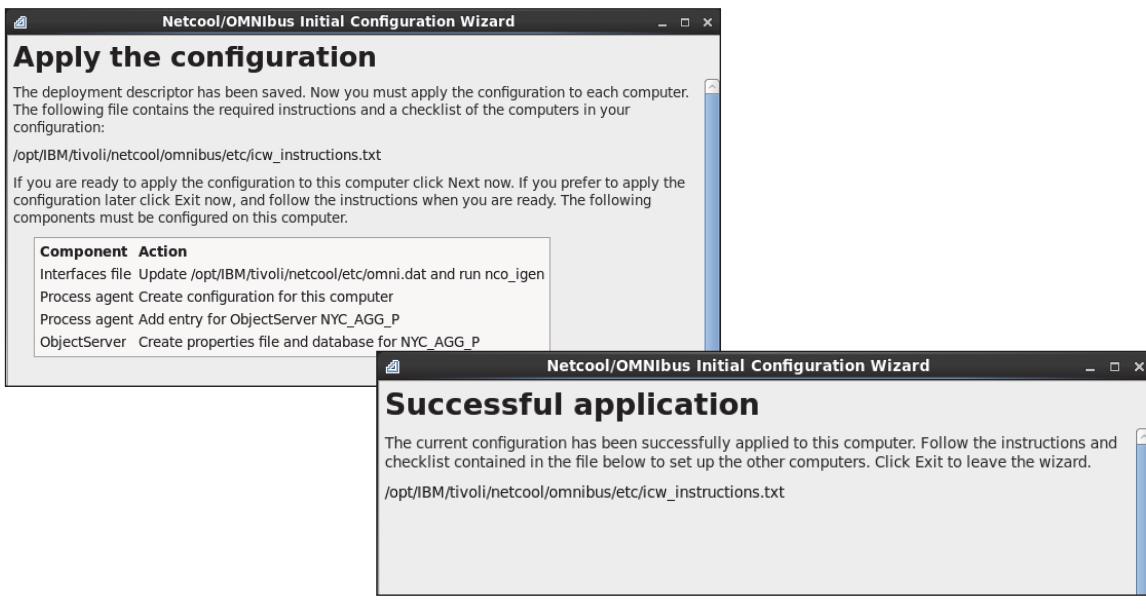


© Copyright IBM Corporation 2014

Output files

The wizard creates two text files: descriptor file and instruction file. The descriptor file is used by the wizard to create the components: ObjectServers, gateways, and process agents. The instruction file contains text that describes how to use the wizard to deploy the configuration. Up to this point none of the components are created. The only things that exist are the two text files.

Apply the configuration



© Copyright IBM Corporation 2014

Apply the configuration

The last step in the wizard is to *apply* the configuration. The wizard reads the descriptor file and creates the components that are defined to run on the local host. In a high-availability configuration, components are defined on two servers. One server hosts the primary ObjectServer and a process agent. The second server hosts the backup ObjectServer, ObjectServer gateway, and a process agent. The wizard uses the host name in the descriptor file to determine which components to create on the local server. In a high-availability configuration, you can run the wizard on server1 and define all components. When you apply the configuration on server1, the wizard creates only the components for that server. For server2, you can either rerun the wizard and create the same configuration or you copy the descriptor file from server1 to server2. If you copy the descriptor file, you need only to run the wizard and *apply* the configuration. The wizard then creates the components for server2.

To run the wizard from the command line, you enter the following command:

```
/opt/IBM/tivoli/netcool/omnibus/bin/nco_icw
```



Lesson 3 Preliminary operation



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to create and run an ObjectServer. You also learn how to use a supplied probe to verify basic function. After completing this lesson, you should be able to perform the following tasks:

- Describe ObjectServer communications
- Create an ObjectServer
- Start an ObjectServer
- Start a probe

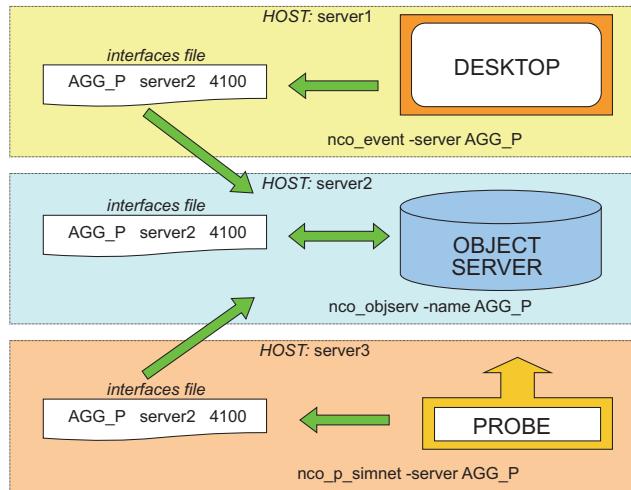
Tivoli Netcool/OMNIbus communications

Tivoli Netcool/OMNIbus components use the IDUC protocol to communicate Insert, delete, update, control

IDUC uses an interfaces file as an address book

Servers, for example, ObjectServers, consult the interfaces file to start up

Clients, such as desktops and probes, consult the interfaces file to know how to reach their server



© Copyright IBM Corporation 2014

Tivoli Netcool/OMNIbus communications

Netcool/OMNIbus is a distributed client/server system and not constrained by hardware system. The architecture can be configured with all components on a single server. It can also be configured with each component on a separate server, or with a combination of components on the same server. All Netcool/OMNIbus components use the TCP-based IDUC (insert, delete, update, control) protocol to communicate.

This protocol identifies server components: ObjectServers, proxy servers, and process agents with a unique server name, by convention, in uppercase. This unique server name corresponds to the socket, host name, and port number, of the server.

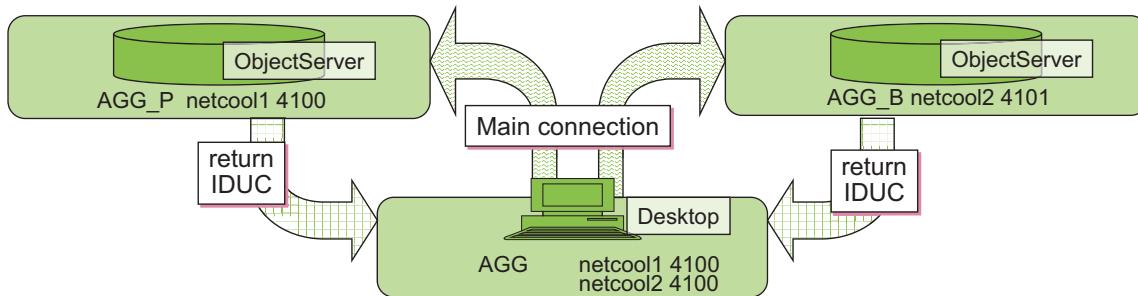
This information is stored in an interfaces file in the **\$NCHOME/etc** directory. This directory is referred to as the address book. The format of the interfaces file is platform-specific. An interfaces file written on one architecture, for example, AIX®, cannot be used on another architecture, for example, Linux.

Server components access the interfaces file when they start. Client components, desktops, and probes, access the interfaces file to know where to connect to their named ObjectServer.

You should not edit this text-only interfaces file directly. Instead, you can use GUI and non-GUI editing utilities that ensure correct formatting of the interfaces file.

IDUC in firewall environments

By default, the ObjectServer selects a random port to use for return IDUC and informs desktops and gateways which port it will use



In a firewall environment, you must use one of the following methods to statically set the outbound IDUC port

- Add an entry for the ObjectServer in /etc/services
 - nco_
 - AGG_P
 - 7890/tcp
- Use the -listeningport command line option for nco_objserv
 - \$OMNIHOME/bin/nco_objserv -name AGG_P -listeningport 7890 &

IDUC in firewall environments

With IDUC, the ObjectServer can inform the desktops and gateways that they must request an update. The value of the *granularity* property controls the frequency of the update. Every 60 seconds, by default, the ObjectServer updates all its IDUC clients with the event data.

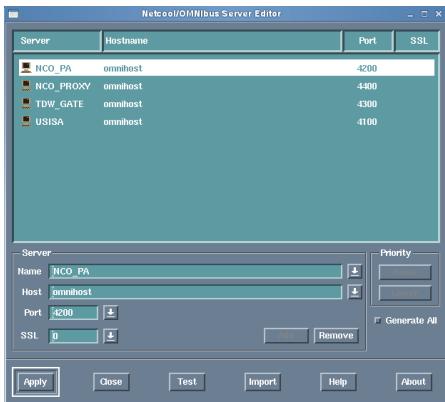
By default, IDUC is set to a random, unused port number. The ObjectServer randomly selects an available port to use for IDUC and informs all clients of its value.

In a firewall environment, you must statically set the IDUC port to enable the definition of the firewall rules. This setting ensures that IDUC network traffic is allowed through the firewall.

The port of an ObjectServer can be set either within the **/etc/services** file, by using the **-listeningport** command-line option for nco_objserv, or the **Iduc.ListeningPort** ObjectServer property value.

Configuring the interfaces file

Using the GUI: nco_xigen is a GUI tool to configure the interfaces file



Launch with
\$OMNIHOME/bin/nco_xigen

Define server name, host, and port

Click Apply to generate these files

\$NCHOME/etc/omni.dat

\$NCHOME/etc/interfaces.<arch>

*The Configuration Wizard does this process automatically
No longer requires a manual step*

Using the command line:

- Edit \$NCHOME/etc/omni.dat to add servers in established file format
- Save and run \$NCHOME/bin/nco_igen to create the interfaces file

© Copyright IBM Corporation 2014

Configuring the interfaces file

Two files are associated with IDUC communications. The first file is \$OMNIHOME/etc/omni.dat. This file is a textual list of specific Netcool/OMNIbus components: ObjectServers, gateways, and process activity agents. The file contains one entry for each component. The entry for each component contains a textual name for the component. The entry also contains the server, host name, or IP address, where the component runs and the TCP/IP port number the component uses. If SSL encryption is required, the entry contains the SSL port number. You can manually edit this file with any conventional text editor. Netcool/OMNIbus provides a graphical utility that offers a more convenient mechanism for manipulating the contents of this file. This utility is nco_xigen and is installed when the desktop feature is selected.

When **Apply** is clicked in this utility, the **omni.dat** file is written to disk and a second utility is started, nco_igen. It is the nco_igen utility that reads the **omni.dat** file and creates the platform-specific interfaces file.

The second file that is associated with IDUC communications is the interfaces file. This file is unique to each operating system and is in the directory, **\$NCHOME/etc/interfaces.<arch>**, where <arch> is a platform-specific suffix. The nco_igen utility creates this file. You should not edit this file manually. Instead, the user should update the **omni.dat** file manually or use the nco_xigen utility. You can then run nco_igen manually or use the nco_xigen utility.



Important: The Initial Configuration Wizard creates the **omni.dat** and interfaces file. If the wizard is used, you do not have to manually create these files.

Defining the interfaces file on a server

Servers use the interfaces file to determine how to start

On a server, perform the following tasks:

- Enter the ObjectServer name, for example, AGG_P
- Enter the host name and port number, for example, netcool1 4100
- If you are using SSL, enter the SSL port number
- Click **Add** to create a new entry, or **Update** to modify an existing entry

Note: Server name must match the ObjectServer name

© Copyright IBM Corporation 2014

Defining the interfaces file on a server

All Netcool/OMNibus components, except for the Web GUI, use IDUC communications and require a copy of the interfaces file.

When you start an ObjectServer, you provide the name on the command line. The ObjectServer uses this name to locate the files that define its structure, and uses that name to locate the corresponding entry in the interfaces file. This entry tells the ObjectServer what TCP/IP port number to use when running.



Note: The use of the term **server** on this slide is a reference to a **server component**. In Netcool/OMNibus, the ObjectServer and Proxy server are considered **server components**.

Defining the interfaces file on a client

Clients (desktops, probes, gateways) must know how to reach their target ObjectServer

- Host name
- Port number

The client is not defined in the interfaces file

On a client, you perform the following tasks:

- Enter a name for the ObjectServer, for example, AGG_P
- Enter the host name and port number, or SSL port number, for example: netcool1 4100
- Click **Add**

© Copyright IBM Corporation 2014

Defining the interfaces file on a client

Systems that host desktops, probes, and gateways also require an interfaces file. When a native desktop is activated, the user is provided a list of available ObjectServers. The list of ObjectServer names comes directly from the interfaces file. When the user selects an ObjectServer from the list, the desktop uses the interfaces file to determine how to communicate with the ObjectServer, what host and port number. The same applies to probes and gateways. However, with probes and gateways the ObjectServer name is provided in a property file. When the probe or gateway starts, it reads the property file and then uses the interfaces file to determine how to communicate with the configured ObjectServer.

Netcool/OMNIbus is typically deployed in a heterogeneous hardware environment. ObjectServers might be running on AIX systems, probes might be running on Linux systems, and desktops, might be running on Windows systems. Every system must have a copy of the interfaces file and the files are platform-specific.



Note: The use of the term *client* on this slide is a reference to a *client* component. In Netcool/OMNIbus, the native desktop, probes, and gateways are considered client components.

ObjectServer names

The user configures the ObjectServer name

- Uppercase letters and integers only
- Cannot begin with an integer
- No spaces, and no special characters, except the underscore (_)
- Maximum length 29 characters

Typical conventions

- Name based upon geography, for example, NYC, LON
- Suffix indicates role
 - NYC_P = Primary
 - NYC_B = Backup

Configuration Wizard

- Primary aggregation ObjectServer is AGG_P, and cannot be changed
- Option to add a prefix to the name, for example: LON_AGG_P

© Copyright IBM Corporation 2014

ObjectServer names

You can choose the component names that the interfaces file uses. You must follow some rules that are associated with these names, but you can choose other naming conventions.

The name must be all uppercase alphabetic characters and numeric characters. It can contain integers but cannot start with an integer. No spaces are allowed in the name. No special characters are allowed, except for the underscore. The maximum length is 29 characters.

Users often name their ObjectServers based on a convenient geographic characteristic, like the city where the server is located. Many Netcool/OMNibus deployments use multiple ObjectServers for different roles. For example, in a high-availability configuration, there are two ObjectServers: primary and backup. By applying a suffix to the ObjectServer name, you can indicate the specific role.

The best practice advice is to use the Initial Configuration Wizard when deploying Netcool/OMNibus. This technique ensures that the components are configured with the suggested best practice property settings. However, to use the wizard, you must adhere to the component names the wizard uses, which include ObjectServer names. You can use your own ObjectServer names. However, you must manually modify the supplied configuration files to adjust all references to the ObjectServer names.

Building the ObjectServer

Memory resident SQL database

Event data repository central to Netcool product suite

Creates the database framework at run time

Databases, tables, columns, permissions

nco_dbinit script creates new ObjectServers

- \$OMNIHOME/bin/nco_dbinit -server <ObjServName> creates these files:
 - Properties (.props) file in \$OMNIHOME/etc
 - All database files in \$OMNIHOME/db/<ObjServName>/
- Creates an ObjectServer with two default users
 - root : Full superuser privileges and access
 - Nobody: For events having no user

*The Configuration Wizard does this process automatically
No longer requires a manual step*

© Copyright IBM Corporation 2014

Building the ObjectServer

In other database systems, the database is created with a long list of commands, a GUI wizard, or a script, that sets up the database framework. In these systems, this process must be run only once because the framework is held on persistent storage media.

Because the ObjectServer is memory resident, the database framework is created in memory each time the ObjectServer starts up.

The nco_dbinit utility is used to create the ObjectServer. The \$OMNIHOME/bin/nco_dbinit utility creates specification files in a subdirectory of \$OMNIHOME/db/. It populates certain tables and creates a props file in \$OMNIHOME/etc containing runtime properties. The -force option overwrites an existing server name.



Important: The Initial Configuration Wizard creates the ObjectServer. If the wizard is used, it is not necessary to manually create the ObjectServer.

Starting a Netcool/OMNibus system

- Configure communications address book interfaces file
- Build the ObjectServer
- Start ObjectServer
 - nco_objserv**
 - Example: \$OMNIHOME/bin/nco_objserv -name AGG_P &
 - Starts NCMS. Check \$OMNIHOME/log/AGG_P.log for errors
- Start probe
 - nco_p_probename**
 - Example: \$OMNIHOME/probes/nco_p_simnet -server AGG_P &
 - Starts a simnet probe and sends its output to NCMS. Check \$OMNIHOME/log/simnet.log for errors
- Start desktop
 - nco_event**
 - Example: \$OMNIHOME/bin/nco_event
 - Starts Event List

© Copyright IBM Corporation 2014

Starting a Netcool/OMNibus system

If you forget the exact syntax of any Netcool/OMNibus command, you can use the -help option.

Entering the command `$OMNIHOME/bin/nco_objserv -name NAME` & starts the ObjectServer NAME in the background. This command is valid if NAME is in the local interfaces file, and NAME is built with the `nco_dbinit` command.

The command `$OMNIHOME/bin/nco_ping NAME` can be used to verify whether the ObjectServer is running. If it is not, check the contents of the `$OMNIHOME/log/NAME.log` for errors.

The command `$OMNIHOME/probes/nco_p_simnet -server NAME` & starts the Simnet probe in the background. This Simnet probe generates simulated events at random. It sends its data to the NAME ObjectServer. Probes are not in the `bin` directory, but in the `probes` directory.

The `$OMNIHOME/bin/nco_event` command starts the desktop. Then you can log in to an ObjectServer. You can start an event list to look at and manipulate its contents.



© Copyright IBM Corporation 2014

Student exercises

Perform the exercises for this unit.

Summary

You now should be able to perform the following tasks:

- Install Tivoli Netcool/OMNibus V8.1 core product
- Update communications file
- Create and start a new ObjectServer
- Start a probe
- Use the desktop to view ObjectServer events

© Copyright IBM Corporation 2014

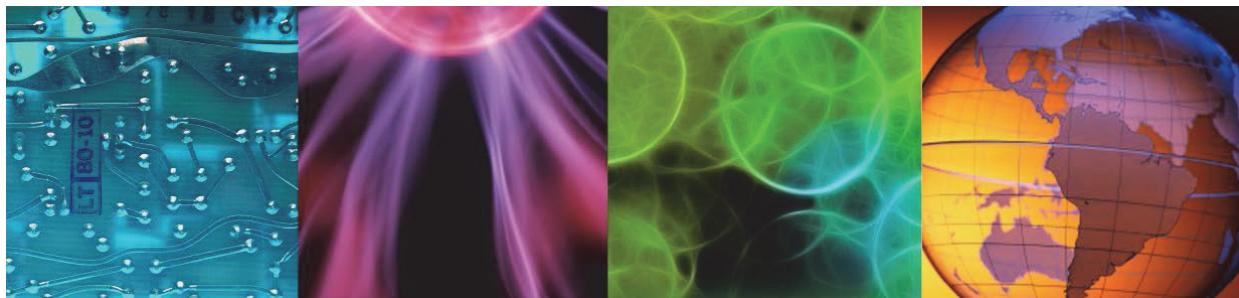
Summary



4 Installing Netcool/OMNIbus Web GUI



4 Installing Netcool/OMNIbus Web GUI



© Copyright IBM Corporation 2014

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

This unit covers the installation of Web GUI and a brief introduction to the features of the product.

References: SC27-6264-00 *Installation and Deployment Guide*
SC27-6505-00 *Web GUI Administration and User's Guide*

Objectives

In this unit, you learn to perform the following tasks:

- Describe the functional architecture of the Web GUI component
- Describe the features and functions
- Describe the requirements for installation
- Perform a complete default installation
- Perform the postinstallation steps
- Validate basic functions
- Configure Dashboard Application Services Hub to use an LDAP repository

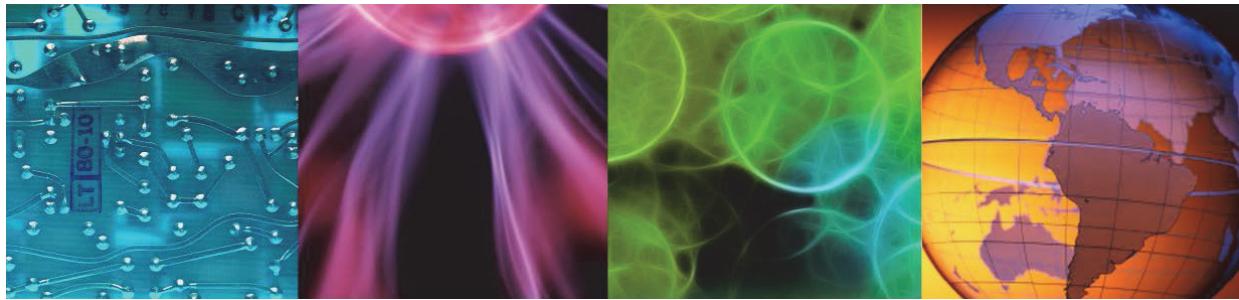
© Copyright IBM Corporation 2014

Objectives

Lesson 1 Installing Netcool/OMNIbus Web GUI



Lesson 1 Installing Netcool/OMNIbus Web GUI



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to install and initialize the Web GUI component.

Installation

Web GUI requires the following components:

- Jazz for Service Management
- Dashboard Application Services Hub

These are not bundled with Web GUI

IBM Installation Manager is used to install these items:

- Jazz for Service Management
- Dashboard Application Services Hub
- WebSphere Application Server
- Web GUI

Same installation options as Netcool/OMNIBus core

- Install packages using local copies of files
- or
- Configure IBM Installation Manager to download files and install

© Copyright IBM Corporation 2014

Installation

Jazz™ for Service Management brings together the Open Services for Lifecycle Collaboration (OSLC) community's open specifications for linking data and other shared integration services, including administrative, dashboard, reporting, and security services. It underpins client-defined management scenarios such as cloud, performance monitoring, and IT Service Management.

The integration services provide key features, such as these examples:

- Shared data repository for products that integrate through Registry Services in Jazz for Service Management
- Consistent dashboard and visualization experience through IBM Dashboard Application Services Hub in Jazz for Service Management
- Simplified administration of products and solutions that integrate through Administration Services in Jazz for Service Management
- Ad hoc, self-service reporting through IBM Tivoli Common Reporting in Jazz for Service Management
- Lightweight Third-Party Authentication (LTPA) single sign-on participation by non-WebSphere IBM® and other application servers that integrate through Security Services in Jazz for Service Management

Dashboard Application Services Hub is also known as the Visualization Service within Jazz for Service Management. Dashboard Application Services Hub is based on WebSphere®.

Netcool®/OMNIBus Web GUI is a collection of applications that run within WebSphere. The user interface to visualize these applications is Dashboard Application Services Hub.

To install Netcool/OMNIbus Web GUI, you must install Jazz for Service Management, Dashboard Application Services Hub, and WebSphere. Optionally, you can install Web GUI into an existing deployment of these components.

Installing Jazz for Service Management

Retrieve installation files

- Jazz for Service Management
- WebSphere Application Server

IBM Installation Manager install file is bundled with Jazz for Service Management

Expand files into a temporary directory

- Jazz installation files are in JazzSMRepository
 - WebSphere installation files are in WASRepository
- IBM Installation Manager installation files are in im.linux.x86_64

Start the installation launchpad

`./launchpad`

© Copyright IBM Corporation 2014

Installing Jazz for Service Management

The first step when installing Web GUI is to install Jazz for Service Management, Dashboard Application Services Hub, and WebSphere. These components are installed with IBM Installation Manager. The installation files for Dashboard Application Services Hub are bundled with Jazz for Service Management. The WebSphere installation files are separate.

You retrieve Jazz for Service Management and WebSphere from Passport Advantage. Expand the files into a temporary directory. This process creates the repository directories that IBM Installation Manager uses.

The Jazz for Service Management installation files includes an installation utility referred to as launchpad. The launchpad utility is used to install IBM Installation Manager. Then, IBM Installation Manager is used to install the remaining components.

Installation options

Full

- Installs DB2, Jazz, WAS and all features

Custom

- Select packages
- Select features within packages

Tools

- Option to manually run:
 - DB2 setup
 - IBM Installation Manager
 - IBM Prerequisite Scanner
 - Tivoli Common Reporting installation

Welcome to Jazz for Service Management 1.1.0.3

Jazz for Service Management provides the following integration services: Administration Services, IBM Dashboard Application Services Hub, IBM Tivoli Common Reporting, Registry Services, and Security Services. Use the launchpad to guide you through performing either a full or custom installation.

For links to the latest release and support information, expand **Release Information**.

Click one of the following links to guide you through installing Jazz for Service Management:

- [Full](#)

Use full installation for evaluation or development purposes. After you provide information, the launchpad silently installs Jazz for Service Management integration services, IBM DB2, and IBM WebSphere Application Server on a single server.

- [Custom](#)

Use the custom workflow to install and update specific Jazz for Service Management integration components and supporting middleware.

- [Tools](#)

Use Tools to run the following tools from the launchpad: IBM DB2 Setup Launchpad, IBM Installation Manager, IBM Prerequisite Scanner, and IBM Tivoli Common Reporting installation program

© Copyright IBM Corporation 2014

Installation options

The launchpad utility provides two options for installation: Full and Custom.

The Full option can be used if all components of Jazz for Service Management are installed. Jazz for Service Management provides a comprehensive collection of services. Not all of these services are required for Web GUI.

With the Custom option, you can select which components of Jazz for Service Management are installed. You use this option in your student exercises.

The launchpad also provides an option to run one or more tools. These tools can be used to do things like run a prerequisite scan or configure an instance of DB2.

Custom installation

Custom Workflow

Use this workflow to install or update the following Jazz for Service Management integration services and supporting middleware:

- Administration Services
 - Administration service provider
 - User Interface
- Registry
- Reporting, provided by IBM Tivoli Common Reporting
- Security
- Visualization, provided by IBM Dashboard Application Services Hub
- IBM WebSphere Application Server
- IBM DB2 Enterprise Server Edition (installation only)

Required for Web GUI

- Visualization (Dashboard Application Services Hub)
- IBM WebSphere Application Server

© Copyright IBM Corporation 2014

Custom installation

The Jazz for Service Management launchpad provides the option to install one or more of the following features:

- Administrative Services
- Registry Services
- Tivoli Common Reporting
- Security Services
- Visualization Services (Dashboard Application Services Hub)
- IBM WebSphere Application Server
- IBM DB2

When the Full option is used, all of these components are installed and you cannot select individual components. Web GUI requires only Dashboard Application Services Hub and WebSphere.

Jazz for Service Management Home Location

Specify Jazz for Service Management Home Location

Existing environment

Existing Jazz for Service Management home location:

Leave blank for a new installation

Enter path to update existing copy of Jazz for Service Management

© Copyright IBM Corporation 2014

Jazz for Service Management Home Location

To reuse an existing copy of Jazz for Service Management, you enter the directory path in this screen. If the screen is left blank, a new copy is installed.

Source Locations

Locations for installation files

Existing files are located based on the directory names:

- JazzSMRepository
- WASRepository

Not installing DB2 or Tivoli Common Reporting

The screenshot shows the "Specify Source Locations" dialog box. It contains two warning messages at the top:

- The IBM DB2 10.1 or later, Enterprise Server Edition installation image field is empty. This field is required when you want to install DB2.
- The IBM Tivoli Common Reporting 3.1 or later installation image field is empty. This field is required when you want to install Tivoli Common Reporting.

Below the messages, there are sections for specifying source locations:

- Jazz for Service Management 1.1 or later package repository:
- DB2, WebSphere Application Server, and Tivoli Common Reporting source locations:
 - IBM DB2 10.1 or later, Enterprise Server Edition installation image:
 - IBM WebSphere Application Server 8.5 or later package repository:
 - IBM Tivoli Common Reporting 3.1 or later installation image:
 - IBM Tivoli Common Reporting 3.1.0.2 or later update image:

© Copyright IBM Corporation 2014

Source Locations

When the launchpad is used for the installation, the utility determines the location of the repositories. In the sample screen capture, you see that the launchpad finds the repositories for Jazz for Service Management and WebSphere. It did not find a repository for Tivoli Common Reporting or DB2.

Select Operations

Select Operations

For each component, select the operation to run. Select **None** if you do not want to run an operation for a specific component.

Component	Current	Target	Operation
Administration Services	-	1.1.0.3	<input type="button" value="None"/>
Service provider	-	1.1.0.3	<input type="button" value="None"/>
User Interface	-	1.1.0.3	<input type="button" value="None"/>
Registry Services	-	1.1.0.3	<input type="button" value="None"/>
Reporting Services	-	1.1.0.3	<input type="button" value="None"/>
Security Services	-	1.1.0.3	<input type="button" value="None"/>
Visualization Services	-	3.1.0.3	<input type="button" value="Install"/>
IBM DB2 Enterprise Server Edition	10.5.0.3	-	<input type="button" value="None"/>
IBM WebSphere Application Server	-	8.5.0.1	<input type="button" value="Install"/>

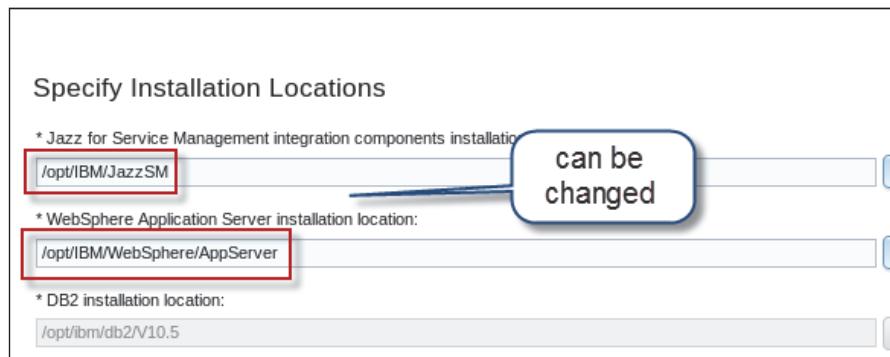
change these to None

© Copyright IBM Corporation 2014

Select Operations

When the Full option is used for the installation, this screen does not open. With the Custom option, you can select which components to install. When installing Jazz for Service Management for only use by Web GUI, you must change the options to **None** for the components that are not required.

Installation Locations



© Copyright IBM Corporation 2014

Installation Locations

This screen capture shows the default installation locations for Jazz for Service Management and WebSphere. You can change the locations if necessary.

Administrator user

The screenshot shows a 'Specify Credentials' dialog box. At the top, it says 'WebSphere Application Server Profile'. Below that, there are three input fields:

- 'Administrator name:' followed by the value 'smadmin'.
- 'Administrator password:' followed by a series of six asterisks '*****'.
- 'Confirm administrator password:' followed by a series of six asterisks '*****'.

© Copyright IBM Corporation 2014

Administrator user

WebSphere requires an administrative user. This user is the *superuser* that has all of the administrative authorities. The screen capture shows the default user name, **smadmin**. You must provide a password.

Installation tasks

Run Tasks

The following tasks will be run in a sequence:

Task

- Check task dependencies
- Check system prerequisites
- Install IBM WebSphere Application Server 8.5.0.1
- Install Jazz for Service Management extension for IBM WebSphere 8.5 1.1.0.2
- Install IBM Dashboard Application Services Hub 3.1.0.3

Click **Run** to continue.

Installation runs approximately 1 hour

© Copyright IBM Corporation 2014

Installation tasks

The installer completes a series of tasks, including running a prerequisite check. The prerequisite scanner contains an option for verifying the installation requirements for Jazz for Service Management. You can run the verification manually before the installation, or the installer can run the verification. If the verification fails during the installation process, the installation stops. You must correct any deficiencies before you restart the installation.



Important: The installation runs for approximately 1 hour.

Review results

The screenshot shows a 'Review Results' page with a table of tasks and their statuses. A red box highlights the 'Check system prerequisites' task, which has a yellow 'Warning' status. A red arrow points from this warning to the text 'Prerequisite scanner fails because of CPU speed'.

Task	Status	Action
Check task dependencies	Complete	View Details
Check system prerequisites	Warning	View Details
Install IBM WebSphere Application Server 8.5.0.1	Complete	View Details
Install Jazz for Service Management extension for IBM WebSphere 8.5.1.1.0.2	Complete	View Details
Install IBM Dashboard Application Services Hub 3.1.0.3	Complete	View Details

Property	Result	Found	Expected
os.RAMSize	PASS	5.7GB	4GB
os.space.shared	PASS	NOT_REQ_CHECK_ID	[dir:root=USERHOME]185MB
os.space.shared.nonroot	PASS	20480MB	[dir:non_root=USERHOME]185MB
intel.cpu	FAIL	2.20GHz	2.4GHz
os.space.opt.root	PASS	NOT_REQ_CHECK_ID	[dir:root=/opt/IBM/JazzSM]675MB
os.space.opt.nonroot	PASS	20480MB	[dir:non_root=/opt/IBM/JazzSM]675MB

ODP - Common prerequisites in JazzSM [ODP]:

Property	Result	Found	Expected
OS Version	PASS	Red Hat Enterprise Linux Server rel...	Red Hat Enterprise Linux Server release Red Hat Enterprise Linux Server 6.*

© Copyright IBM Corporation 2014

Review results

The screen capture shows a sample of the results of the prerequisite scan as completed during the installation. In this example, the scan failed because of an insufficient processor speed. This failure is considered noncritical so the installation is allowed to continue.

Dashboard Application Services Hub

The image consists of two side-by-side screenshots. On the left, the 'IBM Dashboard Application Services Hub' login page is shown. It features a dark header with the title, a user ID field containing 'smadmin', a password field with masked input, and a blue 'Go' button. Below the fields is a copyright notice: '© Copyright IBM Corp. 2005, 2014.' On the right, a screenshot of the dashboard interface is displayed. It has a sidebar with icons for search, star, and user. The main area is titled 'Learn. Explore. Connect.' and contains three buttons: 'VIDEOS' (with a play icon), 'TOUR' (with a compass icon), and 'COMMUNITY' (with a people icon). At the bottom, a large green 'Get started' button with the subtext 'Start creating your dashboard.' is visible, along with the 'IBM' logo.

When the installation is complete,
Dashboard Application Services Hub is
running

Dashboard Application Services Hub

When the installation is complete, Dashboard Application Services Hub is running. You can log in as the administrative user, **smadmin**. However, the Web GUI components are not installed yet.

Installing Web GUI

ObjectServers must be running

Web GUI installation adds users and groups to the ObjectServers

1. Retrieve installation file

2. Expand file into a temporary directory

3. Start installation launchpad

```
cd <tmp install>/im.linux.x86_64  
./userinst
```

This command installs IBM Installation Manager and Web GUI

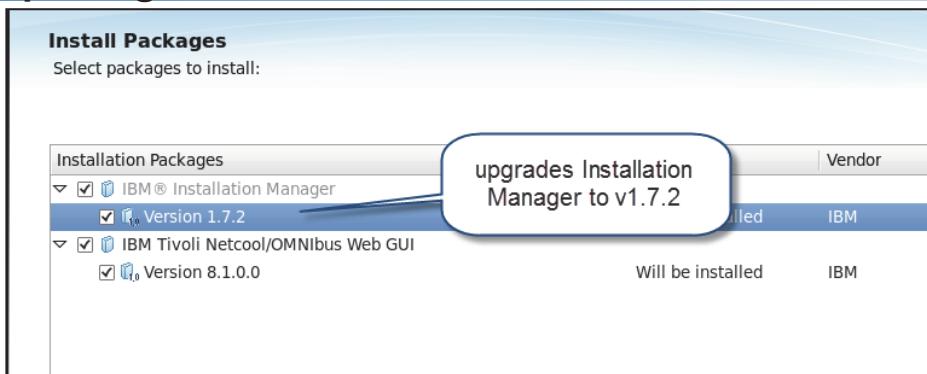
© Copyright IBM Corporation 2014

Installing Web GUI

After Jazz for Service Management, Dashboard Application Services Hub and WebSphere are installed you can install Web GUI. The Web GUI component is distributed in a separate installation file. You must retrieve this file from Passport Advantage® and expand the file into a temporary directory. You install Web GUI by first installing IBM Installation Manager. When Jazz for Service Management is installed, the installation process also installs a copy of IBM Installation Manager. However, the Web GUI installation requires a newer version of IBM Installation Manager. So the Web GUI installer installs a newer version of IBM Installation Manager and then uses that to install Web GUI.

Ideally you want all target ObjectServers running when Web GUI is installed. A utility that runs at the end of the installation process connects to the running ObjectServers and adds objects. If the ObjectServers are not available at the time of installation, you can disable the utility at the end of the installation. You can run the utility manually after the installation is complete and the ObjectServers are available.

Install packages



The Jazz for Service Management installation installed IBM Installation Manager v1.6

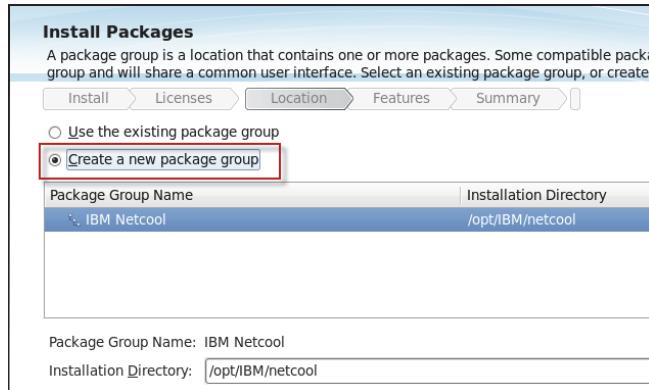
Web GUI installation requires IBM Installation Manager v1.7.2

© Copyright IBM Corporation 2014

Install packages

The screen capture shows the version of IBM Installation Manager that is bundled with the Web GUI installation files.

Package group



Create a new package group

Web GUI cannot be installed into the same package group as Jazz for Service Management

Web GUI installation directory can be changed

© Copyright IBM Corporation 2014

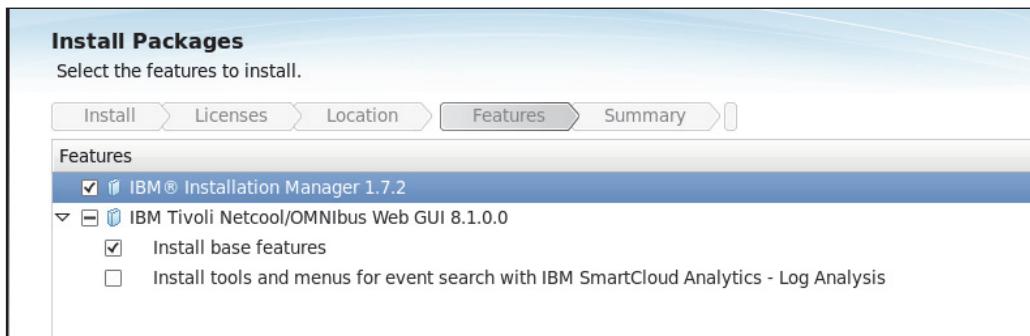
Package group

The package group is used by IBM Installation Manager to organize various products. Some products can be installed into a common package group and others cannot. A package group exists for Jazz for Service Management. However, Web GUI cannot be installed into the same group. The default option is to create a new group.



Hint: If you happen to change the option to use an existing group, an error message is displayed that indicates that you cannot use the existing group.

Package features



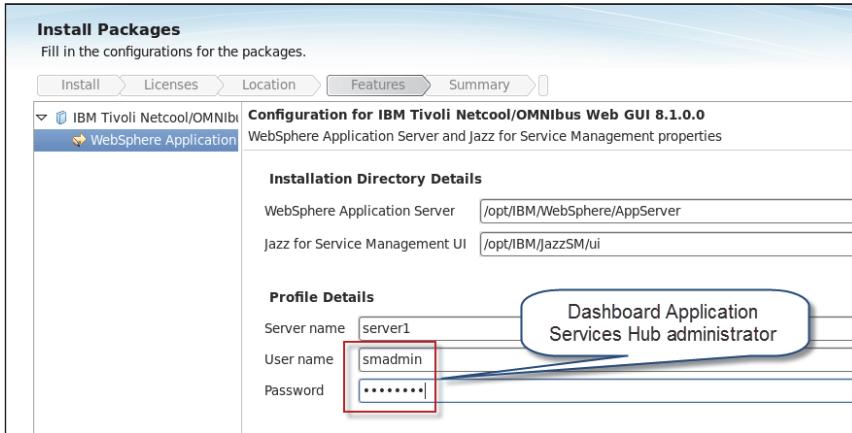
The option is available to install the integration for IBM SmartCloud Analytics Log Analysis

© Copyright IBM Corporation 2014

Package features

When you installed Netcool/OMNIBus core in a previous exercise, there were a number of features available for installation. For the Web GUI, there are currently only two choices. The default is to install the base features. There is an option to install tools and menus that are used with IBM SmartCloud Analytics Log Analysis. You select this feature only if you are using that product in your environment.

Administrator credentials



The administrator user is defined when the Dashboard Application Services Hub component is installed

The installer verifies that the credentials are valid and starts Dashboard Application Services Hub if necessary

© Copyright IBM Corporation 2014

Administrator credentials

The installation process connects to Dashboard Application Services Hub to install the Web GUI components. You must provide the administrator user ID and password. The installer verifies that the credentials are valid and starts Dashboard Application Services Hub if necessary.

Installation summary

Install Packages
Review the summary information.

Install Licenses Location Features Summary

Target Location

Package Group Name: IBM Netcool
Installation Directory: /opt/IBM/netcool
Shared Resources Directory: /opt/IBM/IMShared

Packages

Packages
IBM® Installation Manager 1.7.2
IBM Tivoli Netcool/OMNibus Web GUI 8.1.0.0
Install base features

Click **Install** to start the installation

Typical installation runs approximately 15 minutes

© Copyright IBM Corporation 2014

Installation summary

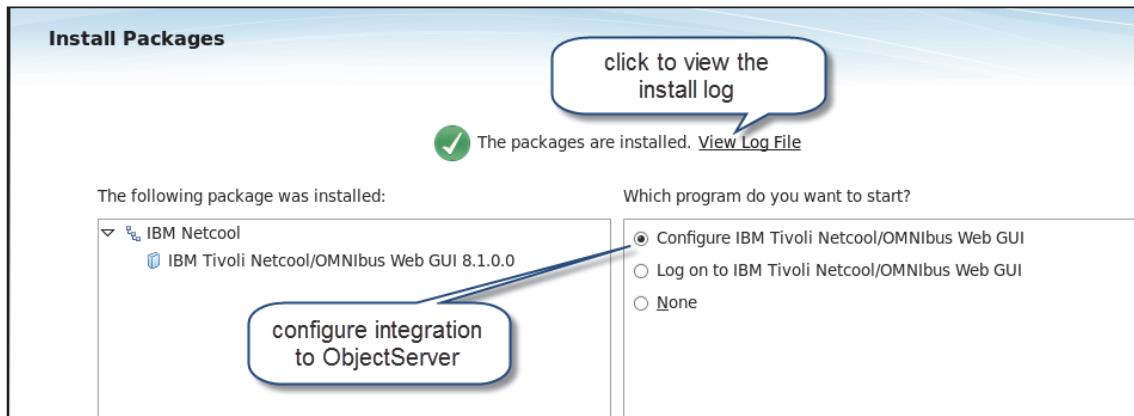
The installer provides a summary of components.



Note: A typical installation runs for approximately 15 minutes.

Installation complete

- Successful installation
- Review log for issues



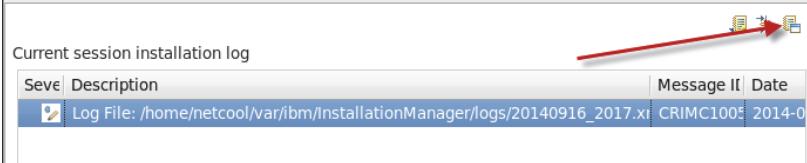
© Copyright IBM Corporation 2014

Installation complete

When the installation is complete, you can view the log file by clicking the link as shown in this screen capture.

The option to configure the integration to the ObjectServers is selected by default. When this option is selected, a utility is started. This utility must have access to running ObjectServers. If the ObjectServers are not available, change the option to **None**.

Installation log file



The screenshot shows a window titled "Current session installation log". It contains a table with columns: Severity, Description, Message ID, and Date. A red arrow points to the "Description" column of the first row, which displays the path "Log File: /home/netcool/var.ibm/InstallationManager/logs/20140916_2017.xml".

Severity	Description	Message ID	Date
INFO	Log File: /home/netcool/var.ibm/InstallationManager/logs/20140916_2017.xml	CRIMC1005I	2014-09-16 20:17:47

Install Log

Started at: 2014-09-16T20:17:47+00:00

Level	Message ID	Time	Message
1	INFO	00:00:08	No log properties file found in: /tmp/ciclogs_netcool
2	INFO	00:00:09	Initial log file: /tmp/ciclogs_netcool/20140916_2017.xml
3	INFO	00:02:37	Store does not exist. Creating a new store...
4	INFO	00:02:64	OS Windowing System: gtk 2.18.9
5	INFO	00:02:67	No log properties file found in: /tmp/ciclogs_netcool
6	NOTE	CRIMC1005I	Log File: /home/netcool/var.ibm/InstallationManager/logs/20140916_2017.xml
7	INFO	00:02:68	No log properties file found in: /home/netcool/var.ibm/InstallationManager/logs
8	INFO	00:02:68	Starting Installation Manager.
9	INFO	00:02:68	osgi.os:linux, osgi.arch:x86_64, 64bit
10	INFO	00:02:68	os.name:Linux, osgi.nl:en_US, java.version:1.6.0
11	INFO	00:02:68	user.name:netcool, non-administrator
12	INFO	00:02:68	Default encoding: UTF-8
13	INFO	00:02:68	Command line arguments: -toolId userinst -accessRights nonAdmin input @osgi

© Copyright IBM Corporation 2014

Installation log file

The screen capture shows a sample of the installation log file output.

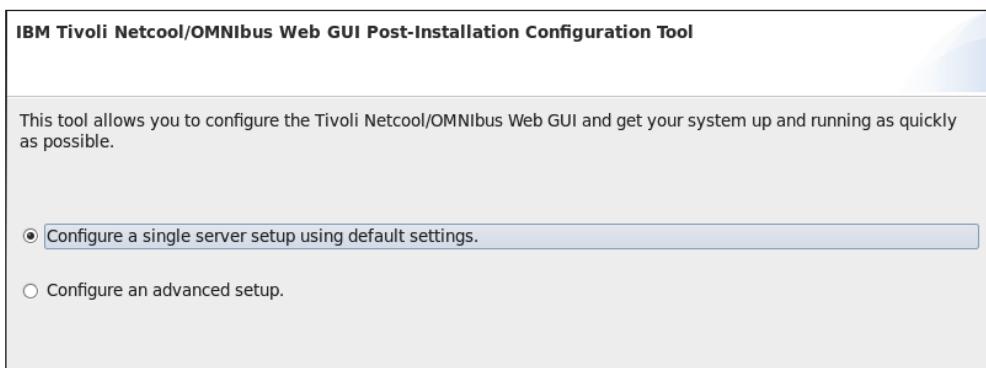
Postinstallation configuration

Launched at the end of the installation

Can be run manually

```
cd /opt/IBM/netcool/omnibus_webgui/configtool/linux.gtk.x86_64  
.ncwConfigUI -WASUserID <admin user> -WASPassword <password>
```

Use advanced setup for ObjectServer high availability



© Copyright IBM Corporation 2014

Postinstallation configuration

If you select the postinstallation utility, it starts at the end of the installation. Otherwise, you can run it manually as follows:

```
cd /opt/IBM/netcool/omnibus_webgui/configtool/linux.gtk.x86_64  
.ncwConfigUI -WASUserID <admin user> -WASPassword <password>
```

You can use the utility to configure a single ObjectServer or a high-availability pair of ObjectServers.

Postinstallation steps

Define ObjectServer access data

- Host
- Port
- User ID
- Password

IBM Tivoli Netcool/OMNIBus Web GUI Post-Installation Configuration Tool

The following ObjectServer will be configured as the default authentication provider in IBM Dashboard Application Services Hub. It will also configure a data source called "OMNIBUS".
The default OMNIBus users and groups will be created.

ObjectServer

Host:	<input type="text" value="host1.tivoli.edu"/>	Port:	<input type="text" value="4100"/>
-------	---	-------	-----------------------------------

Authentication

User ID:	<input type="text" value="root"/>	Password:	<input type="password" value="....."/>
----------	-----------------------------------	-----------	--

© Copyright IBM Corporation 2014

Postinstallation steps

The utility requires the access information for the ObjectServer, including these credentials:

- The server that is hosting the ObjectServer
- The TCP/IP port number that the ObjectServer uses for connections
- An ObjectServer user ID and password

Configuration summary

Define ObjectServer data source

Create two users

Create two groups

IBM Tivoli Netcool/OMNibus Web GUI Post-Installation Configuration Tool

You are about to configure the following as the default authentication provider in IBM Dashboard Application Services Hub:

- **ObjectServer Repository**
- host1.tivoli.edu : 4100 (Primary Server)
- root (User)

You have opted to create the following users and groups:

Default Users (2)

- ncoadmin
- ncouser

Default Groups (2)

- Netcool_OMNIbus_Admin
- Netcool_OMNIbus_User

The system will also configure a data source called **OMNIBUS**

Click Next to continue.

© Copyright IBM Corporation 2014

Configuration summary

The postinstallation utility creates a Web GUI data source definition. A Web GUI data source represents an ObjectServer. In addition, the utility creates two sample users, **ncoadmin** and **ncouser**. It also creates two sample groups: **Netcool_OMNIbus_Admin** and **Netcool_OMNIbus_User**. The groups are defined with the Web GUI roles that are required to provide administrative authority or normal user authority.

When the utility is complete, you can log in to Dashboard Application Services Hub as either user. The password that is initially assigned to each user is the same password that is used for the administrative user, **smadmin**. You can manually change the password values later.

The Web GUI data source definition is contained in the following XML file:

`/opt/IBM/netcool/omnibus_webgui/etc/datasources/ncwDataSourceDefinitions.xml`

The utility creates this file with the information that was provided when the utility is run. You can manually edit this file and change the information, if necessary. If you manually edit this file, you must stop and start Dashboard Application Services Hub to activate the changes.

Postinstallation steps

Define aliases

- Dashboard Application Services Hub, and Web GUI start during the installation
- UNIX/Linux aliases are convenient for managing Dashboard Application Services Hub

Register Java plug-in

- Make Java plug-in available to user browsers

Verify functions

© Copyright IBM Corporation 2014

Postinstallation steps

The Dashboard Application Services Hub server starts as a part of the installation. The server runs as the same user who installed Web GUI. It does not have to be the *root* UNIX/Linux user.



Important: You do not start or stop Web GUI. Web GUI is contained within Dashboard Application Services Hub. So, you start and stop Dashboard Application Services Hub, not Web GUI.

After the installation completes, you must complete a number of steps.

Define aliases

There are commands to start, stop, and run a status check of the Dashboard Application Services Hub server. The command syntax is long. On UNIX or Linux servers, it is convenient to create a set of aliases to provide shortcuts for these commands.

Register Java plug-in

The Web GUI requires access to a specific Java plug-in. It requires this step to identify the location for user browsers.

UNIX/Linux aliases

Define the following aliases to help manage components

```
alias JAZZ_START="/opt/IBM/JazzSM/profile/bin/startServer.sh server1"  
alias JAZZ_STOP="/opt/IBM/JazzSM/profile/bin/stopServer.sh server1 -username smadmin -password object00"  
alias JAZZ_STATUS="/opt/IBM/JazzSM/profile/bin/serverStatus.sh server1 -username smadmin -password object00"
```

© Copyright IBM Corporation 2014

UNIX/Linux aliases

Configure the following alias definitions to provide command shortcuts to manage the Dashboard Application Services Hub server:

```
alias JAZZ_START="/opt/IBM/JazzSM/profile/bin/startServer.sh server1"  
alias JAZZ_STOP="/opt/IBM/JazzSM/profile/bin/stopServer.sh server1 -username smadmin -password object00"  
alias JAZZ_STATUS="/opt/IBM/JazzSM/profile/bin/serverStatus.sh server1 -username smadmin -password object00"
```

These definitions are required only for the UNIX or Linux user that starts and stops the Dashboard Application Services Hub server. Notice that the **smadmin** password shows in clear text in the definition. For security purposes, you can place these definitions in a file that is only accessible by the required user. Another other option is to leave the password out of the definition. When the users run the command, it prompts them for the password.

Register Java plug-in

Identifying the Java plug-in varies by user workstation

- Windows: Use the Tools option in the browser
- UNIX/Linux: Create logical link to the plug-in location

Example in Firefox 64-bit:

```
cd /usr/lib64/firefox/plugins
ln -s
/opt/IBM/tivoli/netcool/platform/linux2x86/jre64_1.7.0/jre/lib/amd64/libnpjp2.so
```

© Copyright IBM Corporation 2014

Register Java plug-in

The server that the student exercises use is based on Red Hat Linux. The exercises use the Firefox browser on that server. So that Firefox can locate the required plug-in, you must run the following commands as the **root** user:

```
$cd /usr/lib64/firefox/plugins
ln -s
/opt/IBM/tivoli/netcool/platform/linux2x86/jre64_1.7.0/jre/lib/amd64/usr/libnpjp
p2.so
```

This step is necessary only if a user wants to use the Firefox browser on Red Hat Linux to access Dashboard Application Services Hub. In a production environment, most users are likely using Windows desktops.

Dashboard Application Services Hub access

<http://<hostname>:16310/ibm/console>



© Copyright IBM Corporation 2013

Dashboard Application Services Hub access

When the installation is complete, the Dashboard Application Services Hub server is running.

Open a browser session to <http://<hostname>:16310>. The server redirects the browser to <https://<hostname>:16311/ibm/console/logon.jsp>.

Enter the administrator user name and password that you specified during the installation. The default user is **smadmin**. You can also use either of the default user IDs that the postinstallation utility created.

Student exercises



Complete the steps for Exercise 1 Installing Netcool/OMNIbus Web GUI

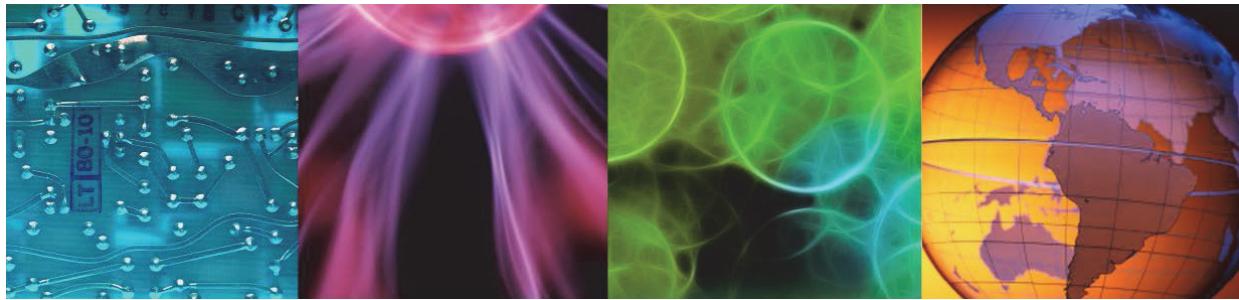
© Copyright IBM Corporation 2014

Student exercises

Lesson 2 Web GUI user features and functions



Lesson 2 Web GUI user features and functions

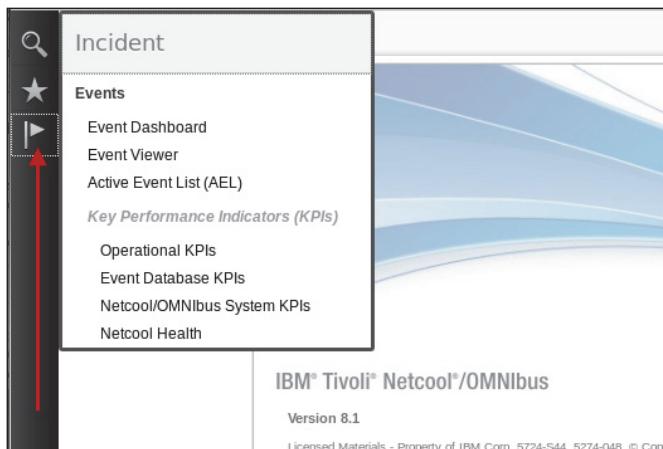


© Copyright IBM Corporation 2014

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

In this lesson, you learn about the Web GUI features associated with a normal user. After completing this lesson, you should be able to describe the Web GUI features available to a normal user.

Web GUI user authority



Granted through *ncw_user* role

Assigned to Netcool_OMNIBus_User group during installation

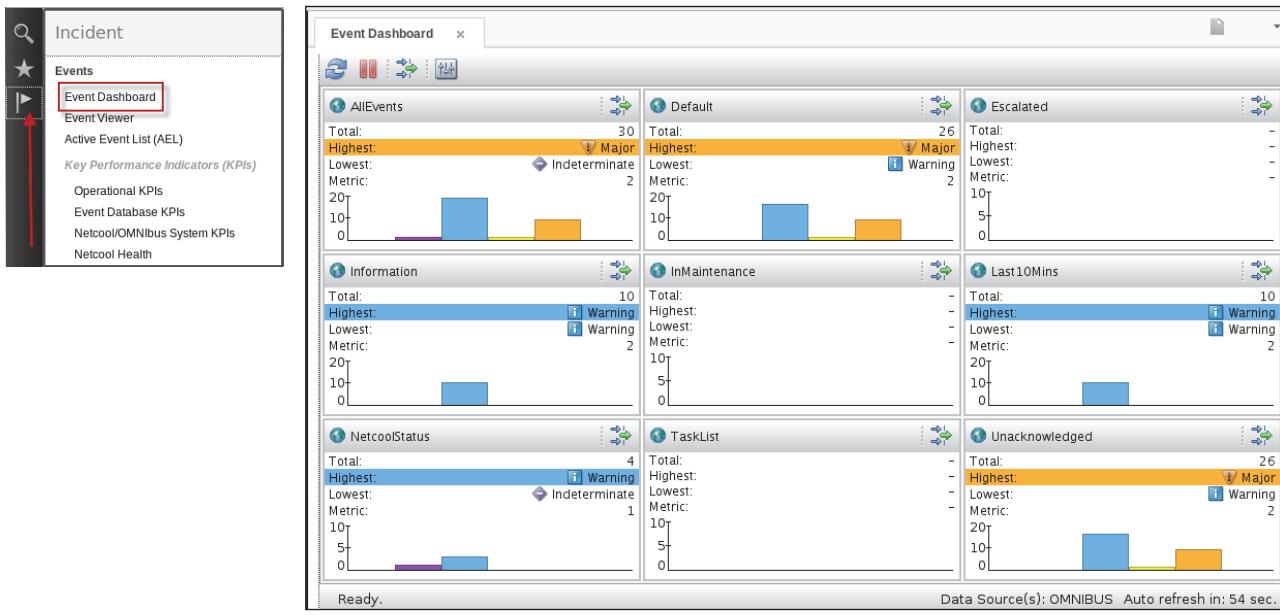
© Copyright IBM Corporation 2014

Web GUI user authority

A user is considered a *normal* Web GUI user if they are assigned the *ncw_user* role. The Dashboard Application services Hub pages that are shown in the screen capture are available to users with the *ncw_user* role. The postinstallation utility associates the *ncw_user* role with the Netcool_OMNIBus_User group.

The following slides provide examples of each of these pages.

Event Dashboard



Event Dashboard

Selecting the Event Dashboard feature produces this view. The Event Dashboard is functionally equivalent to the Native Desktop Event List. Each box in the window results from a *filter* definition. The configuration that is shown is the default one that comes with Web GUI. A user with the *ncw_dashboard_editor* role can add, change, or drop individual monitor boxes from this view. Observe the two areas on the bottom, right corner of this page: Data sources and automatic refresh.

Data sources

The Web GUI does not use the Netcool/OMNIbus interfaces file. The text in this box is part of the data source property file that is created by the postinstallation utility. When ObjectServer high availability is configured, a single data source is defined to Web GUI. The data source definition that is associated with that name contains the access information for the primary and backup ObjectServers. The events that are shown in the Event Dashboard are coming from *one* of the ObjectServers. When the primary is active, they come from the primary.

If the primary fails, the Web GUI retrieves events from the backup. It is not really a failover or fallback scenario as a probe or desktop behaves. Probes and desktops are only physically connected to one ObjectServer at a time. The Web GUI maintains connections to both ObjectServers. It merely retrieves events from one or the other based on availability.

The Web GUI can connect to multiple ObjectServers, in addition to the high-availability capability. Some companies run multiple ObjectServers in a geographically distributed configuration, for example, US, EMEA, APAC. In each instance, there can be high availability pairs of ObjectServers,

for example, US_P and US_B. You can configure the Web GUI to connect to all three sets of ObjectServers. When configured in this manner, the Web GUI dashboard can contain events from any or all of the ObjectServers.

Automatic refresh

The contents of the window update periodically, every 60 seconds by default. The automatic refresh provides a count down to when the next refresh occurs.

Event Dashboard, continued

The screenshot shows the Event Dashboard interface. On the left, there are three monitoring panes: 'AllEvents' (Total: 30, Highest: Major), 'Information' (Total: 10, Highest: Warning), and 'NetcoolStatus' (Total: 4, Highest: Warning). A red arrow points from the 'AllEvents' pane to the 'Active Event List' table on the right. The table has columns: Sev, Ack, Node, Alert Group, and Summary. It lists 30 events, mostly of level 'Warning'. The summary column includes entries like 'Machine has gone offline', 'Link Down on port', and 'Diskspace alert'.

© Copyright IBM Corporation 2014

Event Dashboard, continued

Each box in the dashboard pane is an HTML hyper-link. When you click any box, the corresponding Active Event List page opens.

Hint: The Active Event List opens by default when a monitor boxed is clicked. You can change the behavior to open the Event Viewer instead.

Active Event List

Sev	Ack	Node	Alert Group	Summary
!	No	Sydney	Information...	Shift+I ine has gone offline
!	No	London	Journal...	Ctrl+J Down on port
!	No	link3	Quick Filter	> ine has gone offline
!	No	Washington	Link	Link Down on port
!	No	Moscow	Stats	Diskspace alert
!	No	link6	Systems	Machine has gone offline
!	No	link2	Link	Link Down on port
!	No	Beijing	Stats	Diskspace alert
!	No	host1.tivoli.edu	ClientStatus	Time for all clients in granularity
!	No	host2.tivoli.edu	TIP_NCOS_VM...	A TIP_NCOS_VMM_PRIMARY pro
!	No	host1.tivoli.edu	MemstoreStatus	table_store soft limit: used 1 MB

Sev	Ack	Node	Alert Group	Summary
!	No	Sydney	Acknowledge	Ctrl+A gone offline
!	No	London	De-acknowledge	Ctrl+D gone offline
!	No	link3	Prioritize	> 1 port
!	No	Washington	Suppress/Escalate	> gone offline
!	No	link4	Take ownership	> 1 port
!	No	Beijing	User Assign	> ert
!	No	Beijing	Group Assign	> gone offline
!	No	Beijing	Delete	> 1 port
!	No	Beijing	Ping	> ert
!	No	host1.tivoli.edu	Telnet	> :clients in granularity period (60
!	No	host2.tivoli.edu	Information...	Shift+I VMM_PRIMARY process runnir
!	No	host1.tivoli.edu	Journal...	Ctrl+J oft limit: used 1 MB of capacity
!	No	host1.tivoli.edu	Quick Filter	>

Dashboard Application Services Hub roles (*netcool_ro*, *netcool_rw*) restrict access to tools

© Copyright IBM Corporation 2014

Active Event List

Selecting the Active Event List (AEL) feature opens this window.

The AEL provides all of the capabilities of the native desktop. The user can select an event, right-click, and see a menu of available tools. Dashboard Application Services Hub roles determine access to the menus and tools available in the Active Event List. The *netcool_ro* role hides any tool that modifies event data, for example *delete*. The *netcool_rw* role makes these tools visible in the Active Event List.



Important: The Active Event List uses a Java plug-in.

Event Viewer

The screenshot shows the Event Viewer interface. On the left, a sidebar lists 'Incident' and 'Events' sections, with 'Event Viewer' highlighted. The main area displays event data grouped by severity (All, Major, Minor, Warning) and location (Moscow, Sydney, link6, link4, Beijing). A callout bubble points to the grouping feature. On the right, a context menu for an event row is open, showing options like 'Acknowledge', 'De-acknowledge', 'Prioritize', 'Suppress/Escalate', 'Take ownership', 'User Assign', 'Group Assign', 'Delete', 'Ping', 'Information ...', 'Copy', and 'Systems'. Another callout bubble points to the 'netcool_rw' role. Below this, another Event Viewer window shows a different set of events with a similar context menu, where a callout bubble points to the 'netcool_ro' role.

Read-only or read-write access is controlled by role
Optionally events can be grouped based on one or more columns
Column names are defined within the View definition

© Copyright IBM Corporation 2013

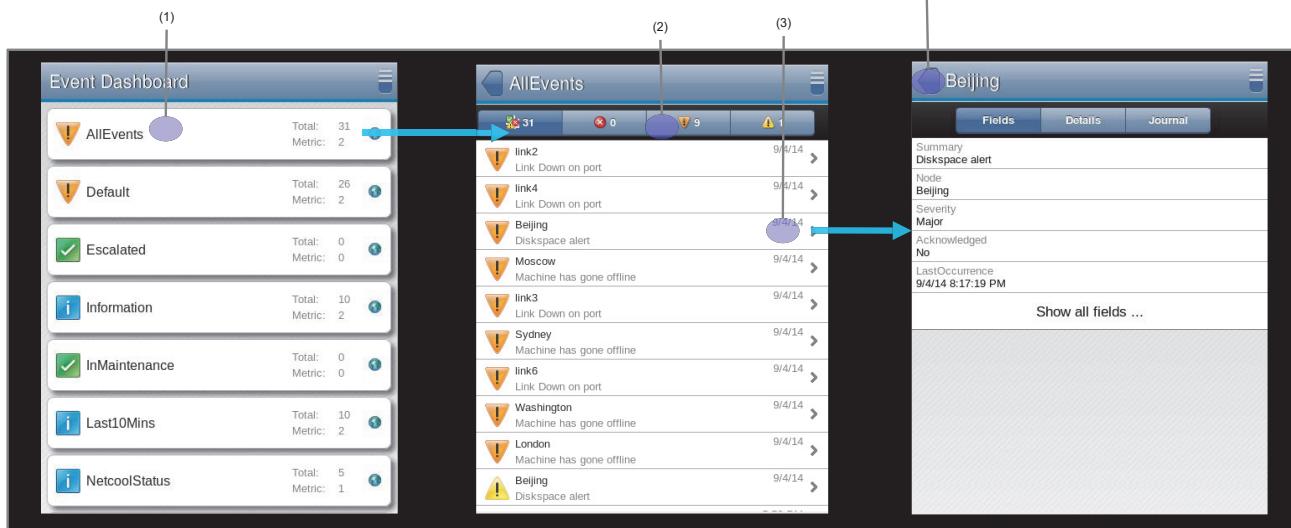
Event Viewer

The Event Viewer provides the same capabilities as the Active Event List. However, the Event Viewer does not use a Java plug-in. Dashboard Application Services Hub roles determine access to the menus and tools available in the Event Viewer. The *netcool_ro* role hides any tool that modifies event data, for example, *delete*. The *netcool_rw* role makes these tools visible in the Event Viewer.

The Event Viewer also provides the option to group events in a hierarchical manner. The grouping feature is configured within the view definition. More information about views, and grouping is provided in the student exercise.

Mobile access to event list

- (1) Tap monitor box to view mobile event list
- (2) Tap to filter events by top three severity levels
- (3) Tap to view event details and journals
- (4) Tap to return



© Copyright IBM Corporation 2013

Mobile access to event list

Access to event data is available on certain mobile devices. The access is based on the same format as the Event Dashboard. The first panel of the screen capture that is shown is functionally equivalent to the Event Dashboard described previously. The second panel represents what happens when the user drills down into the event detail that is associated with an Event Dashboard box. And the last panel represents the event details that are associated with drilling into one event.

The last panel shows the data for a few event record columns. Recall that the ObjectServer event record contains over 100 unique column names by default. The column names in the last panel are referred to as the *important columns*. The Web GUI administrator defines which columns are considered the *important columns*. The column names are defined in the following file:

/opt/IBM/tivoli/netcool/omnibus_webgui/etc/server.init

The default definition is as follows:

```
eventlist.information.fields.important:Summary, Node, Severity, Acknowledged,  
LastOccurrence
```

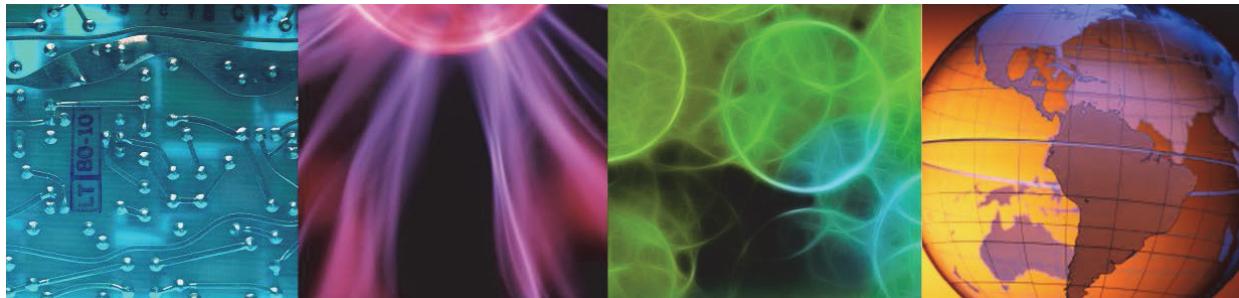


Important: Mobile access to the event data is not dependent on a custom mobile application. Access is provided through any supported browser on the mobile device.

Lesson 3 Web GUI administrative features and functions



Lesson 3 Web GUI administrative features and functions



© Copyright IBM Corporation 2014

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

In this lesson, you learn about some of the basic Web GUI administrative tasks. More detail about Web GUI administration is provided in a subsequent unit. After completing this lesson, you should be able to describe some of the basic Web GUI administrative tasks.

Typical Web GUI administration

- Create groups
- Create users
- Create views
- Create filters
- Create tools
- Create menus
- Create maps
- Create pages

Some of these tasks require Web GUI administrative authority

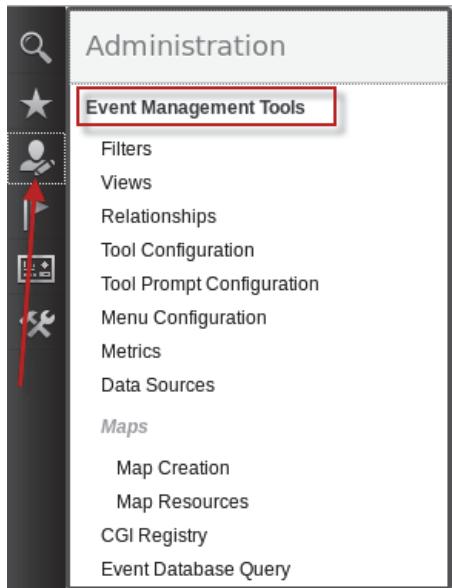
Some of these require Dashboard Application Services Hub administrative authority

© Copyright IBM Corporation 2014

Typical Web GUI administration

This list represents some of the basic Web GUI administrative tasks.

Web GUI administrative authority



Granted through *ncw_admin* role

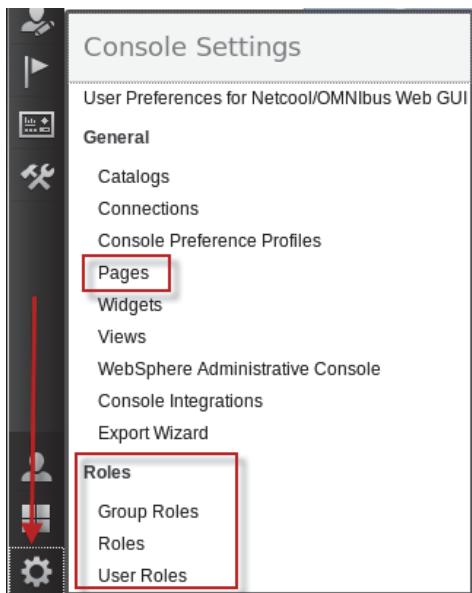
Assigned to Netcool_OMNibus_Admin group
during installation

© Copyright IBM Corporation 2014

Web GUI administrative authority

The *ncw_admin* role grants access to the Web GUI administrative features. The postinstallation utility associates this role with the Netcool_Admin_User group.

Dashboard Application Services Hub administrative authority



Granted through *iscadmins* role

Assigned to **ncoadmin** user during installation

Dashboard Application Services Hub administrative authority

The *iscadmins* role grants access to the Dashboard Application Services Hub administrative features. The postinstallation utility associates this role with the *ncoadmin* user.

Creating groups and users

WebSphere Administrative Console

Must use *smadmin* user ID

© Copyright IBM Corporation 2014

Creating groups and users

Groups and users are created with the WebSphere Administrative Console utility. Dashboard Application Services Hub contains a short-cut that can be used to connect to this utility.



Important: You must use the *smadmin* user ID to log in to Dashboard Application Services Hub to use the short-cut to WebSphere Administrative Console.

Creating a group

The first screenshot shows the main navigation menu with 'Manage Groups' selected. The second screenshot shows the 'Manage Groups' page with a search bar and a 'Create...' button highlighted. The third screenshot shows the 'Create a Group' dialog box where 'Operations' has been entered into the 'Group name' field, and a red arrow points to the 'Create' button at the bottom.

Create new group:

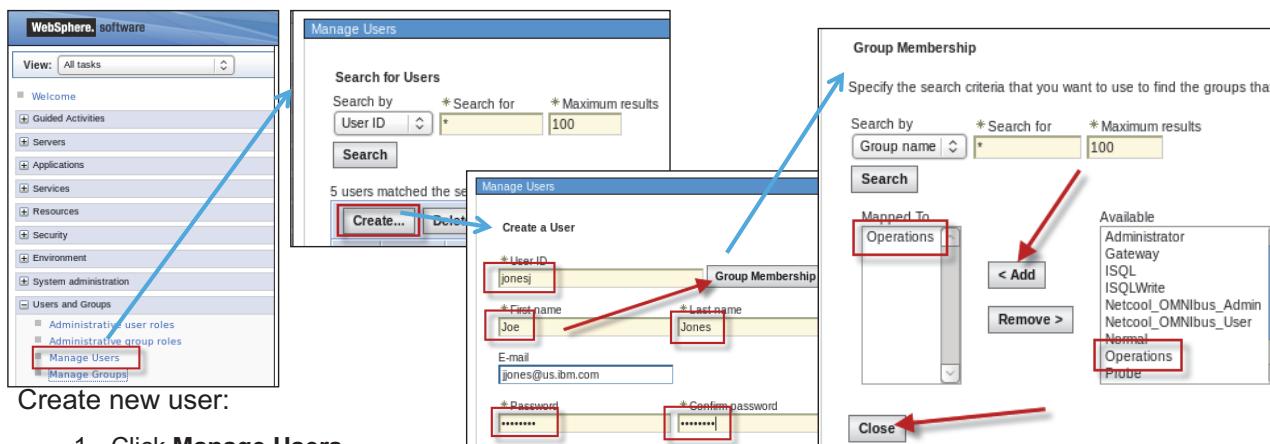
1. Click **Manage Groups**
2. Click **Create**
3. Enter a group name, and click **Create**

© Copyright IBM Corporation 2014

Creating a group

Within the WebSphere Administrative Console user interface, you expand the option for **Users and Groups**. Then you click **Manage Groups**. A new window opens that contains a list of currently defined group names. You click **Create** and a new window opens. You enter a name for the group and an optional description. Group names cannot contain spaces or special characters except for the underscore character. Click **Create** to create the group.

Creating a user



Create new user:

1. Click **Manage Users**
2. Click **Create**
3. Enter user ID, name, and password
4. Click **Group Membership**, assign groups, and click **Close**
5. Click **Create** to create the user

© Copyright IBM Corporation 2014

Creating a user

The process to create a user is similar to creating a group. Within the WebSphere Administrative Console user interface, you expand the option for **Users and Groups**. Then you click **Manage Users**. A new window opens that contains a list of currently defined user names. You click **Create** and a new window opens. You enter a user name, which is the user ID that is used to log in. You also enter a first and last name and a value for the password. Next, you click **Group Membership**. A new window opens that lists all of the currently defined group names. You select a group name from the list and click **Add** to assign the user to the corresponding group. You repeat these steps to add the user to more groups if necessary. You click **Close** to close the group membership window. Click **Create** to create the user.

Assigning roles to a group

Define group with read-write access:

1. Click **Group Roles**
2. Locate and select a group
3. Assign roles and click **Save**

© Copyright IBM Corporation 2014

Assigning roles to a group

The process to create a group or user involves two steps. In the first step, you use WebSphere Administrative Console to create the group or user. In the second step, you use Dashboard Application Services Hub to assign one or more roles to the group or user.

To assign roles to a group, you select **Group Roles**. A new window opens and you can search for a specific group name or list all available groups. Each group name is an HTML hyper-link. You click the group name and a new window opens. This window contains a list of the available roles. You click one or more roles to select them and then click **Save**.

The screen capture demonstrates how to define a group with read-write event access. Assigning roles to groups provides a convenient mechanism for defining role-based access by functional areas.

Creating a view

The screenshot illustrates the steps to create a new view:

- Step 1:** In the 'Administration' sidebar, under 'Event Management Tools', click 'Views'.
- Step 2:** In the 'Views' list, click the '+' icon to start creating a new view.
- Step 3:** In the 'New View' dialog, set the 'Public' type to 'global' (checked).
- Step 4:** In the 'Edit View: New View' dialog, enter the name 'MyNewView'.
- Step 5:** Under 'Available fields', select 'Acknowledged' and click the right arrow button to move it to the 'Display Columns' tab.

© Copyright IBM Corporation 2014

Creating a view

A **view** determines how the event data is shown. The view defines which column names, the order of appearance, the column heading, and others. The **Sort Columns** tab defines which column names are used as the sort values. The sort values determine the order of the event records in the window. The **Group Columns** tab defines the event grouping criteria. The grouping criteria determines how the event records are grouped in the Event Viewer desktop.

The Web GUI administrator defines which columns are available for use in the grouping criteria. The configuration is defined in the following file:

```
/opt/IBM/tivoli/netcool/omnibus_webgui/etc/server.init
```

Two property settings control event grouping. The first property defines the maximum number of column names allowed. The default setting appears as follows:

```
columngrouping.maximum.columns:3
```

The second property setting provides a list of column names to use as grouping criteria. The default setting is shown as follows:

```
columngrouping.allowedcolumns=Acknowledged,AlertGroup,Class,Customer,Location,Node,NodeAlias,NmosCauseType,NmosManagedStatus,Severity,Service
```

Creating a filter

The screenshot shows three windows illustrating the creation of a filter:

- Left Window (Administration):** Shows the "Event Management Tools" section with "Filters" selected. A red box highlights the "Filters" icon in the sidebar.
- Middle Window (Filters):** Shows the "Available filters" list. A red box highlights the "New Filter" button.
- Right Window (New Filter):** A modal dialog titled "New Filter".
 - Public:** "global" checkbox is checked.
 - User:** "Name:" field contains "MyNewFilter".
 - Default view:** "MyNewView" is selected.
 - Description:** Empty.
 - Data Source:** "Click to show".
 - Filter Conditions:** "Basic" tab is selected. A blue box highlights the "Filter criteria" section. A "Severity" field has a value of "5".

A red arrow points from the "New Filter" button in the middle window to the "New Filter" dialog in the right window. A blue box labeled "View" surrounds the "MyNewView" dropdown in the User section of the right dialog.

Creating a filter

1. Select type
2. Enter name
3. Select view
4. Define filter criteria

Filters are SQL WHERE clauses that restrict event records to a specific criteria. Every filter definition creates a new monitor box in the Event Dashboard. More details about *filters* are provided in the student exercises.

Using a filter and view

The screenshot shows two windows side-by-side. On the left is the 'Event Dashboard' window, which has a sidebar with icons for Incident, Events, Key Performance Indicators (KPIs), and Netcool Health. The 'Events' section contains links for 'Event Dashboard' (which is highlighted with a red box and has a blue arrow pointing from the sidebar to it), 'Event Viewer', 'Active Event List (AEL)', and 'Key Performance Indicators (KPIs)'. On the right is the 'Event Dashboard...' window, which displays various event metrics and a table of active events. A new filter named 'MyNewFilter' has been created and is visible in the dashboard. A new view named 'MyNewView' has also been created and is associated with this filter. A blue callout bubble labeled 'new filter' points to the 'MyNewFilter' entry in the dashboard, and another blue callout bubble labeled 'new view' points to the 'MyNewView' entry in the same area.

New filters are immediately available for use

Automatically appear in the Event Dashboard

The view is associated with the filter

© Copyright IBM Corporation 2014

Using a filter and view

Every filter definition creates a new monitor box in the Event Dashboard. When you click the monitor box, the Active Event List opens. The event records are restricted based on the *filter* definition. The event column names that are listed are based on the corresponding *view* definition.

Creating a map

Create a map

1. Enter name
2. Define background image
3. Add active icons, and define properties: filter, view, action
4. Add text

© Copyright IBM Corporation 2014

Creating a map

A map provides access to event data in the context of a graphical presentation. For example, a map can contain a picture of the United States. At various places on the picture, icons indicate the status of event records that are associated with the corresponding geographical reference. Each icon is defined with two primary characteristics:

- Filter

The filter provides the criteria that determines which event records are related to the icon.

- Action

The action defines what happens when the user clicks the icon. The action is typically to open an event display, such as the Active Event List or Event Viewer.

Adding a map to a page

The screenshot illustrates the steps to add a map to a page:

- Console Settings:** Shows the 'Pages' option selected in the sidebar.
- Pages Management:** Shows the 'New Page...' button highlighted.
- Page Settings:** Shows the 'Page name' field set to 'MyNewPage' and the 'Page location' set to 'console/Default/'. The 'Freeform' layout option is selected.
- Widget Selection:** Shows the 'Map' icon highlighted among other widget options like Hotspot, Image Widget, and List.
- Edit Properties:** Shows the 'Edit properties' option for a Map portlet on a page, with the 'Edit' button highlighted.

Create a page, which requires *iscadmin* role

1. Enter name
2. Define page location (optional)
3. Locate portlet, and drag to page
4. Edit portlet properties

Adding a map to a page

Adding a map to a page is another area where Web GUI administration, and Dashboard Application Services Hub administration overlap. A map is a Web GUI component that requires Web GUI administrative access to be created or modified. The map cannot be accessed unless it is placed in a Dashboard Application Services Hub page, which is a Dashboard Application Services Hub administrative function.

To add a map to a page, you start by creating a Dashboard Application Services Hub page. The page management view provides a list of widgets. You click the Map widget, and drag it into the page. Then you edit the widget properties.

Adding a map to a page, continued

The screenshot shows two windows side-by-side. On the left is the 'Map - Edit Preferences' dialog, which has a 'General Settings' section with a 'Map name:' field containing 'Example_Europe'. Below it are checkboxes for 'Sound URL:', 'Refresh rate (in seconds):', 'Enable hover help for active objects:', and 'Show status bar:'. On the right is the 'MyNewPage' dashboard titled 'Global Network Status Europe'. It features a map of Europe with status indicators (1 and 2) and three callout boxes in the top right corner: 'Critical' (Total: 0, Highest: Clear), 'Unassigned' (Total: 6, Highest: Major), and 'Last Day' (Total: 0, Highest: Clear). A red box highlights the 'Example_Europe' map name in the preferences dialog, and a red arrow points from the 'MyNewPage' dashboard to the 'MyNewPage' entry in the navigation sidebar.

Adding a map to a page, continued

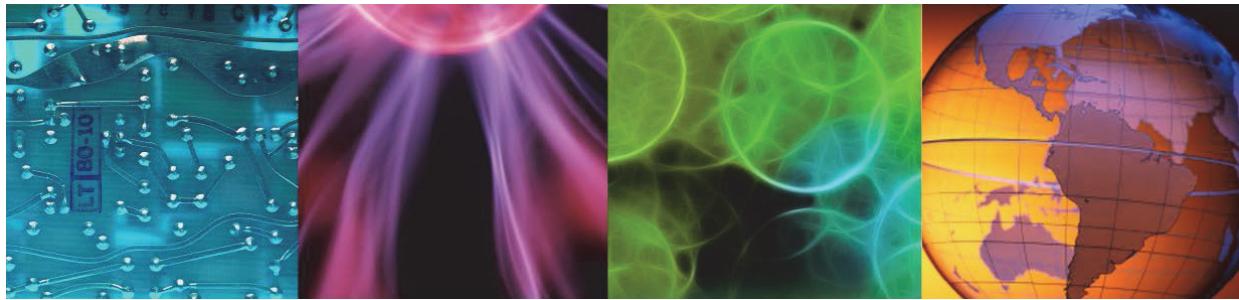
Specify the name of the map in the widget properties definition. After it is saved, the page is available to users.



Lesson 4 Web GUI user authentication



Lesson 4 Web GUI user authentication



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to configure Dashboard Application Services Hub to use an LDAP server as a source of user authentication.

Web GUI user authentication

Web GUI users are defined within Dashboard Application Services Hub **and** the ObjectServer

- Roles determine access to Dashboard Application Services Hub features
- ObjectServer roles determine access to ObjectServer features

Dashboard Application Services Hub is based upon WebSphere Application Server

User authentication within Dashboard Application Services Hub is defined by Virtual Member Manager (VMM)

Three options for VMM sources:

- File-based, internal to Dashboard Application Services Hub
- ObjectServer
- LDAP

The Web GUI default installation configures file-based and ObjectServer

© Copyright IBM Corporation 2014

Web GUI user authentication

User authentication for Dashboard Application Services Hub is provided through the WebSphere Virtual Member Manager (VMM). There are three options for types of VMM repositories:

- Internal file based
- ObjectServer
- LDAP

The administrative user, **smadmin**, is defined in an internal file-based repository. The ObjectServer is the default user repository within the VMM realm. You have the option of replacing the ObjectServer with LDAP as the default user repository. You can use an ObjectServer or LDAP but not both.

ObjectServer user synchronization

The screenshot shows two panels. The left panel is titled 'Manage Users' and displays a search interface for users. The right panel is titled 'Configuration of NYC_... on host1.tivoli.edu:4100' and shows a list of users. Both panels show the same set of users: ncoadmin, ncouser, nobody, and root.

Name	Full Name	Type	ID	External
ncoadmin	ncoadmin tivoli	Normal	1	X false
ncouser	ncouser tivoli	Normal	2	X false
nobody	Nobody	Unknown	65534	X false
root	Root User	Super User	0	X false

smadmin is not added to the ObjectServer because that user does not have ObjectServer roles *ncw_user* or *ncw_admin*

© Copyright IBM Corporation 2014

ObjectServer user synchronization

The default configuration specifies that the ObjectServer is the default user repository for Dashboard Application Services Hub users and groups. With this configuration, a user that is added to Dashboard Application Services Hub is added to the ObjectServer only if the user has one of the Web GUI roles assigned: *ncw_admin* or *ncw_user*. In addition, any user that is created in the ObjectServer is also added to Dashboard Application Services Hub.

User and group entries are not duplicated between Dashboard Application Services Hub and the ObjectServer. The only user entry that is defined within Dashboard Application Services Hub is **smadmin**. The other users are shown because Dashboard Application Services Hub reads the entries from the ObjectServer. If the ObjectServer is not available, the only user in Dashboard Application Services Hub is **smadmin**. When a new user is added to Dashboard Application Services Hub, it is written directly to the ObjectServer.

ObjectServer group synchronization

Netcool_OMNIbus_Admin and **Netcool_OMNIbus_User** created in Dashboard Application Services Hub, and added to ObjectServer

All ObjectServer groups added to Dashboard Application Services Hub

Select	Group name	Description
<input type="checkbox"/>	Administrator	Admin Group
<input type="checkbox"/>	Gateway	Permissions required for a gateway user
<input type="checkbox"/>	ISQL	Read only ISQL access
<input type="checkbox"/>	ISQLWrite	Write ISQL access
<input type="checkbox"/>	Netcool_OMNIbus_Admin	
<input type="checkbox"/>	Netcool_OMNIbus_User	
<input type="checkbox"/>	Normal	Normal Group
<input type="checkbox"/>	Operations	Level 1 operators
<input type="checkbox"/>	Probe	Permissions required for a probe user
<input type="checkbox"/>	Public	Public Group
<input type="checkbox"/>	System	System Group

Configuration of NYC_... on host1.tivoli.edu:4100

- User**
- Groups**
- Roles**

Name /	Description	I
Administrator	Admin Group	2
Gateway	Permissions required for a gateway user	5
ISQL	Read only ISQL access	7
ISQLWrite	Write ISQL access	6
Netcool_OMNIbus_Admin		9
Netcool_OMNIbus_User		8
Normal	Normal Group	3
Operations	Level 1 operators	10
Probe	Permissions required for a probe user	4
Public	Public Group	0
System	System Group	1

ObjectServer group synchronization

With the default configuration, all groups that are created in Dashboard Application Services Hub are also created in the ObjectServer. In addition, any group that is created in the ObjectServer is also added to Dashboard Application Services Hub.

Groups are not duplicated in both Dashboard Application Services Hub and the ObjectServer. They are defined in the ObjectServer and read by Dashboard Application Services Hub.

Web GUI roles

The screenshot shows a list of available Web GUI roles. The roles listed are: monitor, ncw_admin, ncw_dashboard_editor, ncw_gauges_editor, ncw_gauges_viewer, ncw_user, netcool_ro, and netcool_rw. The 'ncw_admin' role is highlighted with a red border.

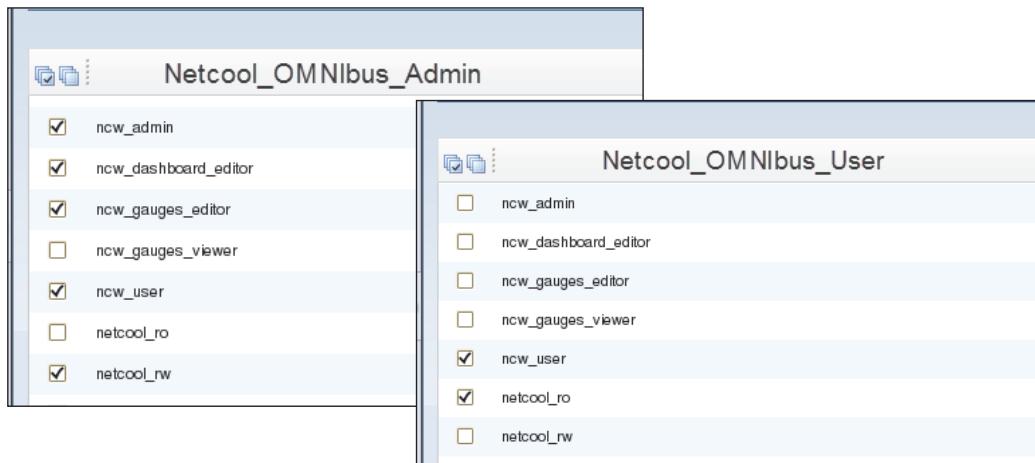
User Roles	
Available Roles	
<input type="checkbox"/>	monitor
<input type="checkbox"/>	ncw_admin
<input type="checkbox"/>	ncw_dashboard_editor
<input type="checkbox"/>	ncw_gauges_editor
<input type="checkbox"/>	ncw_gauges_viewer
<input type="checkbox"/>	ncw_user
<input type="checkbox"/>	netcool_ro
<input type="checkbox"/>	netcool_rw

© Copyright IBM Corporation 2014

Web GUI roles

The screen capture shows all of the available Web GUI roles. The *ncw_admin* and *ncw_user* roles determine whether the user has access to administrative features or not. The *netcool_rw* and *netcool_ro* roles determine which tools are available to the user within the Active Event List or Event Viewer. Refer to the documentation for an explanation of the other roles.

Web GUI group roles



© Copyright IBM Corporation 2014

Web GUI group roles

The Web GUI postinstallation utility creates two groups, Netcool_OMNibus_Admin and Netcool_OMNibus_User. The screen captures highlight the roles that are assigned to each group.

LDAP as an authentication source

LDAP is an option for user authentication

- Can be used for user authentication within Dashboard Application Services Hub
- Can be used for user authentication within the ObjectServer

Decision on how users are authenticated is typically based on the numbers of users that require access to Netcool/OMNIbus

- Small number of users, keep the default option for file-based and ObjectServer
- Large number of users, configure LDAP as the source of authentication

LDAP cannot be configured during Web GUI installation

- It must be manually configured postinstallation

When using LDAP as an authentication source:

- A new LDAP user is automatically created in Dashboard Application Services Hub
- A new user in Dashboard Application Services Hub is automatically added to LDAP

The users are typically limited to those that belong to a specific LDAP subtree

© Copyright IBM Corporation 2014

LDAP as an authentication source

Many companies use an LDAP server as the single point of user administration. Dashboard Application Services Hub provides an option to use an LDAP server as a user repository instead of the ObjectServer. The decision to use LDAP as the authentication source is typically determined based on the number of users that require access to Web GUI. For a limited number of users, it might be more efficient to use the ObjectServer as the default user repository. For many users, it might be more efficient to use LDAP.

LDAP cannot be configured as a user repository during the Web GUI installation. It must be manually configured postinstallation.

When LDAP is configured as the default user repository, all users and groups that are added to LDAP are automatically available in Dashboard Application Services Hub. In addition, all users and groups that are defined in Dashboard Application Services Hub are created in the LDAP server.

Configuring LDAP as an authentication source

The steps to configure user authentication against an LDAP directory are as follows:

- Remove the ObjectServer from the Virtual Member Manager realm
You cannot use ObjectServer and LDAP simultaneously as user repositories
- Add the LDAP directory to the Virtual Member Manager realm
- Configure the Virtual Member Manager realm to write new users to the LDAP directory

The following information is required for the configuration:

- Host name and port number for the LDAP directory
- Type and version of LDAP directory, for example, IBM Security Directory Server V6.2
- The user ID and password that are used to bind to the LDAP server
- Subtree of the LDAP directory that is used for authenticating users

Important: To create users and groups through the Web GUI, the LDAP bind ID must have the appropriate permissions in the LDAP directory

© Copyright IBM Corporation 2014

Configuring LDAP as an authentication source

To use LDAP as a user repository, you must first remove the ObjectServer as the default repository. You cannot have LDAP and the ObjectServer defined simultaneously. Next, you add the LDAP server to the Virtual Member Manager realm. And then you change the realm definition so that new users and groups are written to the LDAP server. All of this configuration is completed with the WebSphere Administrative Console.

Removing the ObjectServer from the Virtual Member Manager realm

The WebSphere Administrative Console is used to remove the ObjectServer definition

The screenshot shows the WebSphere Administrative Console interface. On the left, the navigation tree has 'Security' expanded, with 'Global security' selected. In the center, the 'User account repository' configuration page is displayed. The 'Current realm definition' is set to 'Federated repositories'. Below it, the 'Available realm definitions' section shows 'Federated repositories' selected. A red arrow points from the 'Configure...' button to the 'Remove' button in the 'Repositories in the realm' table. Another red box highlights the 'netcoolObjectServerRepository' entry in the table, which has a checked checkbox next to its base entry.

Select	Base Entry	Repository Identifier	Repository Type
<input type="checkbox"/>	o=defaultWIMFileBasedRealm	InternalFileRepository	File
<input checked="" type="checkbox"/>	o=netcoolObjectServerRepository	NetcoolObjectServer	Custom

© Copyright IBM Corporation 2014

Removing the ObjectServer from the Virtual Member Manager realm

Follow these steps to remove the ObjectServer from the VMM realm:

1. Connect to WebSphere Administrative Console
2. Expand Security
3. Select Global security
4. Locate the User account repository section and click Configure
5. Select the ObjectServer entry and click Remove



Important: The **smadmin** user is defined in the InternalFileRepository. After the ObjectServer is removed, the **smadmin** user is the only remaining user.

Adding LDAP to the Virtual Member Manager realm: Step 1

The WebSphere Administrative Console is used to add the LDAP definition to the realm

The screenshot shows the WebSphere Administrative Console interface. On the left, the navigation tree is expanded under 'Security' to show 'Global security'. The main panel displays the 'User account repository' configuration page. A red arrow points from the 'Configure...' button in the top right of the main panel to the 'Add repositories (LDAP, custom, etc) ...' button in the 'Repositories in the realm' section. The 'Repositories in the realm' section also includes a table with one entry:

Select	Base Entry	Repository Identifier	Repository Type
<input type="checkbox"/>	o=defaultWIMFileBasedRealm	InternalFileRepository	File

Total 1

© Copyright IBM Corporation 2014

Adding LDAP to the Virtual Member Manager realm: Step 1

You add the LDAP server as a new entry in the VMM realm. The screen capture shows the realm definition after the ObjectServer is removed. Click **Add repositories**.

Adding LDAP to the Virtual Member Manager realm: Step 2

Configure the access criteria for the LDAP server

The screenshot shows two configuration panels side-by-side.

General Properties:

- * Repository: A dropdown menu is open, showing "none defined" and "New Repository...". The "New Repository..." option is highlighted with a red arrow.
- * Unique distinguished name: A dropdown menu is open, showing "Custom repository" and "File repository".
- Distinguished name in the URL: This checkbox is unchecked.

LDAP server:

- * Directory type: Set to "IBM Tivoli Directory Server".
- * Primary host name: Set to "host1.tivoli.edu".
- Port: Set to "389".
- Failover server used when primary is not available: This field is empty.
- Security:**
 - Bind distinguished name: Set to "cn=root".
 - Bind password: Set to "*****".
 - Federated repository properties for login: Set to "uid;cn".

© Copyright IBM Corporation 2014

Adding LDAP to the Virtual Member Manager realm: Step 2

The next step is to identify the new repository as an LDAP server. You enter the access information for the LDAP server, including:

- The type of LDAP server
- The server that is hosting the LDAP server
- The TCP/IP port number
- The bind distinguished name and password



Important: For Dashboard Application Services Hub to add new entries to the LDAP server, the bind user ID must have the appropriate LDAP authorities.

Adding LDAP to the Virtual Member Manager realm: Step 3

Configure the LDAP search criteria

- Limits the LDAP data to a specific subtree within the directory
- Defined for Group, OrgContainer, and PersonAccount

The screenshot shows a configuration interface for mapping LDAP object classes. On the left, under 'Additional Properties', the 'Federated repositories entity types to LDAP object classes mapping' section is selected. It lists resources: Group (selected), OrgContainer, and PersonAccount. On the right, a detailed configuration dialog is open for the selected 'Group'. It shows the 'Entity type' set to 'Group' and 'Object classes' set to 'groupOfNames'. Under 'Search bases', the value 'ou=tipgroups,cn=tipRealm,DC=IBM,DC=COM' is entered. The 'OK' button at the bottom of the dialog is highlighted with a red arrow.

© Copyright IBM Corporation 2014

Adding LDAP to the Virtual Member Manager realm: Step 3

In the next step, you define how Dashboard Application Services Hub searches the LDAP server for users and groups. In a production environment, the LDAP server might contain thousands of users, but only a single group that requires access to Web GUI. You can use these screens to limit the search context that Dashboard Application Services Hub uses.

Adding LDAP to the Virtual Member Manager realm: Step 4

Configure the Virtual Member Manager to write new users to LDAP

Defined for Group, OrgContainer, and PersonAccount

The screenshot shows two overlapping windows. The left window is titled 'Repositories in the realm' and lists resources: 'd0-lbm_dc-com' (selected) and 'o-defaultWIMFileBasedRealm'. It has sections for 'Additional Properties' with options like 'Property extension repository', 'Entry mapping repository', 'Supported entity types' (which is highlighted with a red box), and 'User repository attribute mapping'. The right window is titled 'Preferences' and shows 'Entity Type' set to 'Group', 'Base Entry for the Default Parent' set to 'ou=tipgroups,cn=tipRealm,DC=IBM,DC=COM', and 'Relative Distinguished Name Properties' set to 'cn'. A red arrow points from the 'Supported entity types' box in the left window to the 'Group' entry in the right window. Another red arrow points from the 'OK' button in the right window back to the 'Supported entity types' box in the left window.

© Copyright IBM Corporation 2014

Adding LDAP to the Virtual Member Manager realm: Step 4

The last step defines how Dashboard Application Services Hub writes new entries into the LDAP server. The configuration is similar to the search criteria from the previous slide. Again, in a production environment, the customer LDAP server might contain only a single group that requires access to Web GUI. You can use these screens to configure Dashboard Application Services Hub to write new entries within only that context.

Adding LDAP to the Virtual Member Manager realm: step 5

LDAP users known to Dashboard Application Services Hub

- All users within the defined LDAP search subtree appear automatically
- You must add roles to configure access to Dashboard Application Services Hub features

Select	User ID	First name	Last name	E-mail	Unique Name
<input type="checkbox"/>	abraman	Ariana	Braman	abraman@ibm.com	cn=Ariana Braman,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	adurling	Adeline	Durling	adurling@ibm.com	cn=Adeline Durling,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	bwinebarger	Bart	Winebarger	bwinebarger@ibm.com	cn=Bart Winebarger,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	dselan	Dick	Selan	dselan@ibm.com	cn=Dick Selan,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	eange	Earline	Ange	eange@ibm.com	cn=Earline Ange,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	elotempio	Emelda	Lotempio	elotempio@ibm.com	cn=Emelda Lotempio,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	ezegarelli	Else	Zegarelli	ezegarelli@ibm.com	cn=Else Zegarelli,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	gbailio	Gerald	Bailio	gbailio@ibm.com	cn=Gerald Bailio,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	hdold	Houston	Dold	hdold@ibm.com	cn=Houston Dold,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	jguglielmo	Jasper	Guglielmo	jguglielmo@ibm.com	cn=Jasper Guglielmo,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM
<input type="checkbox"/>	jmulberry	Jorge	Mulberry	jmulberry@ibm.com	cn=Jorge Mulberry,ou=tipusers,cn=tipRealm,DC=IBM,DC=COM

Adding LDAP to the Virtual Member Manager realm: Step 5

After LDAP is defined as a user repository within the VMM, all LDAP users and groups are available within Dashboard Application Services Hub. However, no roles are associated with any of the groups or users yet. To complete the configuration, you must add the necessary roles to users and groups that require access to Web GUI.

There are a few points worth mentioning. First, all users and groups are available to Dashboard Application Services that are contained within the LDAP search context that was configured previously. Well-defined search criteria is important because the LDAP server might contain thousands of users and only a small subset of those users require access to Dashboard Application Services Hub. Without a well-defined search context, you wind up with many more users in Dashboard Application Services Hub than need that access.

Second, the user and group information is not saved within Dashboard Application Services Hub. That information appears within the LDAP server. Dashboard Application Services Hub reads the data from the LDAP server and provides access to features to any user based on the associated roles. The student exercises contain a step that demonstrates what happens when the LDAP server is not available.

Synchronizing LDAP users with the ObjectServer

LDAP users known to Dashboard Application Services Hub

Enable the user synchronization function:

- This function creates the LDAP users in the ObjectServer
- Users can access all functions that write to the ObjectServer
- These functions include the Active Event List (AEL) and the Web GUI tools

Two steps required to enable synchronization:

```
/opt/IBM/netcool/omnibus_webgui/etc/server.init  
users.credentials.sync:true
```

```
/opt/IBM/netcool/omnibus_webgui/etc/datasources/ncwDataSourceDefinitions.xml  
<config maxAge="3600"/>
```

The default is once every hour (3600 seconds)

Change as appropriate

© Copyright IBM Corporation 2014

Synchronizing LDAP users with the ObjectServer

When the ObjectServer is configured as the default user repository, new Dashboard Application Services Hub users and groups are written to the ObjectServer. When LDAP is configured as the default user repository, you must manually enable a synchronization feature to provide a similar capability.

When synchronization is configured, new Dashboard Application Services Hub users and groups are added to the ObjectServer based on a user-defined frequency. The default frequency is 3600 seconds, or every hour.

This synchronization is required because all Dashboard Application Services Hub users that require access to Web GUI features must have their user IDs defined in the ObjectServer. If synchronization is not configured, the Netcool administrator must manually add users to the ObjectServer.

Student exercises



© Copyright IBM Corporation 2014

Student exercises

Complete the exercises for this unit.

Summary

You now should be able to perform the following tasks:

- Describe the functional architecture of the Web GUI component
- Describe the features and functions
- Describe the requirements for installation
- Perform a complete default installation
- Perform the postinstallation steps
- Validate basic functions
- Configure Dashboard Application Services Hub to use an LDAP repository

© Copyright IBM Corporation 2014

Summary



5 Basic ObjectServer administration



5 Basic ObjectServer administration



© Copyright IBM Corporation 2014

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

In this unit, you learn about the structure and functions of the ObjectServer. You learn about important tables and important columns. You also learn how to perform basic administration of the ObjectServer.

References: SC27-6265-00 *Administration Guide*

Objectives

In this unit, you learn to perform the following tasks:

- Navigate the Tivoli Netcool/OMNibus directory structure
- Describe the purpose of important directories
- Locate important files and binaries
- Describe a restriction filter
- Describe a group
- Describe a role
- Describe the requirements for an ObjectServer user
- Use the administration GUI to create an ObjectServer user
- Use the administration GUI to modify an ObjectServer
- Use the WebSphere Administrative console to define a group
- Use the WebSphere Administrative console to define a user
- Use Dashboard Application Services Hub to assign roles

© Copyright IBM Corporation 2014

Objectives



Lesson 1 Netcool/OMNIbus directory structure



© Copyright IBM Corporation 2014

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

In this lesson, you learn about the file structure of the Netcool®/OMNIbus installation. After completing this lesson, you should be able to describe the important files.

\$NCHOME directory

Eight directories are created by default in \$NCHOME

```
[netcool@host1 netcool]$ ls -l
total 32
drwxr-xr-x  2 netcool ncoadmin 4096 Sep  4 18:58 bin
drwxr-xr-x  4 netcool ncoadmin 4096 Sep  4 19:33 etc
drwxr-xr-x  3 netcool ncoadmin 4096 Sep  4 18:58 license
drwxr-xr-x  2 netcool ncoadmin 4096 Sep  4 19:47 log
drwxr-xr-x 17 netcool ncoadmin 4096 Sep  4 18:58 omnibus
drwxr-xr-x  3 netcool ncoadmin 4096 Sep  4 18:57 platform
drwxr-xr-x  4 netcool ncoadmin 4096 Sep  4 18:58 properties
drwxr-xr-x  2 netcool ncoadmin 4096 Sep  4 18:57 var
```

© Copyright IBM Corporation 2014

\$NCHOME directory

The default value for \$NCHOME on a UNIX system is **/opt/IBM/tivoli/netcool**. On a Windows system, it is **C:\IBM\Tivoli\netcool**.

\$NCHOME/bin directory

The \$NCHOME/bin directory contains these utilities:

- The command-line interface file generation utility, nco_igen
- The version utility, nco_id

This directory can contain non-Tivoli Netcool/OMNibus executable files in environments where multiple Tivoli Netcool products are installed

Tip: Put \$NCHOME/bin in your \$PATH

© Copyright IBM Corporation 2014

\$NCHOME/bin directory

This directory contains several utilities that are commonly used, including nco_igen and nco_id. The nco_igen utility generates the interfaces file from the contents of **omni.dat**.

The nco_id utility shows the version information that is related to the Netcool/OMNibus software, for example:

```
nco_id
```

```
Netcool/OMNibus 8.1.0 - May 2014
NCHOME: /opt/IBM/tivoli/netcool
IMHOME: /home/netcool/IBM/InstallationManager/eclipse
IMDATA: /home/netcool/var/ibm/InstallationManager
```

Products:

```
IBM Tivoli Netcool/OMNibus - 8.1.0
```

\$NCHOME/etc and \$NCHOME/platform directories

\$NCHOME/etc contains information relating to the Sybase OpenServer middleware

In particular, it contains the file **omni.dat**, which you can edit manually or with the nco_xigen GUI

\$NCHOME/platform contains binary files that are specific to your machine, relating to the Sybase OpenServer middleware

© Copyright IBM Corporation 2014

\$NCHOME/etc and \$NCHOME/platform directories

The most important files in **\$NCHOME/etc** are **omni.dat** and **interfaces.<arch>**. The **omni.dat** file is the text file that generates the interfaces file. This file is automatically modified when you use the nco_xigen utility. Otherwise, you can manually edit it with any text editor and then use it to generate the interfaces file by running nco_igen.

The **omni.dat** file is also created if you use the Initial Configuration Wizard. When you use the wizard, you are prompted to enter information that is related to the deployment. The wizard creates the **omni.dat** file and runs the nco_igen utility to create the interfaces file.

\$NCHOME/license and \$NCHOME/properties

\$NCHOME/license contains textual descriptions of the Netcool/OMNibus license agreement

The same text that is shown when installing Netcool/OMNibus

\$NCHOME/properties contains textual descriptions of installed components

Referenced by **\$NCHOME/bin/nco_id**

© Copyright IBM Corporation 2014

\$NCHOME/license and \$NCHOME/properties

\$OMNIHOME directory

Sixteen directories are created by default in \$OMNIHOME

```
[netcool@host1 netcool]$ ls -l omnibus/
total 64
drwxr-xr-x  2 netcool ncoadmin 4096 Sep  4 18:58 bin
drwxr-xr-x  3 netcool ncoadmin 4096 Sep  4 19:33 db
drwxr-xr-x  7 netcool ncoadmin 4096 Sep  4 18:58 desktop
drwxr-xr-x  6 netcool ncoadmin 4096 Sep  4 19:46 etc
drwxr-xr-x 12 netcool ncoadmin 4096 Sep  4 18:58 extensions
drwxr-xr-x  4 netcool ncoadmin 4096 Sep  4 18:58 gates
drwxr-xr-x  3 netcool ncoadmin 4096 Sep  4 18:58 install
drwxr-xr-x  3 netcool ncoadmin 4096 Sep  4 18:57 java
drwxr-xr-x  2 netcool ncoadmin 4096 Sep 23 16:52 log
drwxr-xr-x  3 netcool ncoadmin 4096 Sep  4 18:57 platform
drwxr-xr-x  3 netcool ncoadmin 4096 Sep  4 18:58 probes
-r--r--r--  1 netcool ncoadmin  586 May 27 12:39 RELEASE_ID
drwxr-xr-x  2 netcool ncoadmin 4096 Sep  4 18:58 tsm
drwxr-xr-x  2 netcool ncoadmin 4096 Sep  4 18:58 upgrade
drwxr-xr-x  2 netcool ncoadmin 4096 Sep  4 18:58 utils
drwxr-xr-x  3 netcool ncoadmin 4096 Sep 23 15:48 var
```

© Copyright IBM Corporation 2014

\$OMNIHOME directory

\$OMNIHOME is defined as **\$NCHOME/omnibus**. \$OMNIHOME is the parent directory for all Netcool/OMNIbus core components.

\$OMNIHOME/bin directory

The \$OMNIHOME/bin directory is where executable files are stored

These files might or might not be binary

Often, these files are scripts that abstract calls to the underlying binary, also known as wrappers

The actual executable files are stored in an architecture-dependent subdirectory, for example:

- \$OMNIHOME/platform/aix5 or
- \$OMNIHOME/platform/linux2x86

Tip: Put \$OMNIHOME/bin in your \$PATH

© Copyright IBM Corporation 2014

\$OMNIHOME/bin directory

Various operating systems support Netcool/OMNibus. To facilitate the packaging of the software to support multiple operating systems, the software incorporates a feature referred to as *wrapper scripts*. Much of the content within the \$OMNIHOME directory structure is text. Various property files, configuration files, license files, and others, are all text. This text does not vary based on operating system. However, one of the things that does vary across operating systems are executable programs. An executable program for an AIX system does not work on a Linux system.

The **bin** directory contains a collection of files that seem to be executable programs. For example, **nco_event**, which is the command that is run to start the Native Desktop Event List. In actuality, the file \$OMNIHOME/bin/nco_event is a symbolic link to a *wrapper script*. The wrapper script runs **\$OMNIHOME/platform/<arch>/nco_event**. The reference to <arch> means an architecture-specific subdirectory. An environment variable defines the text for the <arch> reference in the wrapper script. So, on a Linux system, the environment variable is defined as linux2x86. The wrapper script translates the command to **\$OMNIHOME/bin/platform/linux2x86/nco_event**.

\$OMNIHOME/db and \$OMNIHOME/desktop directories

\$OMNIHOME/db is used for database persistence files

- Each ObjectServer has its own subdirectory

\$OMNIHOME/desktop is used for Tivoli Netcool/OMNibus native desktop configuration files

- Subdirectory **app-defaults/<arch>** contains files specifying the desktop font and style
- The default desktop configuration files:
 - default.elc
 - default.elv

© Copyright IBM Corporation 2014

\$OMNIHOME/db and \$OMNIHOME/desktop directories

The ObjectServer is a memory-resident database. However, the structure is stored on disk. A database checkpoint is created periodically for recovery purposes. **\$OMNIHOME/db** is where the structure and checkpoint files are stored. There is a subdirectory for each ObjectServer, whose name is the same as the ObjectServer name. The Native Desktop uses files that the **\$OMNIHOME/desktop** directory contains.

\$OMNIHOME/etc and \$OMNIHOME/gates directories

Tivoli Netcool/OMNIbus components use the following configuration files and the files are placed in the \$OMNIHOME/etc directory

- Process control configuration files
- ObjectServer SQL initialization files
- ObjectServer properties files
- Generated interface files for communications, symbolic link back to \$NCHOME/etc
- Gateway properties files

Base configuration and supplementary files for gateways are in the \$OMNIHOME/gates directory

© Copyright IBM Corporation 2014

\$OMNIHOME/etc and \$OMNIHOME/gates directories

The **\$OMNIHOME/etc** directory contains various configuration files. One of these files is the ObjectServer property file. The property file is created when the ObjectServer is created.

The **\$OMNIHOME/gates** directory holds configuration files for all installed gateways. These files serve as templates. The best practice for configuring a gateway is to copy the template files from the respective **\$OMNIHOME/gates** directory to **\$OMNIHOME/etc**. You can modify the copied files, not the originals.

\$OMNIHOME/extensions directory

Contains prebuilt optional configurations, such as these examples:

- Additional ObjectServer functions
 - Probe event flood detection
 - Controlled ObjectServer shutdown
 - Multitier generation scripts
 - Return on investment (ROI)
- Integrations with other Tivoli products
 - IBM Tivoli Monitoring
 - Tivoli Application Dependency Discovery Manager (TADDM)
 - Tivoli Common Reporting

Complete details can be found in

Unit19: Extending the functionality of Tivoli Netcool/OMNibus

IBM Tivoli Netcool/OMNibus: Installation and Deployment Guide

© Copyright IBM Corporation 2014

\$OMNIHOME/extensions directory

The **\$OMNIHOME/extensions** directory is home to the collection of prebuilt configurations that are included with Netcool/®OMNibus. You can use some of the configurations to implement optional behavior, for example, probe event flood detection. Use other configurations to provide integration between Netcool/OMNibus and other Tivoli® products. Complete details on all the configurations are in the *IBM® Tivoli Netcool/OMNibus: Installation and Deployment Guide*.

\$OMNIHOME/install and \$OMNIHOME/java directories

\$OMNIHOME/install contains the startup scripts for a UNIX system

For example, you can have Tivoli Netcool/OMNibus start when the UNIX server transitions to a certain run level

\$OMNIHOME/java contains jar files for various applications written in Java

© Copyright IBM Corporation 2014

\$OMNIHOME/install and \$OMNIHOME/java directories

The **\$OMNIHOME/install** directory contains the utility that configures Netcool/OMNibus to start when the server is restarted. The **java** subdirectory contains a collection of JAR files that are used by various utilities that are based on Java.

\$OMNIHOME/log and \$OMNIHOME/platform directories

For all component debug and error log files, look inside the \$OMNIHOME/log directory

- Log files are maintained for ObjectServers, probes, gateways, and process control
- All Tivoli Netcool/OMNibus components produce log messages and can be configured for different levels of debugging
- First place to look when investigating problems

\$OMNIHOME/platform contains binary files that are specific to the supported platform, and operating system

- \$OMNIHOME/platform/linux2x86/bin (32-bit)
 - \$OMNIHOME/platform/linux2x86/bin64 (64-bit)
- The wrapper scripts in \$OMNIHOME/bin point here to the actual executable

© Copyright IBM Corporation 2014

\$OMNIHOME/log and \$OMNIHOME/platform directories

The most important directory on this slide is **\$OMNIHOME/log**. You can configure every Netcool/OMNibus component to generate a log file. The log subdirectory is the default location for all log files. This directory is the first place to look when there is an issue with any component.

\$OMNIHOME/probes, \$OMNIHOME/tsm, \$OMNIHOME/utils, and \$OMNIHOME/var directories

\$OMNIHOME/probes contains probe wrapper scripts

- Binaries are held in a platform-dependent subdirectory, \$OMNIHOME/probes/<arch>

\$OMNIHOME/tsm contains Telco Service Monitor wrapper scripts and binaries (if installed)

\$OMNIHOME/utils directory

- Precreated scripts for use with tools and automations
- Best practice location for user-created scripts
- Useful library of Bourne shell compatible functions that you can source from within your own scripts: nco_function

\$OMNIHOME/var directory

- Store and forward files
- PID files for process control

© Copyright IBM Corporation 2014

\$OMNIHOME/probes, \$OMNIHOME/tsm, \$OMNIHOME/utils, and \$OMNIHOME/var directories

The \$OMNIHOME/probes directory is where all installed probes are stored. Probe executable files are packaged with a similar wrapper script technique as the executable files in \$OMNIHOME/bin. Probes use an architecture-specific subdirectory that is in \$OMNIHOME/probes, for example, \$OMNIHOME/probes/linux2x86.

The \$OMNIHOME/var directory plays two important roles. First, whenever a component starts, an ObjectServer for example, a file is created in \$OMNIHOME/var. The file is typically named something like **AGG_P.pid**. This file is a text file, and it contains the process ID number. This file prevents the same component from being started more than one time. When a component starts, the directory is checked for the respective process ID file. If one exists, the component does not start.



Hint: In a situation where a server crashes, it is likely that the process id files are not removed correctly. When the server restarts, a component might fail with a message that indicates it is already running. In that case, remove the appropriate process id file from \$OMNIHOME/var.

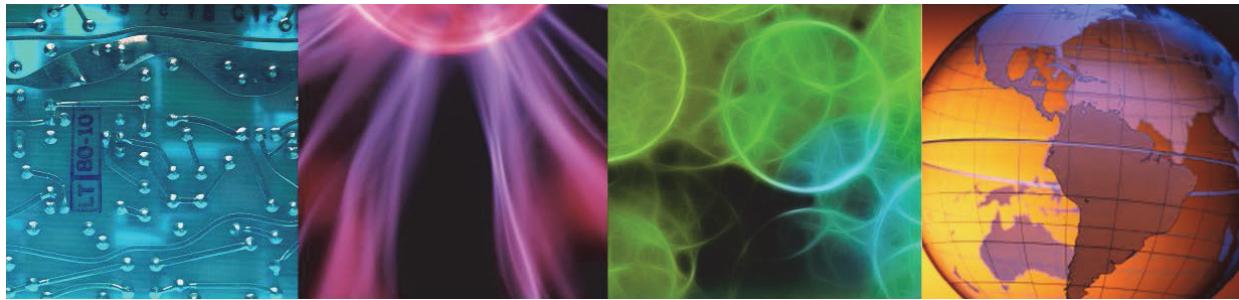
The second role played by \$OMNIHOME/var is as the directory where component store-and-forward (SAF) files are saved.



Lesson 2 ObjectServer structure



Lesson 2 ObjectServer structure



© Copyright IBM Corporation 2014

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

In this lesson, you learn about the structure of the ObjectServer. You learn about important tables, columns, and properties. After completing this lesson, you should be able to perform the following tasks:

- Describe the important tables within the ObjectServer.
- Describe the important columns in the event record.
- Describe the important ObjectServer properties.

Object Server databases

Initially, the Object Server has the following databases:

- alerts: Alert data, and event list configuration
- catalog: System catalog containing Object Server metadata, can be viewed but not modified
- custom: Database for tables users add
- iduc_system: Channel setup for accelerated event notification (AEN)
- master: Compatibility with previous releases, desktop ObjectServer tables
- persist: Triggers, procedures, and signals
- precision: Tables for integration with IBM Tivoli Network Manager
- registry: Contains information about distributed Tivoli Netcool/OMNIbus configurations
- security: Authentication information for users, roles, groups, permissions
- service: Use to support IBM Tivoli Composite Application Manager for Internet Service Monitoring

© Copyright IBM Corporation 2014

Object Server databases

The ObjectServer is typically referred to as a memory resident database. In reality, it is a collection of multiple databases. This slide contains a list of the databases that are defined in any ObjectServer. These databases are the ones that the ObjectServer has initially. A user with the appropriate authority can add a database to the ObjectServer. Users can also add their own databases, database tables, and columns in a database table.

A user typically never directly accesses most of the databases, and tables that are defined in the ObjectServer. For example, several tables are for storing user ID information. The user does not need to know the structure of these tables. The user can add, change, or delete user IDs with the administrator utility without any knowledge of the structure of these tables.



Important: The ObjectServer is case-sensitive. All of the default databases are defined with names that contain all lowercase letters. Any use of the actual database name, in an SQL statement, for example, must reference the name in lowercase.

Database and table restrictions

Some restrictions apply when creating new tables in Tivoli Netcool/OMNibus ObjectServer:

- Maximum field size = 8192 bytes (8 KB)
- Maximum row size = 64 KB
- Maximum number of columns = 512
Excluding system-maintained columns

© Copyright IBM Corporation 2014

Database and table restrictions

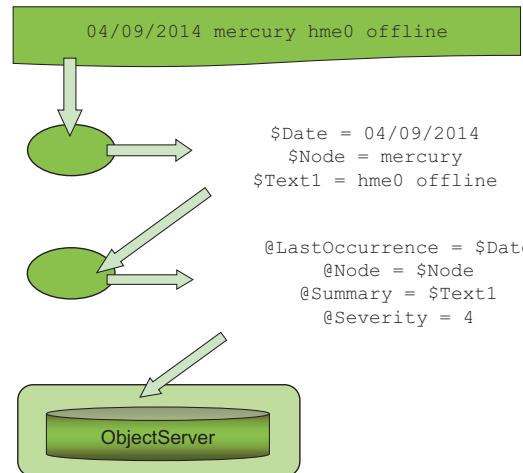
This slide lists some of the architectural limitations of the ObjectServer.

ObjectServer structure: alerts.status

A probe receives data from a device in device-specific format

A probe parses the raw data through the rules file and converts it into the common event format

The probe passes the newly formatted data to Tivoli Netcool/OMNibus



© Copyright IBM Corporation 2014

ObjectServer structure: alerts.status

The ObjectServer table alerts.status defines the structure of the Netcool/OMNibus event record. Netcool/OMNibus probes populate the alerts.status table. The probe collects information from some source, and breaks the information into pieces referred to as tokens. The probe assigns a token to a column in the alerts.status table. The probe populates the table by creating an SQL INSERT statement. That statement is forwarded to the ObjectServer, and causes a record to be created in the alerts.status table.

ObjectServer structure: alerts.details

Under certain circumstances you might be interested in the raw data from the probe:

- When details tracking is enabled, data is stored in the alerts.details table as token-value pairs
- You can view details from the **Details** tab in the Information window, which you access through the **Alerts** menu for selected events
- Details are linked to their respective events using the **Identifier** field, which is a primary key in both the alerts.status and alerts.details tables
- There is a 1-to-1 correspondence between events and details

One detail record (alerts.details) possible for each event record (alerts.status)

© Copyright IBM Corporation 2014

ObjectServer structure: alerts.details

One of the primary roles of a probe is to convert cryptic information from the event source into human-readable text in the ObjectServer event record. There are times when it might be important to see the original cryptic data, typically when debugging some problem. You can configure the probe to send extra data to the ObjectServer whenever an event is created. This additional data is saved in the alerts.detail table. There is a database link between alerts.status and alerts.details, which enables a user to view the details that are related to a specific event.

ObjectServer structure: alerts.journal

When working with events, you might want to track the history of a particular event:

- Who has owned it
- What severity levels it has passed through
- What automations have acted on it

Journals provide these functions:

- Journal information is held in the alerts.journal table
- Journals are linked to their respective events through the Serial field, which is a primary key in both alerts.status and alerts.journal tables

There is a 1-to- n correspondence between events and journals

- There can be n journal records (alerts.journal) for each event record (alerts.status)

© Copyright IBM Corporation 2014

ObjectServer structure: alerts.journal

When working with events in Tivoli Netcool/OMNibus, you often want to track the history of an event. You want to know who owns the event, what severity levels it passes through, what automations act upon it, and more. You can use the *journal* to track the history. When a new journal entry is added, the data is stored in the alerts.journal table.

The journal entry contains the name of the user, the date, time, and text that describes the operation. This information provides an important chronological history of actions that are taken against the event.

You can add entries to the journal manually. You can also set up and run a desktop tool to add entries automatically.

Default ObjectServer fields

Some fields are self-explanatory. However, it is important for system administrators to know the functions of the default fields like the following examples:

- Identifier
- Serial
- StateChange
- FirstOccurrence
- LastOccurrence
- Type
- Acknowledged
- Tally

© Copyright IBM Corporation 2014

Default ObjectServer fields

A default ObjectServer event record consists of approximately 100 columns. Users can add more columns. Not all columns are populated for every event. The following list of columns are always populated.

- **Identifier:** This column is the unique identifier for the alerts.status database and is key to making deduplication work.
- **Serial:** This column is an automatically populated field and is a unique reference for an event within a particular ObjectServer. Netcool/OMNibus automatically assigns a number to this field when a new event is added to the ObjectServer.
- **StateChange:** Netcool/OMNibus updates this field with the current time each time the state of an event changes, either from the probe or the ObjectServer.
- **FirstOccurrence:** This time field column contains a time stamp of when the event is first created in the ObjectServer, and should not change from the initial value.
- **LastOccurrence:** This time field column contains a time stamp of the last occurrence of the event. Unlike FirstOccurrence, this time stamp changes based on deduplication of an event.



Note: ObjectServer time stamps are in Coordinated Universal Time (UTC) format. Time values are saved in whole seconds.

- **Type:** This column is an integer field and can generally take three values: 0, 1, and 2. A type of 0 means that the type is not set. A type of 1 means that the event is a problem event, link down,

for example. A type of 2 means that the event is a resolution event, link up. The generic_clear trigger uses this column.

- **Acknowledged:** A user can acknowledge events in Netcool/OMNibus. It is this field that represents the acknowledgment of an event. It is an integer field but behaves as a Boolean: 0=unacknowledged and 1=acknowledged.
- **Tally:** An integer that indicates the number of times an event deduplicates. The value is set to 1 when the event is initially created. The column increments by 1 each time the event recurs. The deduplication trigger increments the count.

Default ObjectServer fields, continued

- Severity
- ServerSerial
- ServerName
- OwnerUID/GID
- AlertGroup
- AlertKey
- Manager
- Summary
- Class

© Copyright IBM Corporation 2014

Default ObjectServer fields, continued

The list of fields that are always populated continues:

- **Severity:** This field denotes the severity or priority of the event within the ObjectServer. It is an Integer field and has these values by default:
 - 0: Clear
 - 1: Intermediate
 - 2: Warning
 - 3: Minor
 - 4: Major
 - 5: Critical
- **ServerSerial** and **ServerName**: These columns identify the ObjectServer that receives the event first, which is important for architectures with multiple ObjectServers, and gateways.
- **OwnerUID**: The ownership owner ID of the event in alerts.status table. The nobody user owns all events by default. This column changes when a user takes ownership of the event.
- **OwnerGID**: The ownership group ID of the event in alerts.status table.
- **AlertGroup**: Assigned in the rules file to identify an event for correlation in the generic_clear automation.
- **AlertKey**: Assigned in the rules file to identify an event for correlation in the generic_clear automation.

- **Manager:** Assigned in the rules file and normally denotes the probe that generated the event.
- **Summary:** A textual description of the problem that is associated with the event. Set in the rules file.
- **Class:** A way of classifying equipment types to events. Enables assigning tools to be used against events of specific equipment types.

Data storage

The ObjectServer is memory-resident but does make regular database backups to a disk

- Uses checkpoint files
- The default interval between checkpoints is every 60 seconds
- Between checkpoints, replay logs record changes
- Upon a planned shutdown, all permanent database files are written to disk and the replay logs are deleted
- A checkpoint is a memory image of the ObjectServer
 - Because it is a binary file, you cannot manually view it
 - Because it is machine dependent, you cannot port it to another system

© Copyright IBM Corporation 2014

Data storage

At 60-second intervals, the ObjectServer creates a physical checkpoint. The system alternatively backs up (checkpoint) to files in the appropriate **\$OMNIHOME/db/<ObjectServer name>** directory.

Between checkpoints, log files are updated with new inserts and updates. The checkpoint files are not in ASCII text and you cannot view them.



Note: The checkpoint file is essentially a memory snapshot of the ObjectServer. The file is machine-dependent and cannot be moved and used on another system.

Data storage, continued

When you restart the ObjectServer after a planned shutdown, the checkpoint files are read in immediately after the ObjectServer restarts

After an unplanned shutdown, the checkpoint files are read in after the ObjectServer restarts, followed immediately by the replay log files

© Copyright IBM Corporation 2014

Data storage, continued

If the ObjectServer fails, when it is restarted, the last checkpoint file is read and used to create the ObjectServer in memory. Then the corresponding log file is read, and the modifications that are contained in that file are applied.

With a planned shutdown, a checkpoint is created at the point just before the ObjectServer stops. In this instance, there is no need for a log file. When the ObjectServer restarts, it is only necessary to read the checkpoint file.

Region storage and physical checkpoints

Files associated with region storage

- table_store_0.chk and master_store_0.chk
- table_store_1.chk and master_store_1.chk
- table_store_0.log and master_store_0.log
- table_store_1.log and master_store_1.log
- table_store.tab and master_store.tab

When creating a checkpoint, the system alternates between the 0.chk and the 1.chk files for both the table_store and master_store files

At each checkpoint, the logging process starts writing to the alternative log file

© Copyright IBM Corporation 2014

Region storage and physical checkpoints

There are two sets of checkpoint files, which are referred to as 0 and 1. The system alternates checkpoints between them. There are two corresponding log files, also referred to as 0 and 1.

Region storage, continued

Upon a planned shutdown, .tab files are created from the .chk and .log files

- master_store.tab
- table_store.tab

There are no .log files

When the ObjectServer restarts after a planned shutdown, the databases are created from the .tab files

After an unexpected shutdown, the latest .chk and .log files are used to rebuild the databases

If one of the .chk files is corrupted, the other .chk file is used in conjunction with the corresponding .log file

© Copyright IBM Corporation 2014

Region storage, continued

When the ObjectServer is brought down gracefully, a single checkpoint is created. The files that are associated with this checkpoint are labeled with tab. When the ObjectServer starts, the **\$OMNIHOME/db/<ObjectServer>** directory is examined for the tab checkpoint. It uses these files, if found.

Deduplication

Event reduction capability

Recognizes individual events and ensures they are only represented once in the ObjectServer

If an event comes in more than once, the following occurs:

- The **Tally** field is incremented
- The **LastOccurrence** field is updated

© Copyright IBM Corporation 2014

Deduplication

Deduplication is a noise-reducing facility. Deduplication recognizes unique events and ensures that individual events are only represented once in the Event List.

When Netcool/OMNIbus receives a new event, it checks all of the events currently in the ObjectServer. If the event exists in the ObjectServer, based on the value of the Identifier field, it increments the **Tally** field by 1.

Deduplication behavior

The deduplication behavior is controlled through an automation trigger or probe rules

- Automation

The default trigger, deduplication, can be modified to allow additional fields to be updated during deduplication

- Probes

In probe rules, you can use the update(<fieldname>) command to force an update of a specified field, unless the same field is specified in the deduplication trigger

© Copyright IBM Corporation 2014

Deduplication behavior

Netcool/OMNIbus uses an automation to implement deduplication. By modifying this automation, you can define which fields you want updated globally on deduplication. When set in the probe rules file, a specific field can be forced to update on an event-by-event basis.

Properties

The ObjectServer uses properties, which are stored in a properties file, to control its behavior

- The properties file is in the **\$OMNIHOME/etc** directory
- The properties file is named after the ObjectServer, for example, NCOMS **\$OMNIHOME/etc/NCOMS.props**
- You can set properties directly in the properties file
Changes take effect only when the ObjectServer restarts
- You can change properties dynamically using the Netcool/OMNibus Administrator utility, **nco_config**

© Copyright IBM Corporation 2014

Properties

When the **nco_dbinit** utility creates an ObjectServer, it also creates a property file in **\$OMNIHOME/etc**. This file controls the behavior of the ObjectServer after it starts.

Properties, continued

Netcool/OMNIbus Administrator is used to set the ObjectServer properties

Properties set in this way might have an immediate effect, depending on the property

Follow these steps to access ObjectServer properties:

1. Run `$OMNIHOME/bin/nco_config`
2. Log in to the required ObjectServer
3. Ensure that the user is a superuser

© Copyright IBM Corporation 2014

Properties, continued

The Netcool/OMNIbus Administrator utility modifies the structure of a running ObjectServer. This utility provides the option to modify some ObjectServer properties. Some properties cannot be changed. Of the properties that can be changed, some of the changes might take effect immediately. Other property changes require an ObjectServer restart.

Properties using nco_sql

From the command line, use the **nco_sql** command to view the properties that the user can change:

Log in to the appropriate ObjectServer and enter the following commands:

```
1> select PropName from catalog.properties where
   IsModifyable=TRUE;
2> go
```

You can also use procedural SQL to change properties

© Copyright IBM Corporation 2014

Properties using nco_sql

You can use the **nco_sql** command line utility to view ObjectServer properties. You can also use it to change any property that is modifiable. The slide illustrates the SQL statement that you can use to produce a list of those ObjectServer properties that can be modified.

Properties using nco_sql, continued

Some properties take immediate effect when changed

To find out which properties take immediate effect, use this command:

```
1> select PropName, Value from catalog.properties where
   IsImmediate=TRUE;
2> go
```

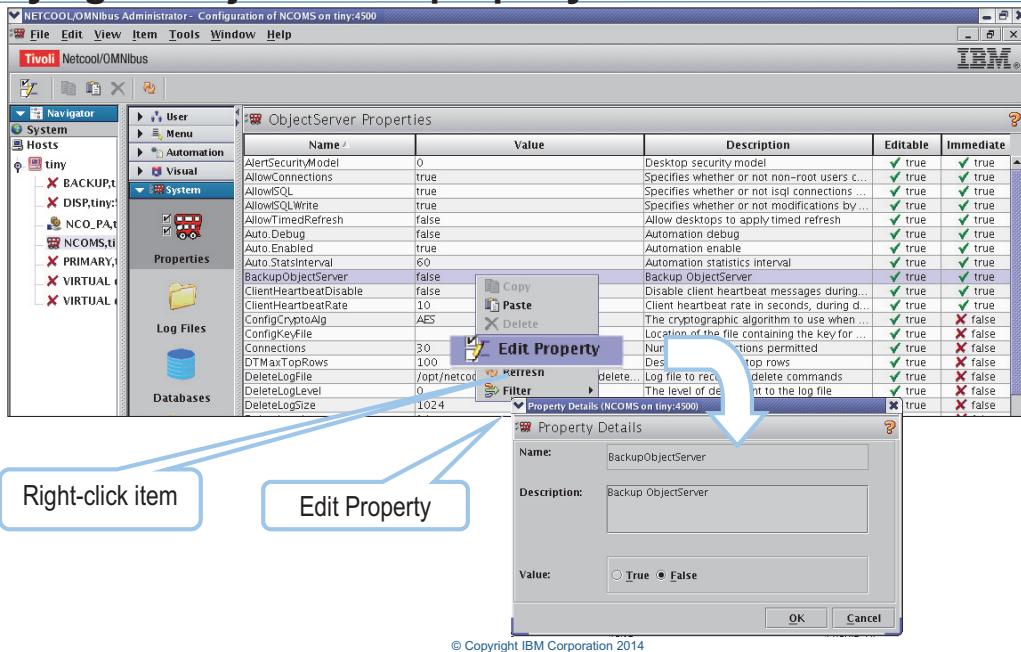
You can also find this information in the GUI using nco_config

© Copyright IBM Corporation 2014

Properties using nco_sql, continued

There are some ObjectServer properties where a change takes effect immediately. The slide illustrates the SQL statement that produces a list of those ObjectServer properties where changes take effect immediately.

Modifying an ObjectServer property



Modifying an ObjectServer property

The screen capture shown here highlights the use of the Administrator to modify an ObjectServer property.

Important properties

ActingPrimary	TRUE FALSE [TRUE]
AlertSecurityModel	integer [0]
AllowConnections	TRUE FALSE [TRUE]
AllowISQL	TRUE FALSE [TRUE]
BackupObjectServer	TRUE FALSE
Connections	integer [256]
DeleteLogFile	string

© Copyright IBM Corporation 2014

Important properties

The ObjectServer contains over 100 property settings. The following slides describe some of the more important properties:

- **ActingPrimary** TRUE | FALSE

The bidirectional gateway uses this property to determine whether the backup ObjectServer is acting in the role of primary ObjectServer.

- **AlertSecurityModel** integer

This property determines whether group row level security is enforced in the event list. By default, group row level security is disabled (0). In this case:

A member of the normal group can modify a row that is assigned to themselves or the nobody user.

A member of the administrator group can modify a row that is assigned to themselves, the nobody user, or a member of the Normal group. If the AlertSecurityModel property is enabled(1), only users in the group that owns the row can modify the row. In this case, a member of the Normal or Administrator group can modify a row that is assigned to a group of which they are a member. A member of the System group can always modify any row.

- **AllowConnections** TRUE | FALSE

Specifies whether non-root users can connect to the ObjectServer. If FALSE, no new connections to the ObjectServer are allowed. The default is TRUE.

- **AllowISQL** TRUE | FALSE

Specifies whether the ObjectServer allows connections by the SQL interactive interface. If FALSE, no user can connect with the nco_sql utility. The default is TRUE.

- **BackupObjectServer** TRUE | FALSE

Provides fail back capability with desktop clients, probes, the proxy server, and the ObjectServer gateway. The default is FALSE, and the desktop clients, probes, and gateways are assumed to be connected to a primary ObjectServer. When TRUE, the desktop clients, probes, and gateways are made aware that they are connected to the backup ObjectServer in a failover pair. If so, the desktop clients, probes, the proxy server, and the ObjectServer gateway automatically check for the recovery of the primary ObjectServer in the failover pair. These components fail back when the primary ObjectServer is available.

- **Connections** integer

Sets the maximum number of available connections for desktop clients, probes, and gateways. The default value is 256.

- **DeleteLogFile** string

The path and name of the delete log file, where all delete commands are recorded if delete logging is enabled. By default, deletes are logged to the file \$OMNIHOME/log/servername_deletes_file.logn.

Important properties, continued

DeleteLogging	TRUE FALSE [FALSE]
DeleteLogLevel	integer [0]
DeleteLogSize	integer [1048576]
DTMaxTopRows	integer [100]
Granularity	integer [60]
Profile	TRUE FALSE [TRUE]
MessageLevel	character

© Copyright IBM Corporation 2014

Important properties, continued

The list of important properties continues:

- **DeleteLogging** TRUE | FALSE

When TRUE, delete logging is enabled. The ObjectServer creates an entry in a log file for every event that is deleted. The default is FALSE.

- **DeleteLogLevel** integer

The log level determines how much information is sent to the delete log file. Possible settings include these examples:

< 0: No logging.

0: Client type, application ID, for example, ctisql for nco_sql and SQL run. This value is the default log level.

1: Time, user ID, client type, and SQL run.

- **DeleteLogSize** integer

The maximum size of the delete log file. When the log file **servername_deletes_file.log1** reaches the specified size, it is renamed **servername_deletes_file.log2**. A new log file, **servername_deletes_file.log1**, is created. When the new file reaches its maximum size, the older file is deleted and the process repeats. The output from a single delete command is never split between log files. Therefore, log files can be larger than the specified size. The default log file size is 1024 K Bytes.

- **DTMaxTopRows** integer

The maximum number of rows that an administrator can specify when using the view builder to restrict the number of rows an event list user can view. The default is 100.

- **Granularity** integer

The default refresh rate for IDUC (insert, delete, update, and control) clients. Native desktops and gateways are IDUC clients. At a high level, this property is the rate at which the ObjectServer informs IDUC clients that an updated batch of data is ready. The client then requests the updated data set from the ObjectServer.

- **Profile:** TRUE | FALSE

This property enables the collection and reporting of statistics that are related to the performance of the ObjectServer. Statistics are collected for each granularity period, 60 seconds. Automations collect the data, and write the results to <OS NAME>_profiler_report.log. The default is TRUE.

- **MessageLevel:** character

This property configures the level of verbosity in the ObjectServer message log. Default is warn.

ObjectServer OSLC interface overview

This evolving implementation facilitates integration adoption across IBM applications and provides the ability for third-party implementations

The Netcool/OMNIbus OSLC interface supplies an event service provider that presents resource linked data view of events and associated journal and detail resources

Messages are exchanged by RDF/XML

Certification is in progress for this interface to be officially recognized as an OSLC interface

The ObjectServer is an Event Service Provider

It is a linked-data interface that is RESTful

© Copyright IBM Corporation 2014

ObjectServer OSLC interface overview

As a language-independent means of accessing ObjectServer data, two HTTP-based application programming interfaces (APIs) are provided: A lightweight API, the HTTP interface, and a resource-based interface, the Open Services for Lifecycle Collaboration (OSLC) interface. Both APIs are hosted in the ObjectServer and use the same technology as the HTTP interface for probes. Access to the URI is authenticated with a known ObjectServer user through basic HTTP authentication. The HTTP interface provides access to table data in the ObjectServer through a structured URI format that uses HTTP. POST, PATCH, GET, and DELETE requests are supported against table URIs or row URIs. For content, application/json messages are supported. Table row data is returned and accepted through a single common JSON message format, the row set. A URI that represents an SQL command factory can run arbitrary SQL commands. The OSLC interface is an event server provider that presents a resource-linked data view of events and also the associated journal and detail resources. Messages are exchanged in RDF and XML syntax. The ObjectServer can be registered in the Jazz® for Service Management (JazzSM) service provider registry. Netcool/OMNIbus V8.1 does not include the Jazz for Service Management installation files. Both interfaces can be enabled and configured by setting properties in the ObjectServer.

Important OSLC properties

NHttpd.AccessLog	character
NRestOS.Enable	TRUE FALSE [FALSE]
NHttpd.EnableHTTP	TRUE FALSE [FALSE]
NHttpd.ListeningHostname	character
NHttpd.ListeningPort	integer [0]

© Copyright IBM Corporation 2014

Important OSLC properties

These properties are important for OSLC:

- **NHttpd.AccessLog:** character

Specifies the name and location of the log file where the server logs all requests that it processes.

- **NRestOS.Enable:** TRUE | FALSE

Enables the HTTP interface and the OSLC interface to the ObjectServer. \ Default is FALSE.

- **NHttpd.EnableHTTP:** TRUE | FALSE

Enables use of the HTTP port. Default is FALSE.

- **NHttpd.ListeningHostname:** character

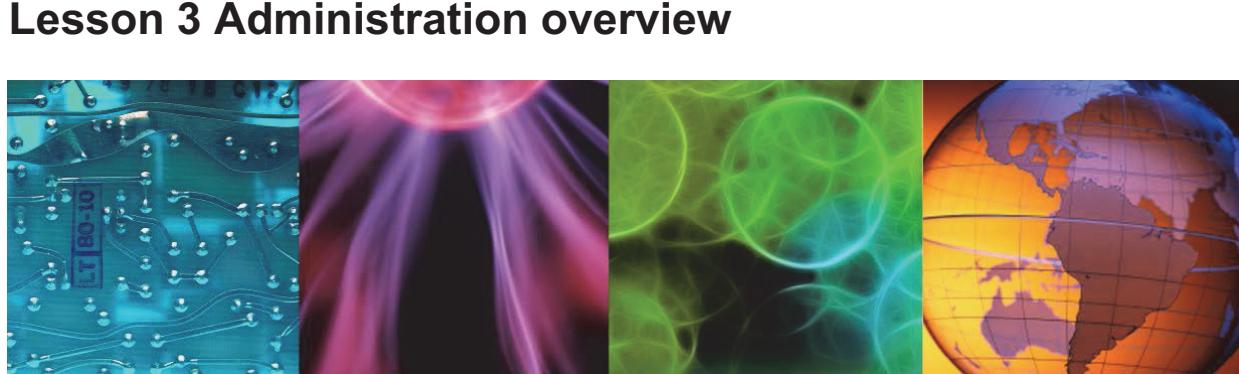
Specifies the listening host name or IP address that is used as the host name part of a URI to the ObjectServer HTTP or HTTPS interface. The default is localhost.

- **NHttpd.ListeningPort:** integer

Specifies the port on which the ObjectServer listens for HTTP requests. The default is 0.



Lesson 3 Administration overview



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to create ObjectServer users. After completing this lesson, you should be able to perform the following tasks:

- Create a restriction filter
- Create a group
- Create a user
- Understand how LDAP is used as a source of password authentication

Basic ObjectServer administration

Separate GUI for administration

- Installed when desktop feature selected
- Can administer multiple ObjectServers from the same GUI

ObjectServer changes require special user authority

- System: Can change anything
- Administrator: Can change menus, tools, colors, conversions, and classes

Typical administration tasks consist of these examples

- Adding or changing users (system)
- Adding or changing desktop menus and tools (system or administrator)
- Adding or changing automations or triggers (system)
- Adding or changing columns in database tables (system)

© Copyright IBM Corporation 2014

Basic ObjectServer administration

The Netcool/OMNIbus Administrator is the utility that administers the ObjectServer. This utility is a separate component that is installed when the Desktop feature is selected. One copy of this utility can administer all ObjectServers regardless of their location.

The authority to change the ObjectServer is granted to two specific groups: System and Administrator. A user in the System group can change any feature in the ObjectServer. A user in the Administrator group is limited to features related to desktop users, including menus and tools.

After an ObjectServer is built and activated, the administrator completes these tasks:

- Adding and changing ObjectServer users
- Creating new desktop menus and tools
- Creating or modifying automations
- Adding new database tables or columns to existing tables

Most changes that are made by the Netcool/OMNIbus Administrator take effect immediately. Stopping and starting the ObjectServer is not needed to activate these changes.

User administration

Default ObjectServers contain two users

- root: System level user, no password configured initially
- nobody

User definitions stored in ObjectServer tables

- Security database

Password authentication source

- ObjectServer: Passwords are stored internally
- External: One of the following methods, but *not* both
 - Operating System: UNIX PAM, for example
 - LDAP: Tivoli Directory Server, Active Directory or other

Note: User names are stored in the ObjectServer, but passwords are authenticated using the defined target (operating system or LDAP)

© Copyright IBM Corporation 2014

User administration

All ObjectServer user IDs are stored within the ObjectServer. The user that creates the ObjectServer user ID defines how password authentication is accomplished. There are two options available:

- **ObjectServer:** The password is stored directly in the ObjectServer.
- **External:** The host operating system maintains the password, or it is stored in an LDAP repository. The ObjectServer is configured to use one or the other of these options, but not both. If the ObjectServer is configured to use LDAP authentication, any ObjectServer user ID that is defined as external authentication uses the defined LDAP repository for their password authentication.

In summary, if a user ID is defined with ObjectServer authentication, the password is stored in the ObjectServer. If a user ID is defined as external authentication, the host operating system or LDAP authenticate the password. This configuration option is a system-wide option.

User configuration requirements

Restriction filters

- An option that can be used to restrict the events the user can access
- Applied at the user or group level

Groups

- Used to associate common attributes to one or more users
- Roles, restriction filters
- Can use to restrict desktop tool access

Roles

- Define database and table access permissions
- Alter, delete, drop, insert, select, and update

Users

- User name: Textual user ID
- Full Name: Long name for user
- Restriction filter: If used
- Assigned groups
- Password: Internal or external

© Copyright IBM Corporation 2014

User configuration requirements

Four specific features are associated with ObjectServer user IDs.

Restriction Filters

This feature is an optional one that can apply to a user or a group. The filter restricts the event records that are available to the user. This filter takes precedence over any additional filters that the user might configure on a personal desktop. One typical use of this feature is with *managed service providers* (MSP).

The MSP can use a single ObjectServer to manage multiple customers. The MSP can create user IDs for their customers and define restriction filters so that one customer's users cannot see another customer's events.

Groups

Groups associate common attributes to users. In the example of the MSP, they can create a separate group definition for each of their customers. This group definition would contain the customer-specific restriction filter. Then, when the MSP defines individual customer user IDs, they need only to assign the user to the respective group. The user ID inherits the restriction filter. Another key attribute of the group is the ability to restrict desktop tool access.

An organization can have users that require access to specific diagnostic tools. The ability to start those tools can be incorporated into the desktop to facilitate access to these tools. The organization

can decide to limit access to one or more of the tools to selected users. By placing users into a specific group, you can define the tool so that access is restricted to members of that group.

Roles

Roles provide permission to make specific database changes, such as insert and delete. Roles define which users possess read access to event records.

Users

The user definition contains a user name. It is the short name, or actual user ID field that allows the user to log in to the ObjectServer. It also provides the option to define a full name. The full name shows in audit tracking capabilities and is convenient for identifying the user.

Connection wizard

You launch Tivoli Netcool/OMNibus Administrator using a command line

- `$OMNIHOME/bin/nco_config &`
- Uses the interfaces file to identify components
- Wizard imports file and defines connections

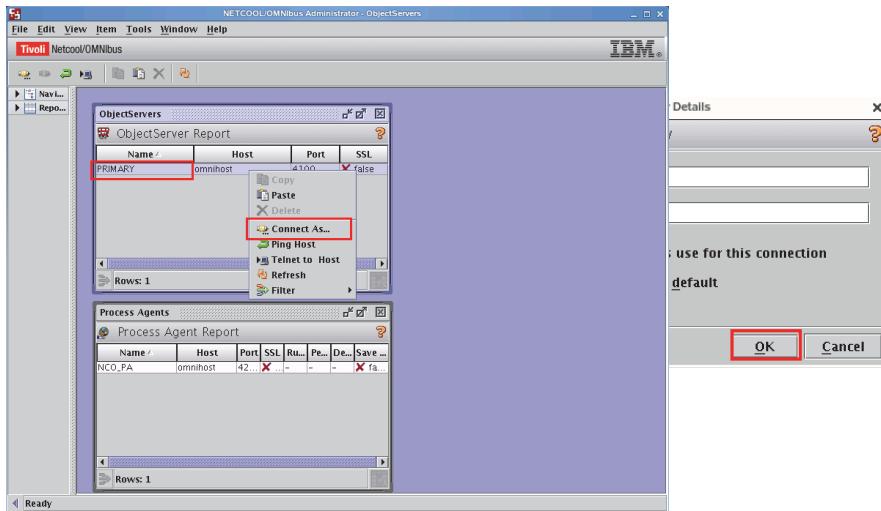


Connection wizard

The Netcool/OMNibus Administrator can manage multiple ObjectServers. It uses the interfaces file to determine what ObjectServers are available and how to communicate with them. The Administrator utility detects changes that are made to the interfaces file. The Administrator utility then triggers a wizard that reads the **omni.dat** file, and imports the definitions. This situation happens only the first time that a change is detected.

Tivoli Netcool/OMNibus Administrator

Select an ObjectServer



© Copyright IBM Corporation 2014

Tivoli Netcool/OMNibus Administrator

After the Administrator utility is activated, you see two boxes. The boxes can be overlaid in the window. You must move one box to see the other box. The ObjectServer Report box contains a list of all ObjectServer definitions that are in the **omni.dat** file. In the example, only one is listed.

To connect to an ObjectServer, you select the line, right-click it to show the menu, and then click **Connect As**. A pop-up window is shown, prompting for user credentials.

The default ObjectServer contains two user IDs: **root** and **nobody**. The **root** user ID is configured with system authority and can change any feature in the ObjectServer. Initially, the **root** user has no password.

Accessing the Users tab

- Users are managed from the **Users** tab
- Note the two users
 - root
 - nobody

Name	Full Name	Type	ID	Extern...	Enabled
nobody	Nobody	Unknown	65534	X false	X false
root	Root User	Super User	0	X false	✓ true

© Copyright IBM Corporation 2014

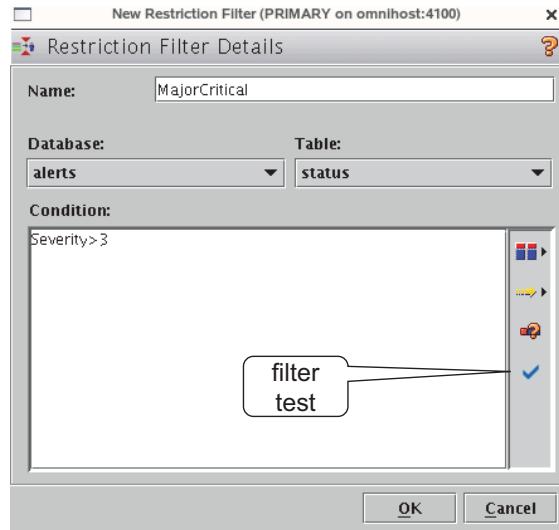
Accessing the Users tab

The tabs in the left pane provide access to ObjectServer features. Click the right arrow next to each category to expand the list of features within the category.

Restriction filters

Restriction filters create a filter (SQL where clause) that is applied to groups and users

- Filter on any database or table
- Filters are inherited by users in group



© Copyright IBM Corporation 2014

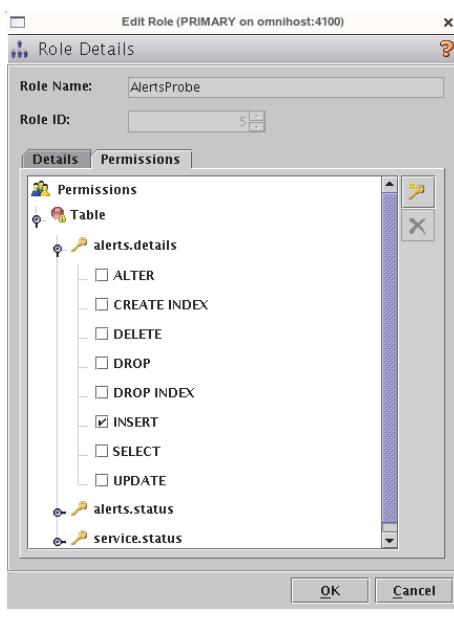
Restriction filters

This slide shows the screen that you use to define a restriction filter. If restrictions are required in your environment, they must be defined first. You reference them when the user or group you define requires the restriction filter.

Several icons are visible on the right side of this screen. You can use these icons to define a restriction filter. An important icon, filter test, is highlighted. When you select this icon, the system validates the syntax of the filter statement.

Roles

- Roles allow actions within groups
- Roles authorize viewing and modification of information



© Copyright IBM Corporation 2014

Roles

The Netcool/OMNibus security model uses roles and permissions to give system administrators control over access in Netcool/OMNibus.

When creating a new user, you must follow these steps in order.

1. Create restriction filters, if required.
2. Create a role or use an existing role.
3. Create a group by assigning a role and restriction filter or use those existing.
4. Create a user by assigning one to a group and creating further restriction filters as necessary.

Groups

The screenshot shows the 'Groups' management interface in Tivoli Netcool/OMNIbus. The left sidebar has links for 'Users', 'Groups', 'Roles', and 'Restriction Filters'. The main area displays a table of groups with columns: Name / Description, ID, and System. A message at the top says 'There are default Tivoli Netcool/OMNIbus groups'. The table data is as follows:

Name / Description	ID	System
Gateway	5	false
ISQL	7	false
Normal	6	false
Probe	3	true
Public Group	4	false
System	0	true
System Group	1	true

© Copyright IBM Corporation 2014

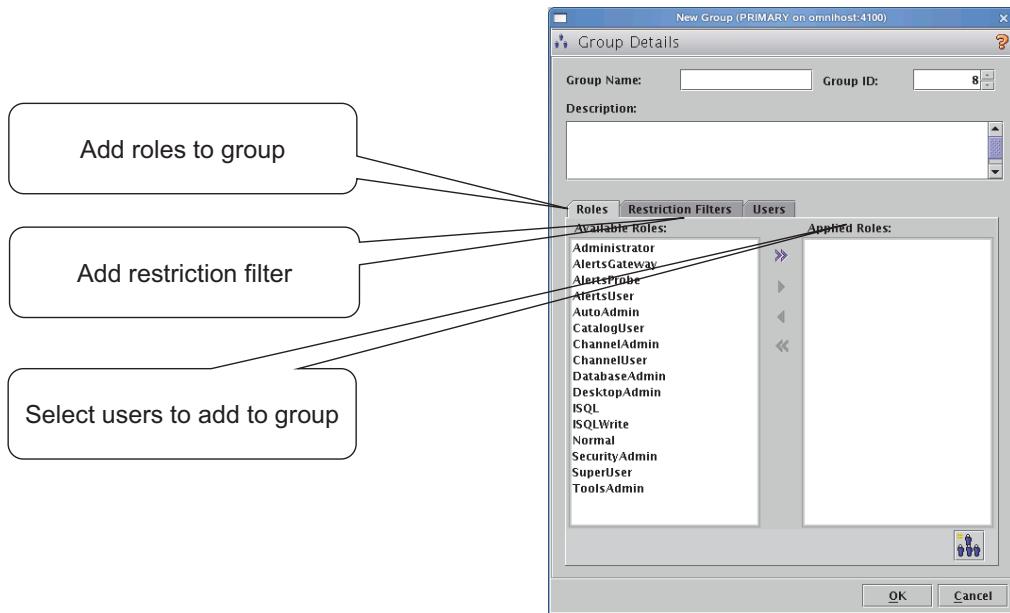
Groups

Group definitions facilitate the association of attributes to multiple users. The attributes include roles, permissions, and restriction filters.

Group definitions help facilitate work flow. Users can create group definitions for functional areas within their operation. When an event shows in the ObjectServer, an option is available for the user to take ownership of the event. When a user assumes ownership of the event, the record is updated with the user's user ID (UID) and group ID (GID). All users in the same group have the same GID value. When a user or group owns an event, another user or group cannot modify that event.

Others can view the event, but they cannot modify the event. The GID field becomes a useful field upon which to base a desktop filter. Users can use this field to filter events so that they see events only for their specific group. Supervisors can create filters for groups and can monitor the events that are assigned to those groups.

Creating a new group



© Copyright IBM Corporation 2014

Creating a new group

On this slide, you see a list of roles that a default ObjectServer provides. The user can create more roles. The following information provides a short description of each role:

- Administrator

This group definition contains the following roles: AlertsUser, CatalogUser, ChannelAdmin, DesktopAdmin, ToolsAdmin.

- AlertsGateway

This role, in combination with the CatalogUser role, provides the permissions that a gateway needs to generate alerts in the ObjectServer. Grant these permissions to any user that runs a gateway application.

- AlertsProbe

This role, in combination with the CatalogUser role, provides the permissions that a probe needs to generate alerts in the ObjectServer. Grant these permissions to any user that runs a probe application.

- AlertsUser

This role, in combination with the CatalogUser role, enables the user to show and manipulate alerts, create filters and views, and run standard tools in the event list.

- AutoAdmin

This role, in combination with the CatalogUser role, provides permissions to create automations in the ObjectServer.

- CatalogUser

This role includes permissions to view information about system, tools, security, and desktop database tables. Assign all groups this role.

- ChannelAdmin

This role includes permissions to set up channels for accelerated event notification.

- ChannelUser

This role includes permissions to receive and act on notifications for accelerated events that are broadcast over channels.

- DatabaseAdmin

This role, in combination with the CatalogUser role, provides permissions to create relational data structures in the ObjectServer.

- DesktopAdmin

This role, in combination with the CatalogUser role, provides permissions to customize the desktop.

- ISQL

This role, in combination with the CatalogUser role, includes permission to view ObjectServer data by using the SQL interactive interface.

- ISQLWrite

This role, in combination with the CatalogUser role, includes permissions to view and modify ObjectServer data by using the SQL interactive interface.

- Normal

This group definition contains the following roles: AlertsUser, CatalogUser, ChannelUser

- Public

All users are assigned this role. By default, the public role is not assigned any permissions. You can modify, but not drop the public role.

- SecurityAdmin

This role, in combination with the CatalogUser role, includes permissions to manipulate users, groups, and roles by using Netcool/OMNIbus Administrator or the SQL interactive interface.

This role also includes permissions to set properties and drop user connections.

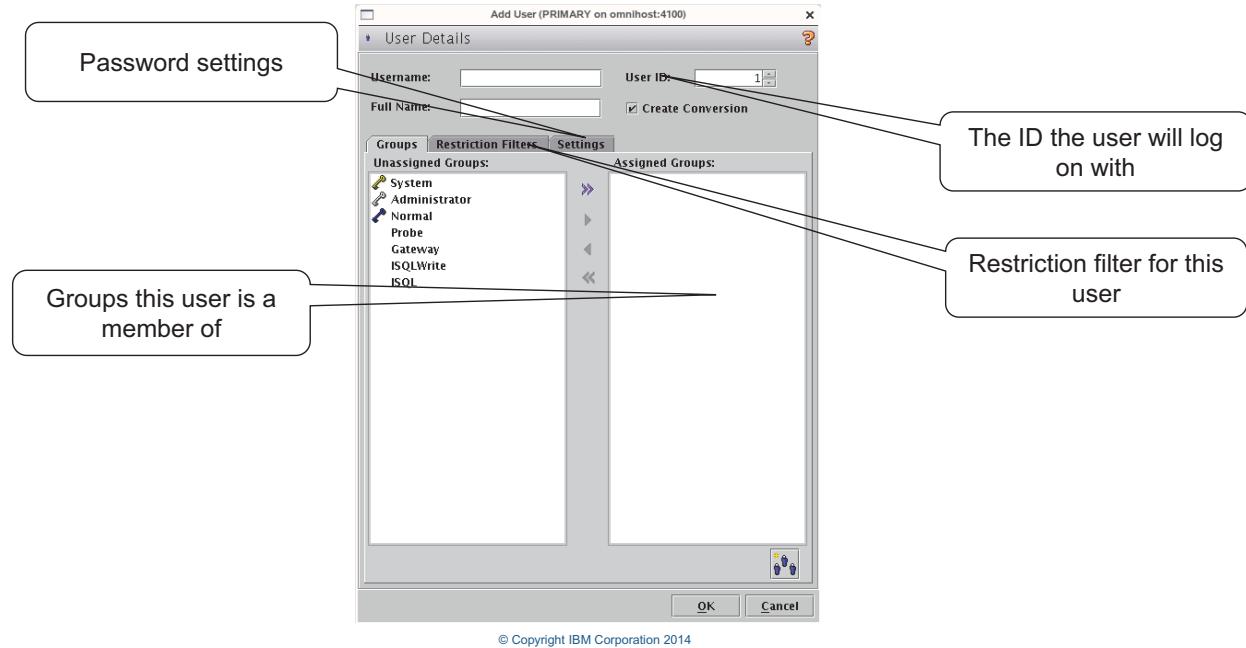
- SuperUser

This role has all available permissions. You cannot modify the SuperUser role.

- ToolsAdmin

This role, in combination with the CatalogUser role, provides permissions to create and modify tools that can be run from the desktop and Netcool/OMNIbus Administrator.

Adding new users



Adding new users

On this slide, you see the window that you use to create a new user ID.

There is a field in the upper right corner that is labeled **User ID** that shows the number 1. Note that the box that is labeled **Create Conversions** contains a check mark.

User IDs are stored in the ObjectServer as integer values (UID). When a user takes ownership of an event record, a column in the event table is updated with the integer that represents the specific user. When the event record is shown in a desktop, the owner field shows the textual value of the user ID. This translation happens automatically during the display process. The displayed textual value is the result of a *conversion*. There is a table within the ObjectServer that contains a list of the UID integer values and the corresponding textual values. The conversion is created automatically for a user when the box is checked for **Create Conversion**.

User administration when using LDAP

The point of administration for many product users is Dashboard Application Services Hub:

- Users that are added to LDAP are automatically available in Dashboard Application Services Hub
- The source of password authentication is the LDAP directory
- Users that are created or modified in Dashboard Application Services Hub are written to LDAP
- Users with Web GUI roles (ncw_admin or ncw_user) are created in the ObjectServer by the automatic synchronization process

© Copyright IBM Corporation 2014

User administration when using LDAP

An important consideration when using Netcool/OMNIbus is related to user administration. No hard and fast rule works for all situations. It is up to the Netcool administrator to decide how best to perform user administration.

These are the primary factors to consider:

- What type of desktop is best suited for the users. The Native Desktop or the Web GUI web-based desktop.
- The numbers of users that use each type of desktop.
- The source of password authentication.

Most users decide that they do not want to use the Native Desktop. The primary reason is because it is software that must be installed on user workstations. Most users are interested in the portability that the web-based desktop provides.

For those users who are primarily interested in using the web-based desktop, the primary point of user administration is Dashboard Application Services Hub. The Dashboard Application Services Hub represents a single point of administration that can accommodate access to multiple ObjectServers. When used with LDAP, this combination provides the best option for managing user access to Netcool/OMNIbus.

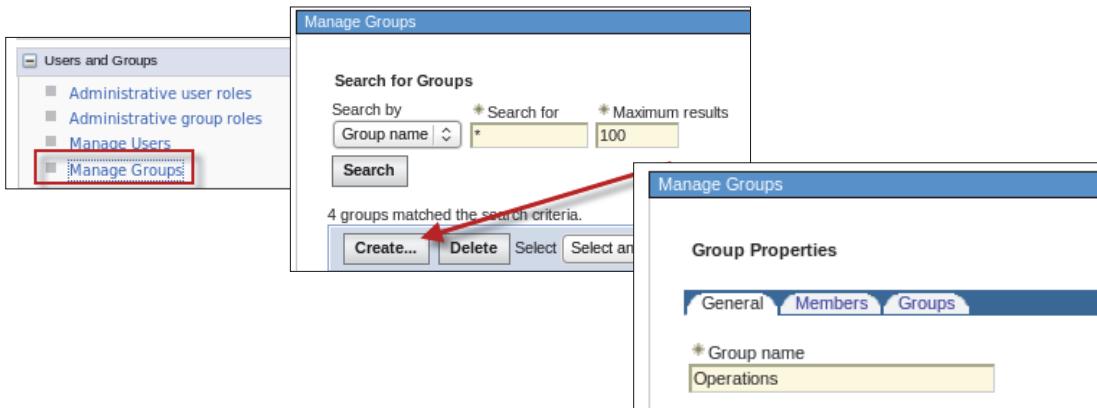


Note: LDAP is an option for user management, not a requirement. You can decide to perform all user management from Dashboard Application Services Hub, and keep the ObjectServer as the default user repository. The advantage to using LDAP is that the user information is already contained in the LDAP repository. Dashboard Application Services Hub merely reads the data from LDAP. Without LDAP, you must create users manually in Dashboard Application Services Hub, and the data is saved in the ObjectServer, which is why the number of users is a key consideration.

Creating groups

Groups are created with the WebSphere Administrative console

Must use the **smadmin** user



© Copyright IBM Corporation 2014

Creating groups

Dashboard Application Services Hub is based on WebSphere®. Part of the user administration is completed in WebSphere, and other parts are completed within Dashboard Application Services Hub. The administrative interface to WebSphere is the WebSphere Administrative Console. The administrative console is a web-based utility that you can access directly from Dashboard Application Services Hub.

Groups are created with the administrative console. Group names are text, but they cannot contain spaces or special characters except for the underscore.



Hint: When using LDAP, all group names within the LDAP server are available in WebSphere. You do not have to create them manually.

Creating users

Users are created with the WebSphere Administrative console

Must use the smadmin user

The screenshot shows three overlapping windows:

- Left Window:** "Users and Groups" navigation pane with "Manage Users" highlighted.
- Middle Window:** "Manage Users" search interface. A red arrow points to the "Create..." button.
- Right Window:** "Manage Users" details page for "User Properties". The "Groups" tab is selected. A red arrow points to the "Groups" tab. The "User ID" field contains "whill". The "First name" field contains "William" and the "Last name" field contains "Hill".

© Copyright IBM Corporation 2014

Creating users

Users are created with the administrative console. The user definition contains a user ID, first name, last name, and group membership.



Hint: When using LDAP, all user names in the LDAP server are available in WebSphere. You do not have to create them manually. Also, in a properly configured environment, any group relationship that is defined in LDAP is also available in WebSphere. With LDAP, you no longer have to manually create users and groups in WebSphere.

Assigning roles

Roles are assigned to groups or users

The screenshot shows the 'Roles' section of the DASH interface. On the left, there's a sidebar with icons for Groups, Roles, and Users. The 'Roles' icon is highlighted with a red box. Below it, three categories are listed: 'Group Roles', 'Roles', and 'User Roles'. A red arrow points from the 'Group Roles' section to a table below. The table has columns for 'Group Name', 'Role', and 'Unique Name'. It shows two entries: 'Netcool_Admin' and 'Netcool_User'. The 'Netcool_Admin' row is selected, indicated by a blue highlight. Another red arrow points from the 'Role' column of this row to a list of available roles on the right. This list includes 'iscadmins' (checked), 'monitor' (unchecked), 'ncw_admin' (checked), 'ncw_dashboard_editor' (checked), and 'ncw_gauges_editor' (checked).

© Copyright IBM Corporation 2014

Assigning roles

You assign roles in Dashboard Application Services Hub. You can assign roles to groups or users or both. The screen capture illustrates the roles that are assigned to the Netcool_Admin group.

When using LDAP, user and group management within WebSphere is eliminated. You must still manually assign roles to groups or users. If you have a well-designed collection of groups within LDAP, the group names are automatically available within Dashboard Application Services Hub. You need only to assign roles to the individual groups. After the roles are assigned, no additional changes are required unless the function of the group changes within the organization. User management can remain within LDAP. A new user that is created within LDAP and assigned to a known group is immediately available within Dashboard Application Services Hub. That user can log in to Dashboard Application Services Hub and has access to all features that relate to the roles that are defined within the group.

ObjectServer synchronization

Users with Web GUI roles (**ncw_admin** or **ncw_user**) are created in the ObjectServer by the automatic synchronization process

The image contains two screenshots of the ObjectServer configuration interface. The top screenshot shows the 'Groups' section with several entries: Administrator, Gateway, ISQL, ISQL Write, Netcool_Admin, Netcool_User, Normal, Operations, Probe, Public, System, and vmmusers. The 'Netcool_Admin', 'Netcool_User', and 'Operations' entries are highlighted with red boxes and labeled 'created by synchronization process'. The bottom screenshot shows the 'Users' section with entries: ncoadmin, ncouser, nobody, root, and whill. The 'root' entry is highlighted with a red box and labeled 'created by synchronization process'.

© Copyright IBM Corporation 2014

ObjectServer synchronization

Any user that requires access to Netcool/OMNibus event records requires an ObjectServer user record. ObjectServer users are required for either the Native Desktop or Web GUI desktop. When the ObjectServer is configured as the default user repository within Dashboard Application Services Hub, all user records are contained within the ObjectServer. You can add a user to the ObjectServer with the Administrator GUI, and that user is able to use the Native Desktop and Dashboard Application Services Hub. If you add a user to Dashboard Application Services Hub, the user record is written into the ObjectServer.

When LDAP is configured as the default user repository, any users in LDAP are not defined in the ObjectServer. If a user in LDAP does not have a record in the ObjectServer, the user can log in to Dashboard Application Services Hub, but cannot use any of the Web GUI features. Dashboard Application Services Hub roles control access to this feature. Access to the ObjectServer requires a user record in the ObjectServer. The user can open a Web GUI feature within Dashboard Application Services Hub, like the Active Event List, but the application fails to connect to the ObjectServer. An error message appears that indicates an authorization issue.

To facilitate user management when LDAP is used, there is an option to configure a timed synchronization process. The synchronization process periodically copies new Dashboard Application Services Hub users and groups to the ObjectServer. In this situation, user, and group records are duplicated. The records show in LDAP, and the same records also show in the ObjectServer.

The synchronization feature is disabled by default. You must manually enable it. You can also change the frequency of synchronization. The default is one time every hour.

ObjectServer, Web GUI, Dashboard Application Services Hub

ObjectServer users that the *administrator* utility creates are automatically added to Dashboard Applications Services Hub *if*:

- ObjectServer is defined as user repository in Virtual Member Manager realm

ObjectServer groups that the *administrator* utility creates are automatically added to Dashboard Applications Services Hub *if*:

- ObjectServer is defined as user repository in Virtual Member Manager realm

New Dashboard Application Services Hub groups and users are automatically added to the ObjectServer if the group or user has one or more ObjectServer roles assigned, ncw_user or ncw_admin, *and* one of the following statements is true:

- ObjectServer is defined as user repository in Virtual Member Manager realm

OR

- LDAP is defined as user repository in Virtual Member Manager and ObjectServer synchronization is enabled

© Copyright IBM Corporation 2014

ObjectServer, Web GUI, Dashboard Application Services Hub

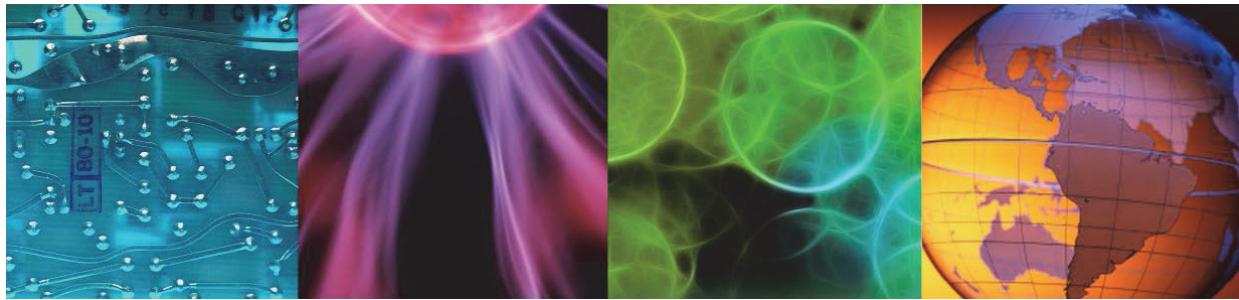
This slide contains a list of some of the relationships between the ObjectServer, Web GUI, and Dashboard Application Services Hub.



Lesson 4 Modifying the ObjectServer



Lesson 4 Modifying the ObjectServer



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to modify the structure of the ObjectServer. You learn how to add a database, a table, and a column. After completing this lesson, you should be able to perform the following tasks:

- Add a column to the alerts.status table.
- Add a database to the ObjectServer.
- Add a table to the ObjectServer.

Modifying the ObjectServer

You can modify the ObjectServer using nco_config

- Add new databases, tables, and columns
 - Changes take effect immediately (at run time)
- Can use new resources without restarting the ObjectServer
- The most common modification is to add columns to the event record (alerts.status table)

© Copyright IBM Corporation 2014

Modifying the ObjectServer

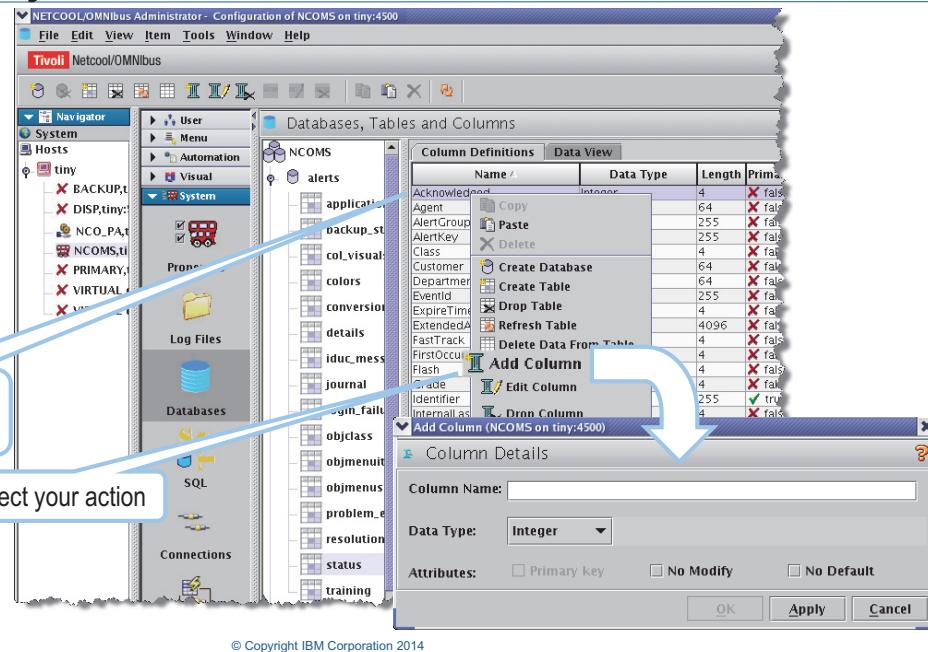
The Administrator utility is used to change databases in Netcool/OMNibus.

Here are some of the changes that you can make:

- Adding new databases
- Adding new tables
- Add new columns

You do not have to restart the ObjectServer before these resources become available.

Modifying the ObjectServer, continued



Modifying the ObjectServer, continued

To change an existing table, right-click and select your action from the list.

You have these options:

- Create database
- Create table
- Drop table, must be an empty table with no dependencies
- Refresh database
- Add column
- Edit column
- Drop column, any dependent object is dropped
- Refresh

Adding ObjectServer resources

Use nco_config

1. Add a new database called MyDataBase
2. Add a new table to MyDataBase called MyTable
3. Add two columns:
 - Text1 VarChar16
 - Int1 int, primary key
4. View the results

© Copyright IBM Corporation 2014

Adding ObjectServer resources

A table can be persistent or virtual.

- **Persistent:** When the ObjectServer restarts, the table is created with the old data present.
- **Virtual:** When the ObjectServer starts, the table is created with no data.

When adding columns, you can specify whether they are a primary key. More than one primary key can be specified.

However, when adding a column to a table that has existing data, you cannot make the column a primary key. The Primary Key push button is not active.

MyDataBase.MyTable

The screenshot shows the Tivoli Netcool/OMNibus interface. On the left is the Navigator pane, which includes sections for Hosts, System, and Visual. Under System, there's a section for 'My DataBase' which contains 'MyTable'. The main area displays the 'Databases, Tables and Columns' view. This view has two tabs: 'Column Definitions' (selected) and 'Data View'. The 'Column Definitions' tab shows a table with five columns: Name, Data Type, Length, Primary, and Null. The data is as follows:

Name	Data Type	Length	Primary	Null
Int1	Integer	4	X false	X
NewPrimary	Integer	4	✓ true	✓
RowID	Unsigned64	8	X false	X
RowSerial	Incr	4	X false	X
Text2	VarChar	1	X false	X

© Copyright IBM Corporation 2014

MyDataBase.MyTable

The screen capture illustrates a custom database called MyDataBase. That database contains a custom table called MyTable. That table contains the following columns:

- Int1
- NewPrimary
- RowID
- RowSerial
- Text2

Records can be added to this table by a probe, but the native desktop and Web GUI cannot view the contents.

Student exercises



© Copyright IBM Corporation 2014

Student exercises

Perform the exercises for this unit.

Summary

You now should be able to perform the following tasks:

- Navigate the Tivoli Netcool/OMNibus directory structure
- Describe the purpose of important directories
- Locate important files and binaries
- Describe a restriction filter
- Describe a group
- Describe a role
- Describe the requirements for an ObjectServer user
- Use the administration GUI to create an ObjectServer user
- Use the administration GUI to modify an ObjectServer
- Use the WebSphere Administrative console to define a group
- Use the WebSphere Administrative console to define a user
- Use Dashboard Application Services Hub to assign roles

© Copyright IBM Corporation 2014

Summary



6 Tools and automations



© Copyright IBM Corporation 2014

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

This unit provides an introduction to menus, tools, and automations. This unit focuses on creating a basic tool, modifying an existing automation, and creating an automation.

References: SC27-6265-00 *Administration Guide*

SC27-6505-00 *Web GUI Administration and User's Guide*

Objectives

In this unit, you learn to perform the following tasks:

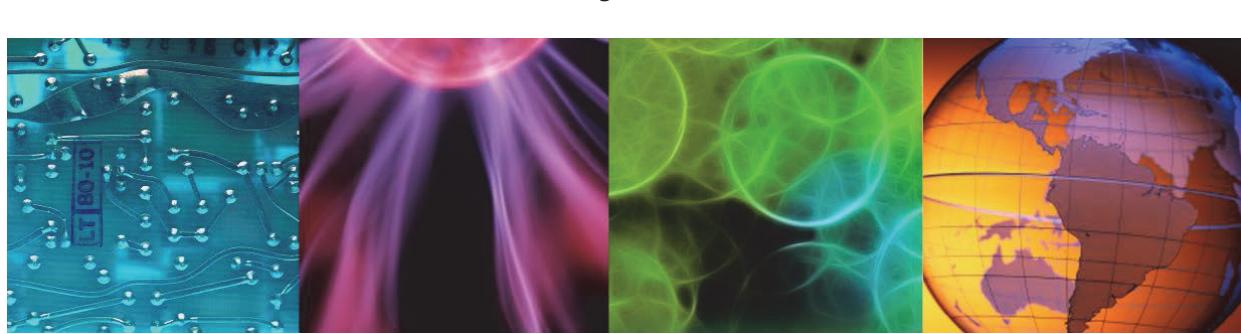
- Describe the basic features of Tivoli Netcool/OMNIbus SQL
- Describe the syntax of several of the most common SQL commands
- Use the **nco_sql** command
- Describe the types of desktop tools
- Create a simple desktop tool and add it to a menu
- Create a journal entry every time the tool is used
- Describe how trigger groups are used
- Describe the types of triggers
- Describe the functional elements of a trigger
- Use the ObjectServer configuration utility to modify trigger definitions
- Describe best practices regarding trigger configurations

© Copyright IBM Corporation 2014

Objectives



Lesson 1 Introduction to ObjectServer SQL



© Copyright IBM Corporation 2014

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

The ObjectServer supports a subset of ANSI standard SQL. In this lesson, you learn which commands the ObjectServer supports. After completing this lesson, you can describe the SQL commands that the ObjectServer supports.

ObjectServer SQL

Subset of American National Standards Institute (ANSI) SQL

Includes some proprietary extensions

Used throughout Tivoli Netcool/OMNIbus in automations, tools, and filters

Three general functional areas

- Data definition: ObjectServer structure and behavior
- Data manipulation: Automations, tools
- System administration: Command line

This module concentrates on *data manipulation*

© Copyright IBM Corporation 2014

ObjectServer SQL

ObjectServer SQL is a subset of American National Standards Institute (ANSI) SQL, which is used widely throughout Netcool®/OMNIbus. For example, it is used in the SQL file that creates the ObjectServer, and is used in automations in the ObjectServer.

ObjectServer SQL commands can be roughly divided into three functional areas:

- **Data Definition:** Use in the SQL file to define and create databases and tables.
- **Data Manipulation:** Use by automations, tools, and filters to retrieve, modify, and delete data.
- **System Administration:** Use on the command line to manage the system.

This lesson covers data manipulation commands, which are useful for automations and tools.

Accessing the ObjectServer with the command line

The **nco_sql** command provides access to ObjectServer on the command line

```
$OMNIHOME/bin/nco_sql -server <ObjectServer Name>
```

- Requires ObjectServer user name and password
- If not specified, root is the assumed user, initially root has no password

To safely shut down ObjectServer, issue these commands:

```
1> ALTER SYSTEM SHUTDOWN;  
2> go
```

ObjectServer shuts down gracefully with no data loss or corruption

© Copyright IBM Corporation 2014

Accessing the ObjectServer with the command-line

Command-line access to the ObjectServer is with the utility **nco_sql**:

```
nco_sql -server <OSNAME> -user name -password password
```

 **Note:** The ObjectServer user ID that you specify in the command must have the ISQL role at a minimum. If the command runs SQL commands that change database tables, the user ID also requires the ISQLWrite role.

From **nco_sql**, you can enter any SQL database commands against the ObjectServer in question. The following commands are useful:

```
delete from table-ref [where search-condition] delete a row  
insert into table-ref values [list of values] insert a new row  
select * from table-ref [where condition] return rows.  
select count(*) from table-ref [where condition] return number of rows
```

Use the following commands with the **nco_sql** utility to safely shut down an ObjectServer:

```
1> alter system shutdown;  
2> go
```

 **Hint:** The characters “1>”, and “2>” are not part of the command. The **nco_sql** utility generates those characters.

You can include nco_sql in a script to shut down the ObjectServer:

```
$OMNIHOME/bin/nco_sql -server NCOMS -user root  
-password fred << EOF > /dev/null 2>&1  
alter system shutdown;  
go  
EOF
```

Viewing data using SELECT

Selecting events

```
SELECT * FROM alerts.status WHERE Severity = 5;
```

```
SELECT *  
FROM alerts.status  
WHERE Severity = 5
```

retrieve all columns
from the alerts database and the status table
that meet this condition

© Copyright IBM Corporation 2014

Viewing data using SELECT

You use the SELECT command to retrieve one or more rows or partial rows of data from an existing table.

You can select an event:

```
select * from database.table where FieldName=condition;  
select *
```

Select all columns.

```
from database.table
```

From the database and table specified

```
where FieldName=condition
```

Based on the following condition

Example:

```
select * from alerts.status where Severity=5;
```

This command retrieves all columns from the alerts.status table and shows every record with a Severity=5.

Fields and operators

Selecting specific fields with multiple conditions

```
SELECT Summary, Class FROM alerts.status  
WHERE Severity = 5 AND Node = 'batman';
```

Inequality comparisons

```
SELECT * FROM alerts.status WHERE Grade >= 3;
```

LIKE operator for string comparisons

```
SELECT * from alerts.status  
WHERE Node LIKE '^bat.*';
```

LIKE is only used with char fields, often with regular expressions

© Copyright IBM Corporation 2014

Fields and operators

ObjectServer SQL supports the ability to retrieve specific fields or columns.

```
select Summary, Class from alerts.status
```

Retrieves only the Summary and Class fields.

ObjectServer SQL supports logical operators.

AND

OR

ObjectServer SQL supports comparison operators.

> greater than [or equal to] >=

< less than [or equal to] <=

<> Not equal to

LIKE and NOT LIKE are typically used in string comparisons and often with regular expressions.

The following meta-characters are the most commonly used:

. Match any single character, for example, link.n matches link2n, not link21n.

* Match none or more of the previous characters, for example, link* matches lin, link, or linkkk.

+ Match one or more of the previous characters, for example, link+ matches link, linkkk but not lin or linxk.

[] Match any single character within the specified range, for example, link[0-5] matches link2 but not link9.

^ Ensure that the pattern matches at the beginning of the string, for example, `.^link.*` matches `linknorth`, but not `northlink`.

\$ ensures that the pattern matches at the end of the string, for example, `.*link$` matches `northlink` but not `linknorth`.

In regular expressions, a backslash (\) escapes special characters (match literal value).

The NOT keyword inverts the result of any comparison.

IN operator and subqueries

Use IN to compare a value to a list of values

```
SELECT * FROM alerts.status WHERE Node IN ('batman', 'catwoman',  
'robin');
```

Can also use IN with subqueries

```
SELECT * FROM alerts.status WHERE Serial IN (( SELECT Serial FROM  
alerts.journal ));
```

The example retrieves all events with a journal entry

© Copyright IBM Corporation 2014

IN operator and subqueries

The IN list comparison operator compares a value to a list of values.

Example:

```
select * from alerts.status where Severity IN (1,3,5)
```

The query returns the rows in which Severity is equal to the number 1, 3 or 5.

With a subquery, the SQL in the subquery runs first, and the result is embedded in the command. Then, the remainder of the command runs.

Example:

```
SELECT * FROM alerts.status WHERE Serial IN (( SELECT Serial FROM alerts.journal  
));
```

The subquery is contained within the parentheses:

```
(( SELECT Serial FROM alerts.journal ))
```

This portion of the command runs first, and the result is embedded within the parentheses:

```
SELECT * FROM alerts.status WHERE Serial IN (( 899 ));
```

Modifying data using UPDATE

The UPDATE command changes table data

- Single field

```
UPDATE alerts.status SET Severity = 5  
WHERE Severity = 4 AND Acknowledged = 0;
```

- Multiple fields

```
UPDATE alerts.status  
SET Severity = 5, Service = 'Web Host'  
WHERE Grade = 4 AND Customer like 'ISP';
```

© Copyright IBM Corporation 2014

Modifying data using UPDATE

The UPDATE command modifies the contents of columns in an existing row of data in a table.

UPDATE statement updates a database.table set assignment where condition.

update

Update the records.

set assignment

Set the following assignment.

where condition

Where the following condition is true.

In this case, the asterisk or field name is not required. This statement updates the table that is defined, with the set assignment, which is based on the where condition.

For example:

```
update alerts.status set Severity=4 where Severity=3;
```

This statement first locates any record in the alerts.status table with a Severity of 3. It changes the Severity to 4 for every record found.

Creating data using INSERT

The INSERT INTO command adds a row to a table

```
INSERT INTO alerts.status  
(Node, Severity, Summary, Identifier)  
VALUES ('was1', 5, 'Disk Util 95%', 'was15DUT')
```

Removing data using DELETE

The DELETE command removes one or more rows from an existing table

```
DELETE FROM alerts.status WHERE Severity=0;
```

Deleted events cannot be recovered

© Copyright IBM Corporation 2014

Creating data using INSERT

The INSERT command creates a new row of data in an existing table. If you are not inserting values for every column in the row, you can specify a comma-separated list. This list has columns that are inserted within parentheses, followed by the VALUES keyword, followed by a comma-separated list of values within parentheses.

Insert Statement

```
insert into database.table (IntegerField, StringField, IntegerField2,  
StringField2) values (3, 'text', 3, 'more text');
```

String field values are single-quoted. You must specify a value for the primary key columns in the INSERT command. The optional UPDATING keyword forces the specified columns to be updated if the insert is deduplicated.

Example:

```
insert into status (Identifier, Severity, Tally, Serial)  
values ('MasterMachineStats15', 5, 12, 21)  
updating (Severity);
```

In this example, a new record is inserted into the alerts.status table. The new record has the following four fields specified:

Identifier: MasterMachineStats15

Severity: 5

Tally: 12

Serial: 21

If a record exists in alerts.status with the same Identifier, then deduplication occurs. During deduplication, only specific fields are updated, and Severity is not one of them. By including the **updating (Severity)** text, the ObjectServer is forced to update the Severity field. Without that text, the Severity field does not change.

Delete Statement.

```
delete from database.table where condition;
```

This statement deletes the rows from the table that match the specified condition, for example:

```
delete from alerts.status where Severity=0;
```

This statement removes every record from alerts.status that is green (Severity=0).



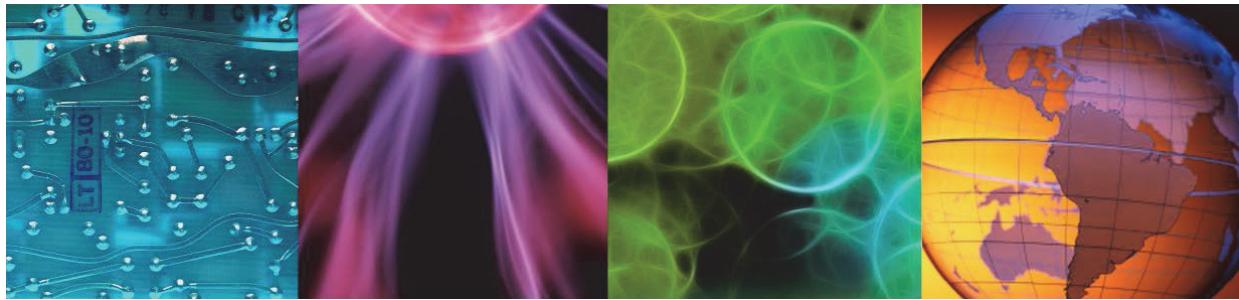
Note: The action takes place immediately, and the result is permanent.



Lesson 2 Desktop tools



Lesson 2 Desktop tools



© Copyright IBM Corporation 2014

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

In this lesson, you learn about desktop tools, and automations. After completing this lesson, you should be able to perform the following tasks:

- Describe the types of tools.
- Create a Web GUI tool.
- Add a Web GUI tool to a menu.

Native desktop tools

Configure tools in the ObjectServer using the *administrator* utility

When using multiple ObjectServers, for example, in high availability, you must define the same tool in both ObjectServers

Two types of tools are available:

- SQL: The SQL command runs within the ObjectServer
- Command: The command runs locally on the user desktop

Access tools from one or more menus

- Several menu options are available

© Copyright IBM Corporation 2014

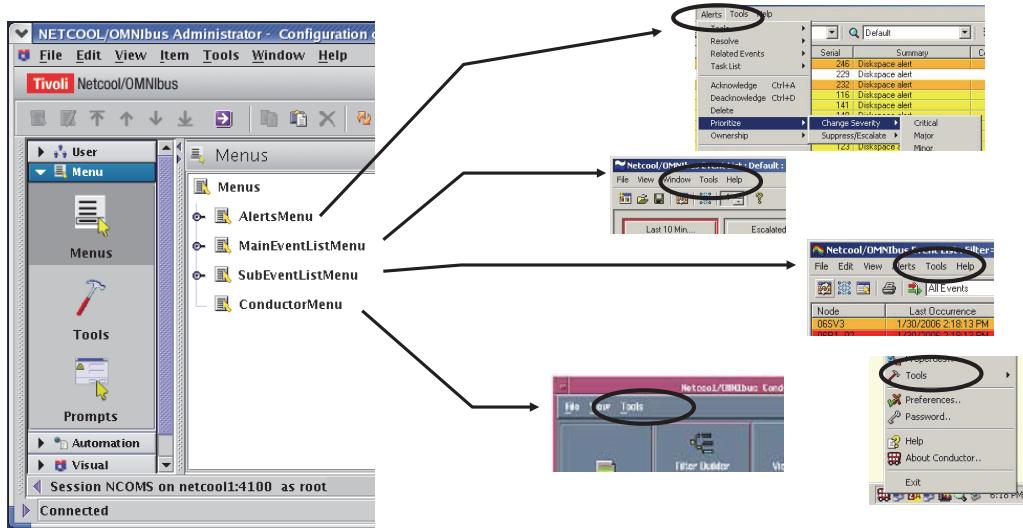
Native desktop tools

Netcool/OMNIbus provides two types of desktops: native desktop, and Web GUI. Both of which support tools. However, the tools for each type of desktop are not the same. Different utilities are used to create tools, and the definitions are stored in two different locations. The Netcool/OMNIbus Administrator utility manages native desktop tools. The tool definitions are stored in tables inside the ObjectServer. Because these tools are maintained in tables in the ObjectServer, if there are multiple ObjectServers in the environment, then the tools must be defined in both ObjectServers.

The native desktop supports two types of tools: SQL, and executable. When an SQL tool is run from the native desktop, one or more SQL commands run inside the ObjectServer. When an executable tool is run, one or more commands run locally to the desktop. The available commands vary based on the operating system that is hosting the native desktop.

Tools are accessed through a desktop menu. With the native desktop, there are several options for menu locations. The location of the menu determines where the user sees the tool.

Native desktop menu options



© Copyright IBM Corporation 2014

Native desktop menu options

Tools can be in the Main Event List, and SubEvent List menu bars. You can start tools from a selected event in the SubEvent List, the Alerts menu. Tools that launch from the SubEvent List can depend on the event class and user group.

When using Netcool/OMNibus to integrate management systems and legacy systems, tools that you use to solve a problem should be available on the same system. You can use tools to start applications, and run operating system scripts or commands. Tools can also modify databases with SQL commands.

A menu consists of the following items:

- Submenu shows another menu, creates a hierarchy or cascade effect.
- Separators place a horizontal line divider in the menu.
- Tools perform actions.

After an item is placed on a menu, it can be rearranged with the top, up, down, and bottom push buttons.

Web GUI tools

Configure tools within the Web GUI server

- Dashboard Application Services Hub user requires *ncw_admin* role

When using multiple ObjectServers, for example, in high availability, you must define the tool only once

- You can access all ObjectServers available to Web GUI with the same tool

Four types of tools are available:

- **SQL:** The SQL command runs within the ObjectServer
- **CGI/URL:** CGI script runs on the Web GUI server
- **Command:** The command runs locally on the user desktop
- **Script:** JavaScript command runs locally on the user desktop

Access Web GUI tools using the Active Event List (AEL) or the Event Viewer

© Copyright IBM Corporation 2014

Web GUI tools

Web GUI tools are defined within the Web GUI server. The application that administers these tools is accessible from Dashboard Application Services Hub. Access to that feature requires the *ncw_admin* Dashboard Application Services Hub role. With Web GUI tools, it does not matter if there are multiple ObjectServers in the environment, as the tool is defined in the Web GUI server. You can use a Web GUI tool on any ObjectServer defined to the Web GUI component.

There are four types of tools available to Web GUI users:

SQL: the same function as the native desktop SQL tool. When the tool is started, one or more SQL commands are run within the ObjectServer.

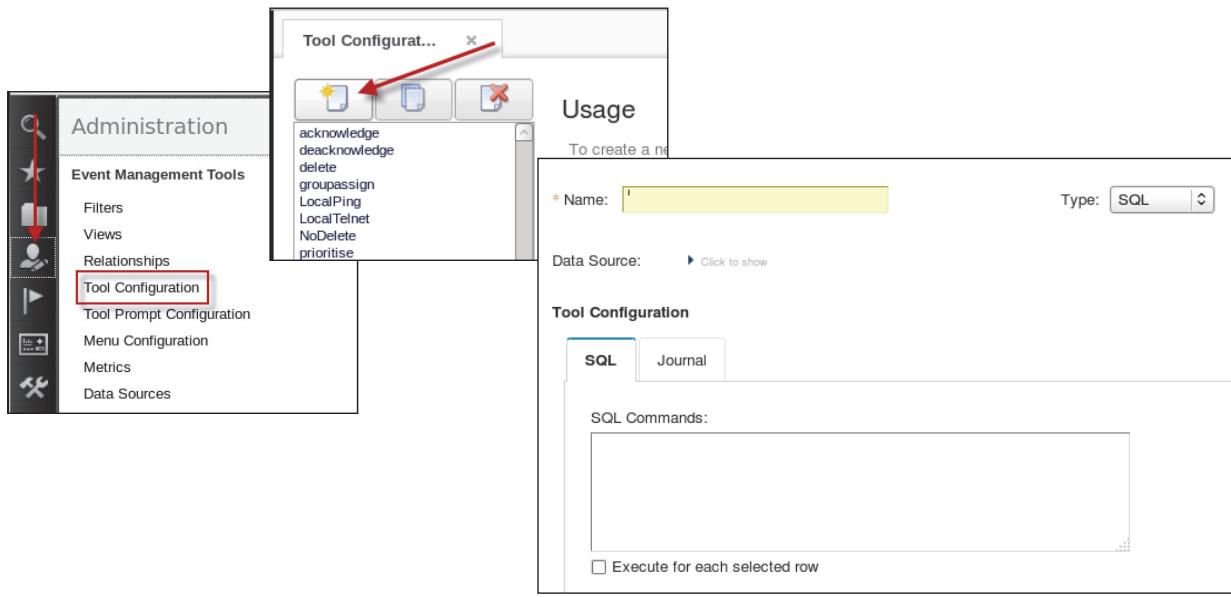
CGI/URL: the tool runs a CGI script, which is embedded in a URL statement. The CGI script runs locally to the Web GUI server.

Command: the same function as the native desktop executable tool. When the tool is started, one or more commands run locally to the workstation that hosts the browser that accesses Web GUI.

Script: the tool runs a Java script locally to the workstation that hosts the browser.

Web GUI tools are available from within the Active Event List and Event Viewer applications. There is only one option for a menu location with Web GUI tools.

Creating a Web GUI tool



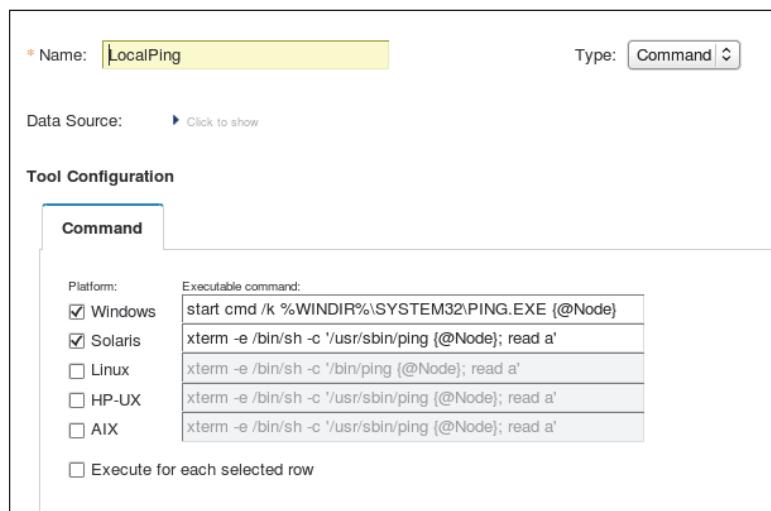
© Copyright IBM Corporation 2014

Creating a Web GUI tool

Select the **Tool Creation** page under **Event Management Tools** to create or modify a tool.

Click the icon that is shown in the screen capture to create a tool. Enter a name for the tool. Names cannot contain spaces or special characters, except underscore. Select the type of tool from the drop-down list. The **Tool Configuration** window changes based on the type of tool you select.

Command tool example



© Copyright IBM Corporation 2014

Command tool example

A Command tool consists of one or more commands that run locally on the workstation that hosts the browser that accesses Web GUI. The screen capture shows the LocalPing tool that comes with Web GUI. Observe that the tool is configured to be available for Windows, or Solaris workstations. If the user is using a Windows workstation, when the LocalPing tool is run, the following command is run on the Windows workstation:

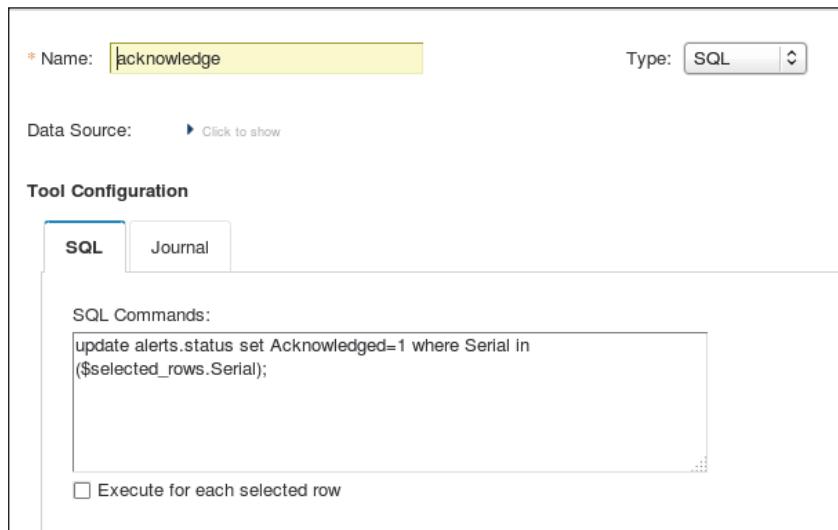
```
start cmd/k %WINDIR%\SYSTEM32\PING.EXE {@Node}
```

The tool is run in the context of an event. The user clicks an event record in the Active Event List window, right-clicks, and selects LocalPing from the menu. The tool opens a command window on the Windows workstation, and runs the **PING.EXE** program found in the **%WINDIR%\SYSTEM32** folder. The value of the Node column in the selected event is passed to the tool. The result is that the tool pings a specific device, for example:

```
ping server1.ibm.com
```

The output from the ping shows in the command window the tool creates.

SQL tool example



© Copyright IBM Corporation 2014

SQL tool example

The Netcool/OMNibus solution comes with a selection of prebuilt tools. There are tools for the native desktop included in every ObjectServer. Tools are also included with every instance of Web GUI server. Many of these tools provide the same function. That is, there is a delete tool for the native desktop, and a corresponding delete tool for Web GUI. Individuals that are new to Netcool/OMNibus might assume that these tools are the same tools. The tools provide the same functions, like delete, but they are not the same tools.

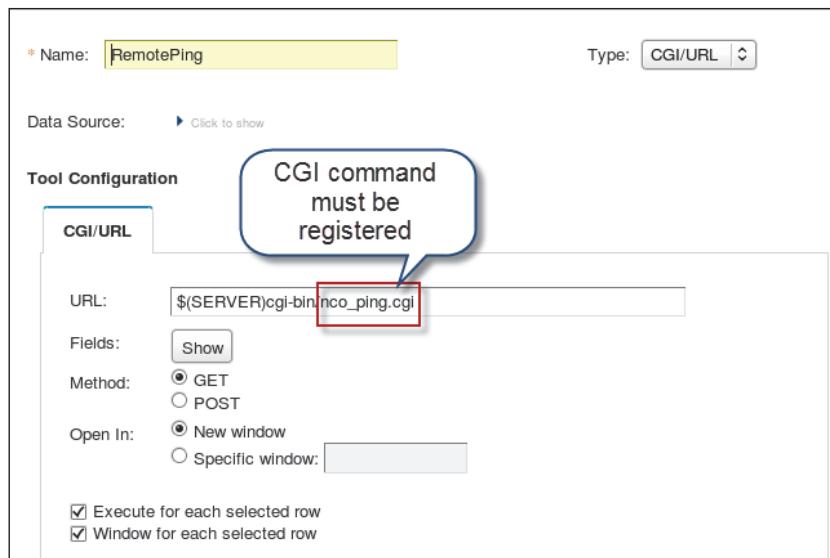
This screen capture shows the definition of one these common tools: acknowledge. Both tools run the same SQL command, and perform the same function. Remember that each tool is defined, and stored in an entirely different manner.

The acknowledge tool contains the following SQL statement:

```
update alerts.status set Acknowledged=1 where Serial in ($selected_rows.Serial)
```

This tool changes the value of the Acknowledged column to 1. The reference to \$selected_rows.Serial indicates that the tool is to change the value of the column for all *selected rows*. Most of the prebuilt tools are designed so that the user can select multiple event records in the event list, run the tool once, and have the tool modify all of the selected records.

CGI tool example



© Copyright IBM Corporation 2014

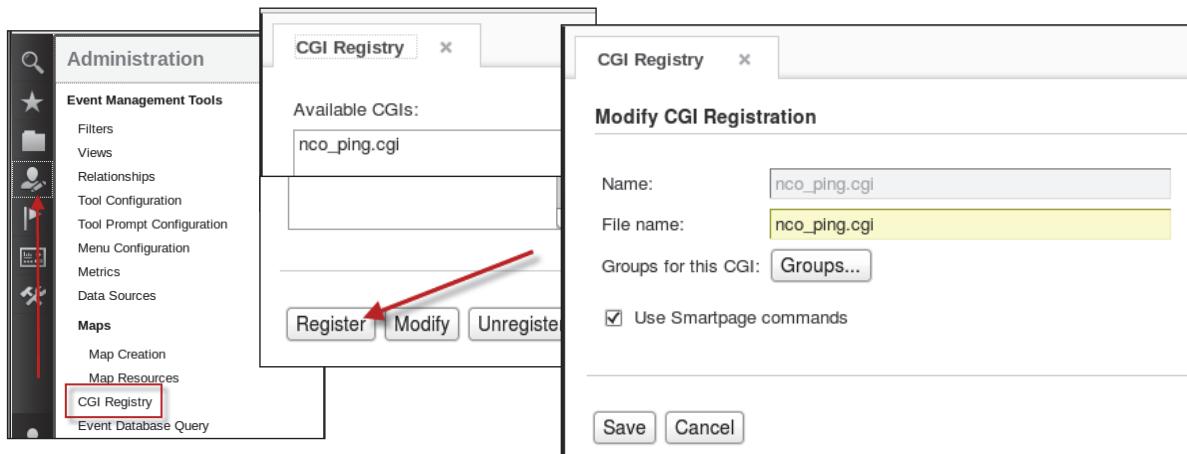
CGI tool example

CGI tools run a cgi script that is saved on the Web GUI server. The RemotePing tool that is shown in the screen capture, runs the nco_ping.cgi script. This script is stored on the Web GUI server in the **\$(SERVER)cgi-bin** folder. After the CGI tool is defined, you must register it before you can use it. The registration process is done with a Dashboard Application Services Hub application, as described on a subsequent slide.

Previously, you examined the SQL tool LocalPing. As the name implies, the ping command runs *locally* to the user.

The CGI tool that is shown here is RemotePing. In this case, the tool runs *remotely* from the user. It runs on the Web GUI server.

CGI command registration



- Save the CGI script on the Web GUI server in the <**webgui-home**>/etc/cgi-bin directory
- The support for CGI in Jazz for Service Management is provided by CGIServlet, extracted from Apache Tomcat

© Copyright IBM Corporation 2014

CGI command registration

Save the CGI script on the Web GUI server in the following location:

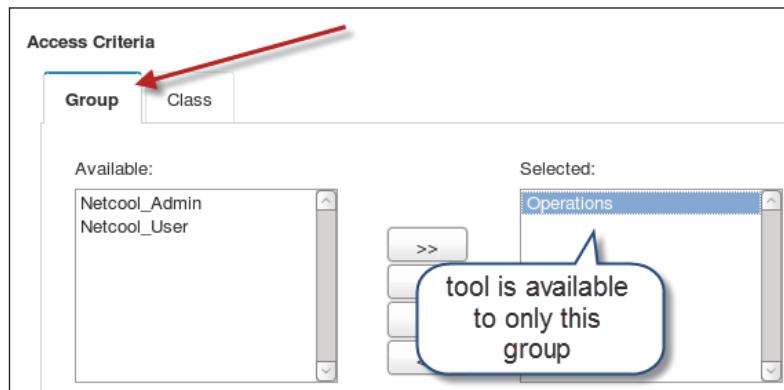
/opt/IBM/tivoli/netcool/omnibus_webgui/etc/cgi-bin

This location is the same location that is referenced in the RemotePing tool command as:

\$ (SERVER) cgi-bin

The registration window does not provide an option for specifying where to store the script because the scripts must be stored in a specific location.

Limiting tool access by group



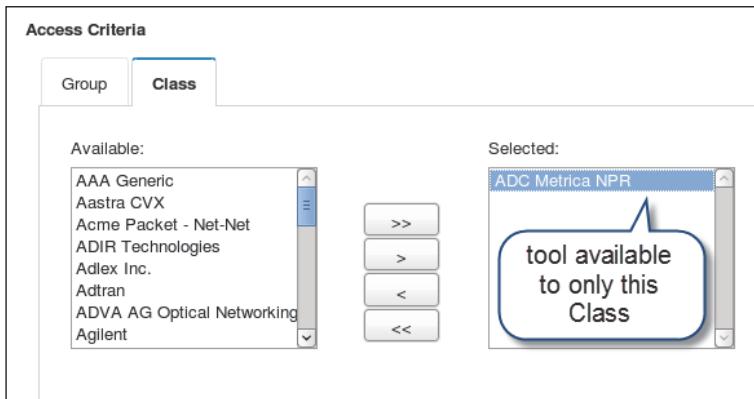
- All groups allowed access by default
- Adding a group to the selected column restricts access to the tool to only members of that group
- Restricted tools are not visible to other users

© Copyright IBM Corporation 2014

Limiting tool access by group

You can restrict access to desktop tools in two different ways: group and class. A tool with a group restriction is invisible to any user that is not a member of the selected group. The group restriction is convenient for preventing users from being able to run certain tools. For example, users that belong to the network support group want the ability to run a vendor-specific element management application to diagnose network device issues. They do not want other users to run this tool. Restricting access to the tool to the network group provides this capability.

Limiting tool access by class



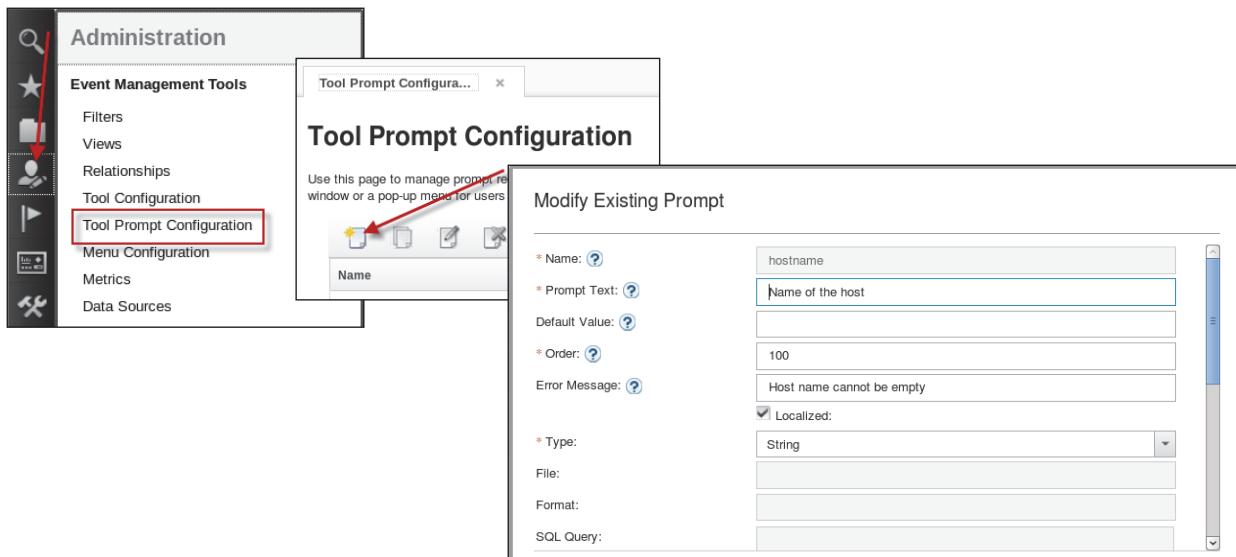
- All event classes have access by default
- Adding a class to the selected column defines the event class which is applicable to the tool
- The tool is accessible in the menu for any event that contains the selected class
- The tool is unavailable and gray in the menu for all other events

© Copyright IBM Corporation 2014

Limiting tool access by class

A tool with a class restriction is visible to all users. However, if the event that is selected does not contain the selected class, the tool appears gray in the menu and cannot be run. If the selected event does contain the selected class, the tool can run. This feature is used when creating a tool that is specific to a particular category of event. Many companies use vendor-specific tools to diagnose issues with equipment from that vendor. If the event records contain class values that are unique to the respective vendors, you can create tools to run each vendor's element management tool. By restricting access by class, the user can run the *Vendor X* tool for just *Vendor X* events. This capability facilitates network operations, and reduces problem-solving time. The creation of tools that contain a class restriction eliminate any possible confusion on the user's part. The user selects an event, right-clicks, and the tools that are available are the ones applicable to that device.

Creating a tool prompt



© Copyright IBM Corporation 2014

Creating a tool prompt

Many tools are configured so that one or more pieces of information are extracted from an event record when the tool is run. Other tools prompt the user to provide information. The process to create a tool that incorporates a prompt requires two steps. In the first step, you create the prompt itself. In the second step, you create the tool and incorporate a reference to the prompt.

There are several options for types of prompts, including but not limited to:

- String: the user can enter any textual characters.
- Fixed Choice: the prompt present a list of choices and the user selects one.
- Dynamic Choice: similar to the fixed choice except that the prompt choices are read from an ObjectServer table.

Using a prompt in a tool

The screenshot shows the 'Tool Configuration' screen for creating a new tool named 'PromptedPing'. The 'Type' is set to 'Command'. A callout bubble highlights the placeholder '\$prompt.<prompt name>'.

Tool Configuration

Command

Platform:

- Windows
- Solaris
- Linux
- HP-UX
- AIX

Executable command:

```
start cmd /k %WINDIR%\SYSTEM32\PING.EXE $prompt.hostname
xterm -e /bin/sh -c '/usr/sbin/ping [$prompt.hostname]; read a'
xterm -e /bin/sh -c 'bin/ping [$prompt.hostname]; read a'
xterm -e /bin/sh -c '/usr/sbin/ping [$prompt.hostname]; read a'
xterm -e /bin/sh -c '/usr/sbin/ping [$prompt.hostname]; read a'
```

Alert Group

Alert Group	Summary
Systems	Machine has gone offline
Stats	
Systems	Acknowledge
Link	De-acknowledge
Link	Prioritize
Systems	Suppress/Escalate
Link	Take ownership
Systems	User Assign
Link	Group Assign
Stats	Delete
Systems	Ping
ClientStatus	Telnet
MemstoreStat	Information...

Internal Command Parameters

Specify the required parameters below.

Name of the host

OK Cancel

A context menu is open over the 'Ping' item in the 'Systems' row of the alert group table. The menu items are: Ping from server, Prompted Ping, and .00. The 'Ping' item is highlighted with a red box.

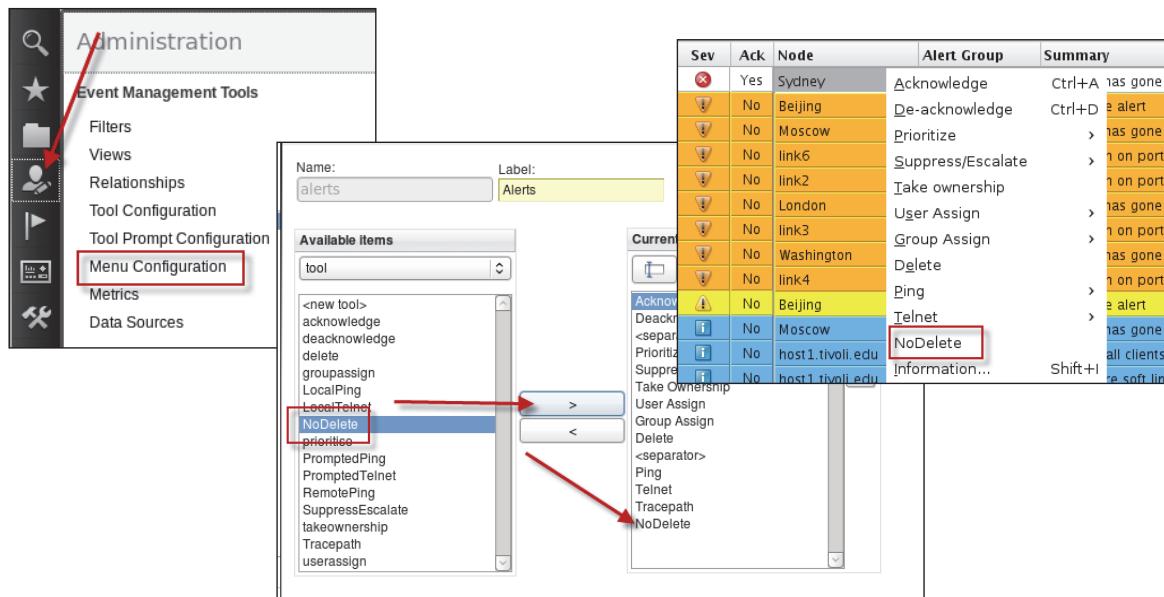
© Copyright IBM Corporation 2014

Using a prompt in a tool

After the prompt is created, it can be referenced in one or more tools. To reference a prompt, you include the following string within the tool:

```
$prompt.<prompt name>
```

Adding a tool to a menu



© Copyright IBM Corporation 2014

Adding a tool to a menu

You create tools and save them. Before you can use the tools, you must place them into a menu. You can use the same tool in different menus, and give it a different name in each menu.

You can change the content of the menus of the Active Event List (AEL) and Event Viewer. You can add tool entries to the menus, create new sub-menus, and modify or delete menu items. A tool can include a prompt window for the user to enter information, and you can edit these prompts or create new prompts.

There are two options for menu locations: Alerts, and tools. The Alerts and Tools menus are configurable menus that can be accessed from the AEL or Event Viewer. The Alerts menu contains a number of SQL tools you can use to interact with alert data and manipulate the data. By default, the Tools menu contains CGI tools and local, command-line tools.

The Alerts menu can be accessed from both the AEL toolbar and by right-clicking an event in the AEL. The Alerts menu is accessed from the Event Viewer only by right-clicking an event. You can access the Tools menu from the AEL toolbar. You can add your own tools and sub-menus to either the Alerts menu or the Tools menu. First, you use the tools editor to create the tools you want. Then, you add the tools to the menus with the menus editor.

You can configure tools in the Tools menu with access criteria that apply to users and events. Tools are visible only if the access criteria applied to them are met, or when no criteria are set because no groups or classes are defined. If multiple events are selected, all access criteria must be satisfied for all selected events for a tool to be shown. By default, no access criteria are defined for any tools. Tools that have no access criteria that are defined are shown for all users for all events. Changes in

access criteria take effect when the AEL or Event Viewer is reloaded, without the need to restart the Web GUI server.



Lesson 3 ObjectServer automations



© Copyright IBM Corporation 2014

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

In this lesson, you learn about the automation features of the ObjectServer. After completing this lesson, you should be able to perform the following tasks:

- Describe a trigger group.
- Describe a trigger.
- Describe a procedure.
- Create a trigger.

Trigger groups

Name	Group
audit_config	false
automatic_backup_system	true
compatibility_triggers	true
connection_watch	true
default_triggers	true
profiler_triggers	true
security_watch	true
stats_triggers	false
system_watch	true
trigger_stat_reports	true

Trigger group membership set in Trigger GUI

- Use trigger groups to manage multiple triggers
- Each trigger must belong to only one trigger group, but you can move the trigger between groups
- A trigger group can only be deleted if empty

© Copyright IBM Corporation 2014

Trigger groups

A trigger group helps organize triggers into functional groups, which when enabled or disabled, can alter the behavior of the ObjectServer.

A trigger group setting overrides the individual trigger setting. A trigger can be enabled, but it does not run if the group is disabled. However, if the group is enabled and the trigger is not, the trigger does not run.

In the administrator GUI, you can move a trigger among groups with the **Group** pull-down menu in the Trigger window. You can create or remove trigger groups from the **Trigger Groups** tab in the configuration GUI. However, a trigger group cannot be removed unless it is empty.

You can also manage trigger groups from the command line with the following command, an example of a command to move a trigger to a trigger group:

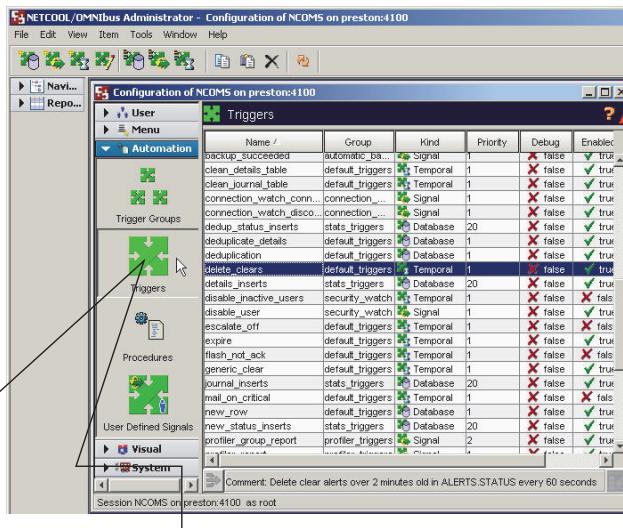
```
ALTER TRIGGER trigger_name
SET PRIORITY integer
SET ENABLED (TRUE | FALSE)
SET GROUP trigger_group_name
SET DEBUG (TRUE | FALSE)
```

Triggers

Use triggers for these purposes:

- Automatically manage events
- Perform escalation
- Perform correlation
- Perform external commands

In administration utility, expand Automation and click the Triggers tab



© Copyright IBM Corporation 2014

Triggers

Triggers, also known as automations, the names are interchangeable, are Netcool/OMNIbus features. They can change event data and run external actions automatically when certain conditions exist in the ObjectServer.

This list shows some examples of triggers:

- A trigger can delete events that have a severity of *clear* and are in the system for longer than a specified time.
- A trigger can inform a user or group if an event has a critical severity, but is not assigned or acknowledged after a specified time period.
- A trigger can change event data or run an external command if a combination of events occurs in the ObjectServer. For example, if unrecoverable events are received from the four major routers in your network, a trigger increases their severity or creates an event.

On the Automations page, you can view all triggers that are included with the product. You can create, edit, and drop triggers from this point.

How triggers work

- Three types of triggers:
 - Temporal executes on a time interval
 - Database executes on a database condition
 - Signal executes on a system or user defined signal
- All triggers have these fields:
 - **Settings** sets condition under which trigger executes time interval, database action, receipt of signal
 - **When** determines if the action should be executed now
 - **Evaluate** builds a read-only temporary table of signal and temporal triggers only
 - **Action** determines what is actually done
 - **Comment** for documentation

© Copyright IBM Corporation 2014

How triggers work

Triggers activate based on a specific condition, such as time interval, database action, or internal ObjectServer Signal, and perform a specified action.

There are three types of triggers:

- **Temporal**: A trigger that runs on a periodic basis, frequency. These triggers can periodically complete an action on the ObjectServer.
- **Database**: A trigger that runs if a specified condition (insert, update, delete, reinsert) occurs in an ObjectServer database table.
- **Signal**: A trigger that runs when a system-defined or user-defined signal takes place. Signals are configured separately and can monitor the status of Netcool/OMNibus processes.

Triggers are configured with these tabs:

- **Settings**
 - Trigger name: This value can contain only letters, numbers, and the underscore character.
 - Trigger-specific conditions: Time interval, database condition, or a signal
 - Group: Select from pull-down list.
 - Priority: Sets trigger fire order on simultaneous hits, 1=highest.
 - State: Debug, enabled
- **When**
 - More tests before running action, typically time-based

- **Evaluate**

SQL code to create temporary tables for action.

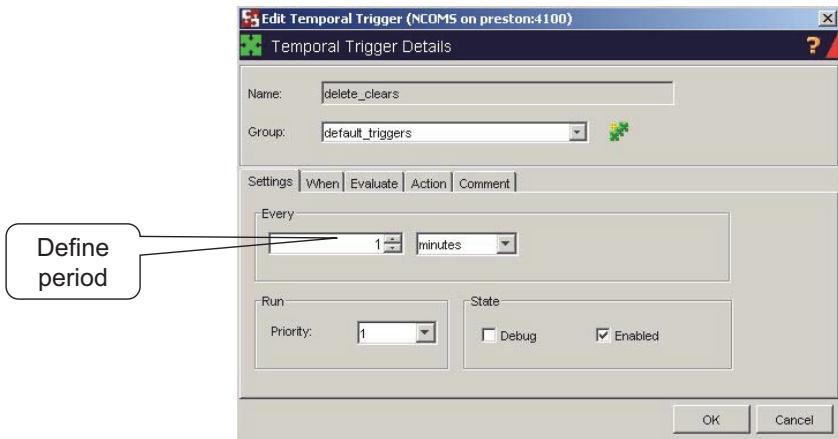
- **Action**

- Actions are SQL statements or calls to SQL or external procedures.
- External procedures require process control.

- **Comment**

Free-text documentation window

Temporal Trigger



Define period

© Copyright IBM Corporation 2014

Temporal Trigger

The **Settings** tab defines when the trigger runs. Temporal triggers are run on a frequency, which can be defined in hours, minutes, or seconds. The priority setting is found on the **Settings** tab. The priority setting should be different for triggers that run with the same interval, with 1 being the highest priority. It includes check boxes to enable the trigger and enable debug mode.

The **When** tab allows inserting an (optional) SQL condition to meet before the action is run. If a condition is in this tab, the action runs if the frequency and condition are both true. Here are examples of valid WHEN conditions:

It is not Saturday or Sunday.

```
(dayofweek(getdate()) <> 7) and (dayofweek(getdate()) <> 1)
```

Deduplication time is below a certain interval.

```
(new.LastOccurrence - new.FirstOccurrence) < 60
```

The **Evaluate** tab is not found in **Database Triggers** because they already act on a database table. You can use this tab to build a temporary result set from a single select statement to process in the trigger action. You can create a database query and bind it as a variable to use in the action statement.

The **Action** tab contains SQL statements that are run when the conditions specified in the **Settings** tab and **When** tabs are true. In this temporal trigger, the **Action** tab contains actions that are performed every minute. The **Action** tab can contain multiple SQL statements and procedural SQL constructs.

The **Comment** tab is a free-text documentation and description window that is stored with the trigger in the ObjectServer.

Temporal Trigger example

delete_clears trigger

SETTINGS	every 1 minute
WHEN	[not used]
EVALUATE	[not used]
ACTION	<pre>begin delete from alerts.status where Severity = 0 and StateChange < (getdate() - 120); end</pre>
COMMENT	Delete clear (Green) alerts over 2 minutes old in alerts.status every 60 seconds

© Copyright IBM Corporation 2014

Temporal Trigger example

The **delete_clears** trigger performs a basic housekeeping function, ensuring that cleared (Severity = 0) events are removed from the ObjectServer alerts.status table after a period of inactivity.

This trigger is a temporal trigger, so the **Settings** tab indicates that it runs every minute.

The **When** tab is empty in this trigger, indicating that it always runs.

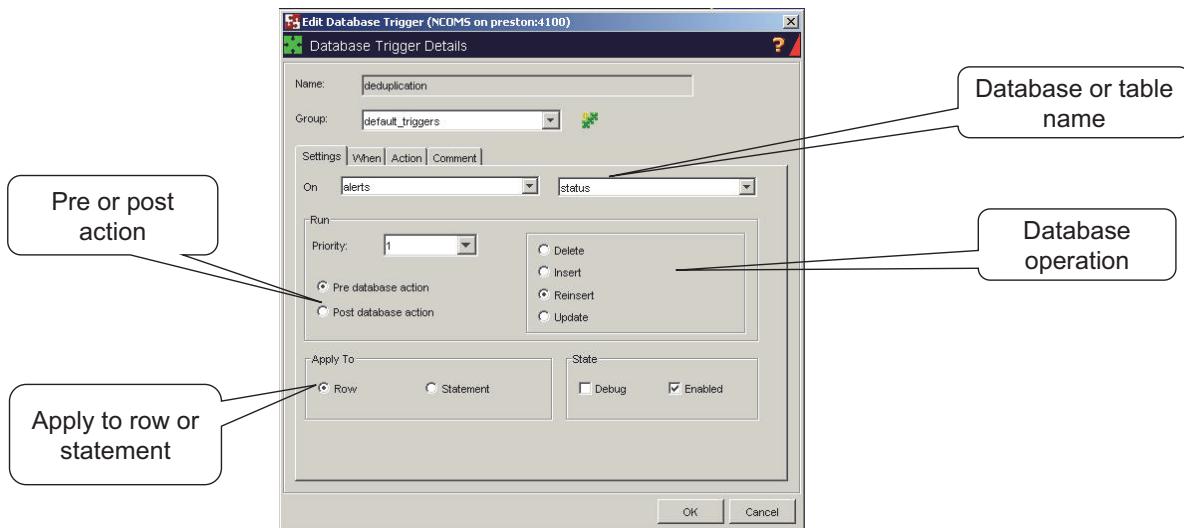
The **Evaluate** tab builds a temporary result set. This set is not needed in this trigger; so it is empty.

The **Action** tab contains the SQL statement that accomplishes the work of this trigger:

```
begin
    delete from alerts.status
    where Severity = 0 and
        StateChange < (getdate() - 120);
end
```

Although the user can modify any trigger, the best practice advice is to not modify any trigger that is bundled with Netcool/OMNibus. Instead, make a copy of the existing trigger and change the name. Then disable the existing trigger and modify the copy. Using a copy of the trigger ensures that any user modifications to triggers are not lost when Netcool/OMNibus is upgraded.

Database Trigger



© Copyright IBM Corporation 2014

Database Trigger

The following section describes the **deduplication** trigger.

The **Settings** tab is the only difference between the triggers. In a database trigger, the ObjectServer looks for a database operation to occur against a table rather than a time interval. The database operation can be delete, insert, reinsert, or update. The **Pre/Post Action** selector determines whether the action runs before or after the specified database operation.

Apply to Row/Statement, if set to row, the default, means that the contents of the **Action** tab run as many times as there are selected rows. When set to **Statement**, the action runs only once, regardless of how many rows are affected.

On the **Action** tab, everything that is previously learned applies. However, database triggers also have access to implicit variables, that the system automatically sets.

Row fields before change: `old.fieldname`, for example, `old.Severity`
Row fields before change: `new.fieldname`, for example, `new.Severity`

Note: In some operations, new or old row variables might not be available. For example, if a row is deleted, there is no new row to read or modify. The *Administration Guide* contains a table that illustrates when the new and old variables are available. The availability depends on the database operation.

Database Trigger example

deduplication trigger

SETTINGS before a reinsert action into alerts.status

WHEN [not used]

ACTION begin

```
set old.Tally = old.Tally + 1;
set old.LastOccurrence = new.LastOccurrence;
set old.StateChange = getdate();
set old.InternalLast = getdate();
set old.Summary = new.Summary;
set old.AlertKey = new.AlertKey;
if ((old.Severity = 0)and(new.Severity > 0))
then
    setold.Severity = new.Severity;
end if;
end
```

COMMENT Deduplication processing for alerts.status

© Copyright IBM Corporation 2014

Database Trigger example

The deduplication trigger is an example of a database trigger.

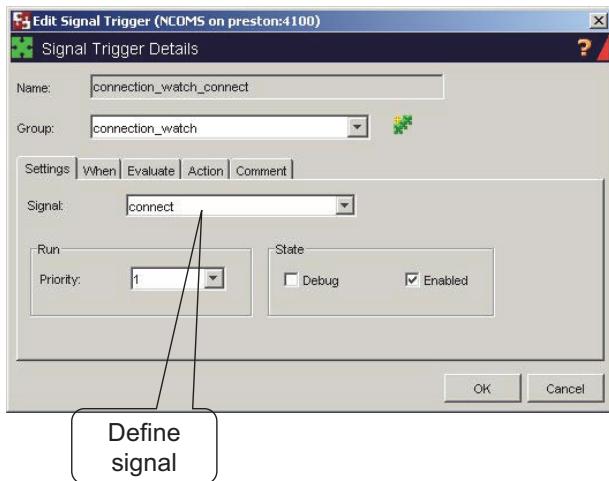
The **value in the Settings** tab indicates that it traps based on a Reinsert condition. A reinsert condition is an attempted insert into a table where the unique key exists.

The **When** tab is empty; so it always runs.

The **Evaluate** tab is missing in database triggers.

The **Action** tab updates fields on the existing event selectively, replacing the existing old values with the incoming new values. It also increments the tally, and updates the severity only if it is set to clear (0).

Signal Trigger



Two types of signals

- **System signals**
Indicate ObjectServer changes: startup, connections, and others
- **User signals**
User-created, raised using SQL
allow multiple actions against one trigger

© Copyright IBM Corporation 2014

Signal Trigger

The screen capture illustrates the **connection_watch** trigger. A **signal** is an occurrence in the ObjectServer that can be detected and acted upon. Signals can have triggers attached to them. The ObjectServer can then respond with a specific action when a signal is raised.

The ObjectServer raises system signals that are based on changes to the system, for example:

- System start, system shutdown
- Client connect, client disconnect, connection failure
- Backup success or failure

When a system signal is raised, attributes that identify the cause of the signal are attached to the signal:

%signal.at, %signal.server, %signal.node

You cannot delete or modify these attributes.

You choose the signal to run this trigger on the **Settings** tab. The signal can be a system or user signal.

The **When**, **Evaluate**, and **Action** tabs function as with the previous types of triggers covered.

Signal Trigger example

connection_watch_connect trigger

SETTINGS WHEN EVALUATE ACTION	Trigger on a connect signal [not used] [not used] <pre>begin if(%signal.description = '') then insert into alerts.status (Identifier, Summary, ... OwnerUID) values (%signal.process+'@'+%signal.node+' connected '+ to_char(%signal.at), ... 65534); else insert into alerts.status (Identifier, Summary, ... OwnerUID) values (%signal.process+':'+%signal.description+'@'+%signal.node+ 'connected '+to_char(%signal.at), ... 65534); end if; end</pre>
COMMENT	Create an alert when a new client connects

© Copyright IBM Corporation 2014

Signal Trigger example

The **connection_watch_connect** trigger inserts a new event into the ObjectServer when a client component connects.

The **Settings** tab indicates that it runs based on a connection signal.

The **When** and **Evaluate** tabs are empty. The trigger always runs, and no temporary table is built.

The **Action** tab might look complicated, the example that is given is shortened. However, the action inserts one type of event if the `%signal.description` attribute is null, and another type if it is not.

Signal triggers automate numerous system functions:

- Auditing

The triggers generate ObjectServer events when you make administrative changes, such as the addition of new fields in an ObjectServer table.

- Profiling

Triggers collect and report statistics about the performance of the ObjectServer.

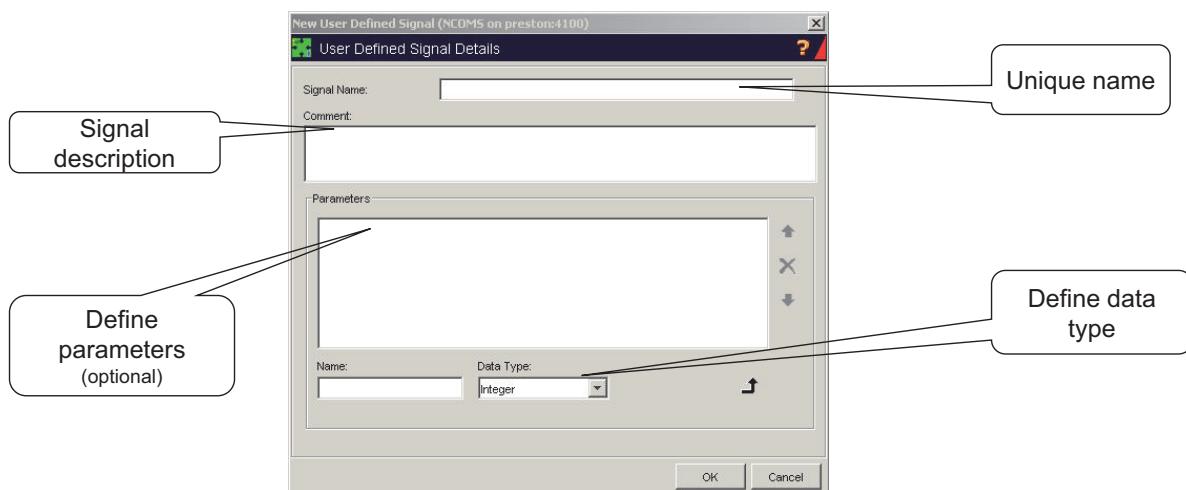
- Connections

Triggers generate events that are based on connects, disconnects, and connection failures.

- Failover and failback

Triggers implement controlled fail over and fail back in ObjectServer high availability configurations.

User Defined Signal



© Copyright IBM Corporation 2014

User Defined Signal

You can create user-defined signals within any signal trigger. Signal names must be unique and comply with naming conventions. User-defined signals are explicitly created, raised, and dropped. You can drop signals only if no other triggers reference the signal.

To raise a user-defined signal, you must use the RAISE SIGNAL command.

Syntax: RAISE SIGNAL signal_name expression

To drop a user signal, you use the DROP SIGNAL command.

Syntax: DROP SIGNAL signal_name

It is not currently possible to configure a trigger to run at a specific point in time, for example: Saturday, 01:00. A temporal trigger is defined with a frequency. Even if the frequency is set to 1 hour, the trigger might not run exactly on the hour. If the system that hosts the ObjectServer has a facility to run a command at a specific time, like the UNIX cron facility, it is possible to use a user signal to run at a specific time. You configure the operating system feature to run a script or batch file at the intended time. The script or batch file runs the nco_sql utility, which runs an SQL command to RAISE the user signal. When the signal is raised, a signal runs and performs the wanted function.

Event correlation

done using the generic_clear trigger

Correlate problem (Type = 1) events with resolution (Type = 2) events

Verify that both events are from the same

- Node - using Node field
- Interface - using AlertKey field
- Category - using AlertGroup field
- Probe - using Manager field

Correlation also must check in both events

- Problem or resolution - using Type field
- Time relationship - using LastOccurrence field
- Severity - using Severity field

© Copyright IBM Corporation 2014

Event correlation

Many situations in infrastructure management are problems that are later resolved. A simple example involves a problem with a network link. When the link fails, the system generates a *link down* condition, which indicates a problem. This condition typically produces a *critical* event in the ObjectServer. At some point, the link issue is resolved.

When the link recovers, the system generates a *link up* condition, which indicates that the problem is resolved. This condition typically produces a *clear* event in the ObjectServer. In Netcool/OMNIbus, a trigger correlates the two events, problem and resolution, and clears the problem event.

The **generic_clear** trigger standardizes correlation. It looks at the contents of the **Type** field, which is a generic indication of a problem or a resolution. The generic_clear trigger uses the Type column instead of looking for specific resolutions, such as link up, or port restored. The trigger correlates them with specific problems, link down, port fail.

For efficiency, the actual code that does the correlation looks complicated, but the principle is simple. For example, find all resolution (Type 2) events and match them with corresponding problem (Type 1) events. Check that they have identical values in certain fields (**Node**, **AlertKey**, **AlertGroup**, and others). Check that they have the correct time relationship, the problem occurs before the resolution. Then clear them.

Automation and trigger best practices

In a WHERE clause, compare integers first, then characters
Leave intensive comparisons and regular expressions for last

Ensure trigger does not catch events already managed, important for external scripts

Stagger trigger priorities so that they do not all run at the same time

Add a description for all automations

Tivoli Netcool/OMNIbus comes with standard triggers that you can change to meet customer needs

The *best practice* for modifying a trigger is to create a copy, disable the original, and modify the copy

► **ATTENTION:** You can modify automations, but changes can affect the running of the ObjectServer

Automation and trigger best practices

When a temporal trigger runs, it must read every event from the alerts.status table. Reading every ObjectServer event is the only way that the trigger can evaluate the WHERE condition to locate candidate events for the action statement. There are some guidelines to follow to help ensure that the triggers operate as efficiently as possible.

WHERE Clause

When constructing the expression in the WHERE clause that contains more than one condition, pay attention to type of field. Perform all integer testing first (Severity=5), character processing next (Node=XYZ), and regular expression testing last. The current version of Netcool/OMNIbus contains a feature that automatically optimizes these filter expressions. The system optimizes the filter for you.

Out Condition

When creating the filter condition, include criteria that ensures that the trigger does not select the same event over and over. This condition is important for those triggers that perform automated actions, like sending an email or a page.

Stagger Priorities

If you define multiple temporal triggers to run at the same frequency, every 30 seconds, stagger the priority settings among them. Staggering them ensures that they do not all run at the same time.

Description

Triggers are a powerful, and useful feature. Most users introduce new triggers periodically to resolve some operational situation. Over time, it might be difficult to determine why a trigger exists, and what it is intended to accomplish. Take the time to add a short description to any new triggers. Consider capturing details, such as when the trigger is created, the author, and why the trigger is created.

Procedures

Executable code called to perform common operations

Two types of procedures:

- SQL procedures manipulate data in an ObjectServer database
- External procedures run an executable on a remote system

Can call procedures from nco_sql, a trigger, or a tool

Syntax:

```
{EXECUTE|CALL} [ PROCEDURE ] procedure_name(expr,...);
```

Example:

```
EXECUTE PROCEDURE myproc();
```

or with parameters:

```
EXECUTE PROCEDURE myproc("text",@Node);
```

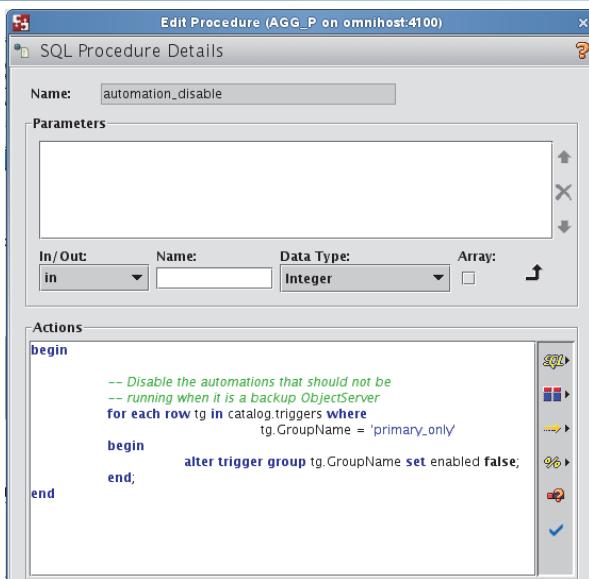
© Copyright IBM Corporation 2014

Procedures

Procedures are objects that you can call in SQL to perform an SQL operation or an external operation. A procedure is similar to a macro in a programming language. It is a prebuilt collection of code (SQL statements) that another process can call. You can adjust the behavior of the procedure, which is based on variables or parameters that you pass to the procedure when you call it.

The procedures are stored in an appropriate table in the ObjectServer. The tables are catalog.sql_procedures or catalog.external_procedures.

SQL procedure example



© Copyright IBM Corporation 2014

SQL procedure example

This screen capture illustrates the **automation_disable** SQL procedure:

```
begin

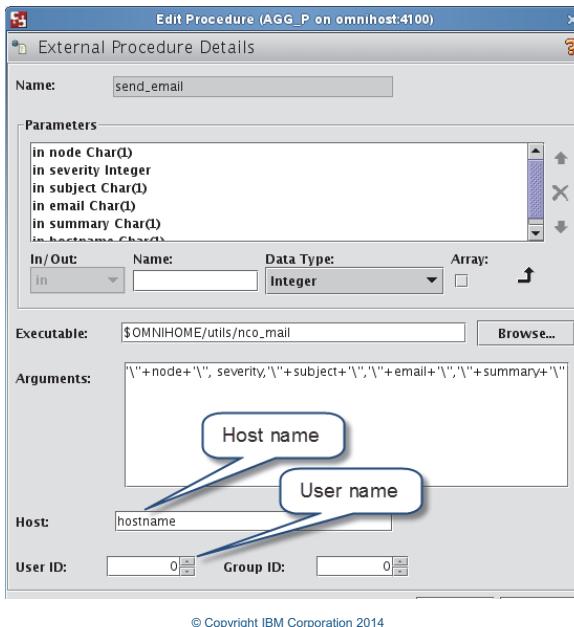
    -- Enable the automations that should be
    -- running when it is a primary ObjectServer
    for each row tg in catalog.triggers where
        tg.GroupName = 'primary_only'
    begin
        alter trigger group tg.GroupName set enabled true;
    end;
end
```

This procedure is called from the **backup_counterpart_up** signal trigger, as follows:

```
begin
    IDUC ACTCMD 'default', 'SWITCH TO PRIMARY';
    -- Disable the trigger groups that need not to
    -- run when object server acts as backup
    execute procedure automation_disable;
end
```

This procedure is one of the pieces of the failover and failback feature that manages the state of automations on the backup ObjectServer in a high-availability configuration.

External procedure example



External procedure example

This screen capture illustrates the **send_email** external procedure. There are four areas of interest on this slide:

- **Executable:**

`$OMNIHOME/utils/nco_mail`

This string is the command that is run when this procedure is called.

- **Arguments:**

`'\''+node+'\'', severity,\''+\subject+'\',\''+\email+'\',\''+\summary+'\''`

This string is the list of parameters that are passed to this procedure. The parameters that are defined in the Arguments window are passed on the command line to the executable. The number and location of each parameter in the arguments list is a function of the executable.

- **Host:**

`hostname`

The name or IP address of the system where the command runs. It does not have to be the local host.

- **User ID:**

`0`

The user ID value for the user that runs the command.

This procedure is called from the mail_on_critical temporal trigger as follows:

```
begin
    for each row critical in alerts.status where critical.Severity = 5 and
    critical.Grade < 2 and
        critical.Acknowledged = 0 and
        critical.LastOccurrence <= ( getdate() - (60*30) )
        begin
            execute send_email( critical.Node, critical.Severity, 'Netcool
Email', 'root@localhost', critical.Summary, 'localhost' );
            update alerts.status via critical.Identifier set Grade=2;
        end;
    end
```

Observe the execute statement and the values that are enclosed in the parentheses:

```
execute send_email( critical.Node, critical.Severity, 'Netcool Email',
'root@localhost', critical.Summary, 'localhost' )
```

The values in parentheses represent the list of parameters that are passed to the procedure. Most of the parameters in this example are taken from columns in the event record. Only two parameters contain hardcoded values.

Journal entry by automations

Create a journal entry in a trigger using the stored procedure: JINSERT

```
call jinsert( old.Serial, %user.user_id, getdate, 'This is my journal entry');  
Must change old.Serial to whatever is appropriate
```

Example using mail_on_critical trigger

```
begin  
    for each row critical in criticals  
        begin  
            execute send_email( critical.Node, critical.Severity, 'Netcool  
Email', 'root@localhost', critical.Summary, 'localhost');  
  
            update alerts.status via critical.Identifier set Grade=2;  
  
            call jinsert( critical.Serial, %user.user_id, getdate, 'Email sent  
via mail_on_critical Automation');  
        end;  
    end
```

© Copyright IBM Corporation 2014

Journal entry by automations

In a previous lesson, you learned about journal records. All of the desktop tools that come with Netcool/OMNIbus create journal records whenever they are used. You can also generate journal entries whenever an automation is run. To generate a journal entry when an automation is run, you add an SQL statement to the trigger action to call a procedure.

This screen capture illustrates how the mail_on_critical trigger can be revised to call the **jinsert** procedure to produce a journal entry.

Student exercises



© Copyright IBM Corporation 2014

Student exercises

Summary

You should now be able to perform the following tasks:

- Describe the basic features of Tivoli Netcool/OMNibus SQL
- Describe the syntax of several of the most common SQL commands
- Use the nco_sql command
- Describe the types of desktop tools
- Create a simple desktop tool and add it to a menu
- Create a journal entry every time the tool is used
- Describe how trigger groups are used
- Describe the types of triggers
- Describe the functional elements of a trigger
- Use the ObjectServer configuration utility to modify trigger definitions
- Describe best practices regarding trigger configurations

© Copyright IBM Corporation 2014

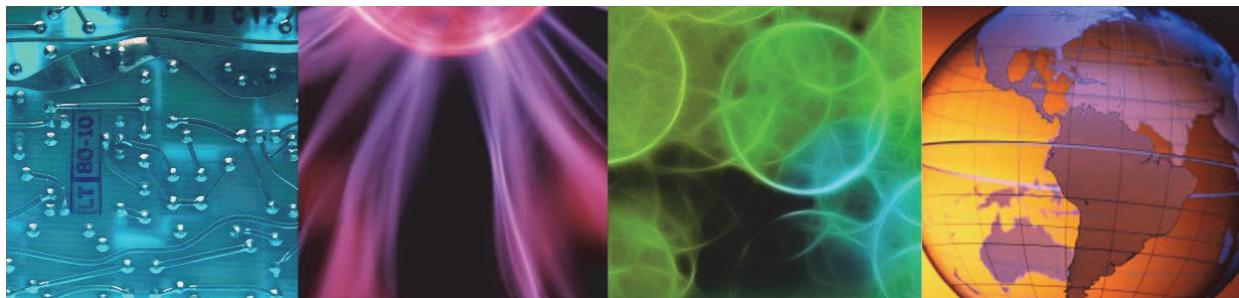
Summary



7 Common integrations



7 Common integrations



© Copyright IBM Corporation 2014

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

This unit is focused on installing and configuring the components that are used for most common integrations. In this unit, you learn how to install and configure the Syslog probe, and the SNMP probe. These probes are common probes that you use in most installations. They provide the basis for integrating applications that produce Syslog messages and infrastructure components capable of generating SNMP traps.

References: SC27-6266-00 *Probe and Gateway Guide*

SC23-7929-05 *IBM® Tivoli® Netcool®/OMNibus Syslog Probe Reference Guide*

SC11-7728-05 *IBM Tivoli Netcool/OMNibus SNMP Probe Reference Guide*

SC23-6386-13-00 *IBM Tivoli Netcool/OMNibus Knowledge Library Reference Guide*

Objectives

In this unit, you learn to perform the following tasks:

- Implement common integrations in Tivoli Netcool/OMNIbus
- Use a probe at the basic level
- Configure and modify probes
- Install and configure the Netcool Knowledge Library (NCKL)
- Install the UNIX Syslog probe and configure it to use the NCKL integration
- Explore the event rate detection capabilities
- Facilitate remote probe administration with available features

© Copyright IBM Corporation 2014

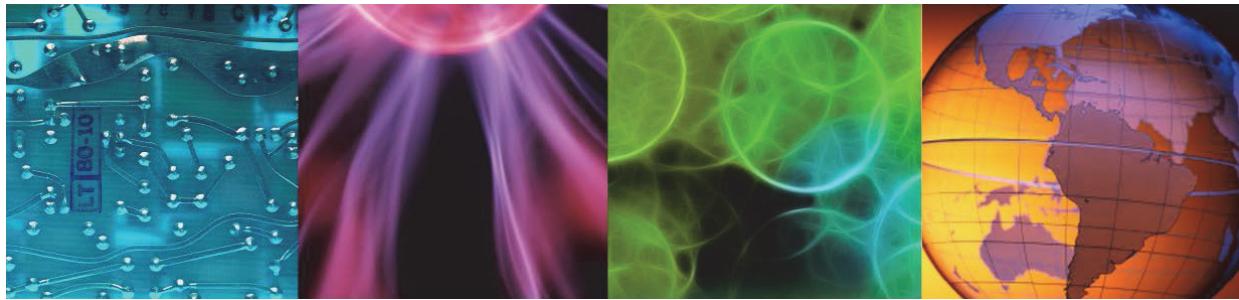
Objectives



Lesson 1 Overview



Lesson 1 Overview



© Copyright IBM Corporation 2014

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

In this lesson, you learn how probes process data and generate events. After completing this lesson, you should be able to perform the following tasks:

- Describe the operation of a probe
- Describe the function of the rules file
- Describe how to add customer-specific data to an event

Common integrations

The question most often asked:

Can Tivoli Netcool/OMNIbus manage vendor XYZ device?

Most common integrations handled using probes

- Syslog probe (UNIX/Linux)
- SNMP probe
- Tivoli EIF probe

Integrations implemented by use of the rules files

- Included with probes
- Netcool Knowledge Library (NCKL)
- Integrated Service Management Library (ISML)

© Copyright IBM Corporation 2014

Common integrations

The term, *common integration*, refers to the techniques commonly used in Netcool®/OMNIbus deployments. In general, integrations are implemented with probes and gateways.

Some probes and gateways are designed for specific integrations. These probes are deployed in many environments. However, a subset of probes is used in nearly every Netcool/OMNIbus deployment.

- Syslog probe (UNIX and Linux)
- SNMP probe
- Tivoli EIF probe

By using one or more of these specific probes, you can integrate nearly any component or application. The actual integration takes the form of a probe rules file.

The probe rules file interprets the vendor-specific or technology-specific data and converts it into the corresponding ObjectServer events. A user must know where to find the rules files that exist for a specific application or technology. There are three primary sources of prebuilt rules:

- Included with probe

Every probe comes with a default set of rules. These probes typically support a key number of vendors and technologies in the case of common probes.

- Netcool Knowledge Library

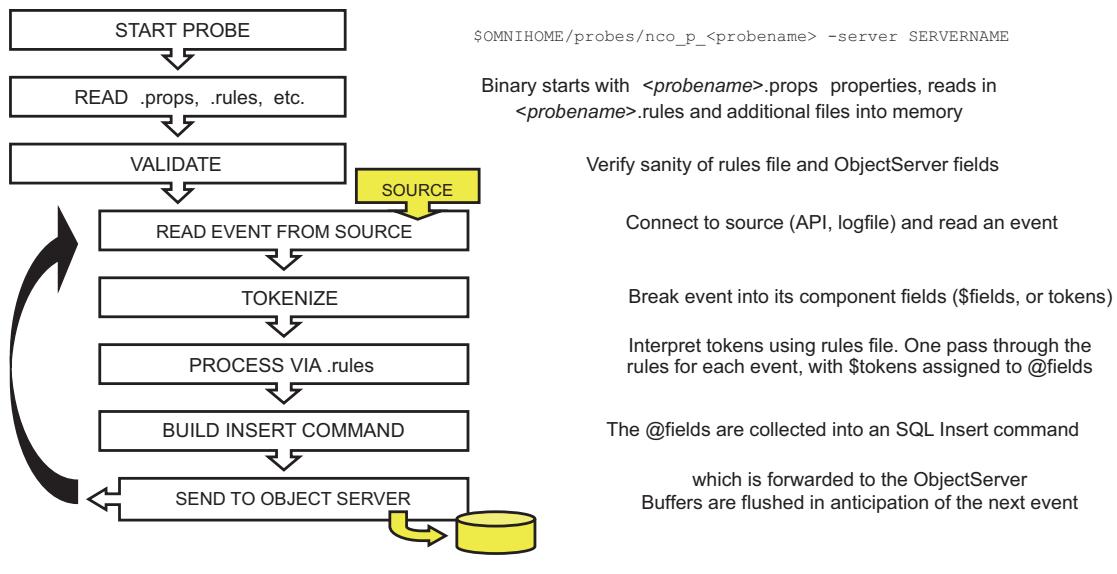
The Netcool Knowledge Library (NCKL) is a collection of prebuilt rules files that IBM® develops. These rules are designed to interpret the data that is produced by many vendors devices and technologies in use today.

- Integrated Service Management Library

The Integrated Service Management Library (ISML) is a collaborative effort to share integrations. The content of ISML comes from individuals such as IBM employees, Business Partners, and hardware vendors. Contributors provide integration packages (rules) that they develop. The integration packages are posted on the ISML site and made available for download by the Tivoli user community. ISML offers packages for all of the IBM software pillars.

Probe operation

A probe is lightweight software for collecting and preprocessing event data



Probe operation

A **probe** is lightweight, small-footprint software that obtains event data, converts it to a common event format (CEF), and passes it to the ObjectServer.

Probes are separate from the ObjectServer. As requirements change, probes can be added or removed without changing the ObjectServer or interrupting service.

Some probes are generic, for example, SNMP, and Syslog. Other probes are specific to applications or devices. Probes use a reliable TCP connection to the ObjectServer to ensure completeness and accuracy of data. If a probe loses contact with the ObjectServer, it can store events until the ObjectServer becomes available. If a pair of ObjectServers exists, a probe can be configured to fail over, sending events to the alternative ObjectServer.

Probe operation can be split into five stages.

- Initialization

The probe connects to the ObjectServer, identifying the format of the alerts.status table. The props and rules files are read into memory, parsed, and the probe is ready to retrieve events.

- Event Retrieve

The probe retrieves an event from the source, from API, SNMP trap, reading a log file, or similar.

- Tokenize

The probe breaks the input record into discrete fields to create tokens (\$ fields), which the rules file uses.

- Process

The tokenized event stream is parsed through the rules file, and ObjectServer alerts.status fields (@ fields) are set.

- Forward

The composed event is forwarded to the ObjectServer. If problems occur in forwarding, the probe might fail over or store and forward.

After forwarding the event, the probe clears variables, retrieves another event, and repeats the cycle as necessary.

The probe is a passive component. The probe receives data from a specific source, and then processes that data. The probe does not actively acquire data. It responds only to data that it receives. For example, the SNMP probe listens for SNMP traps, and the Syslog probe waits for a new entry in the UNIX Syslog file.

Probe basics

A probe consists of a **binary**, a **.rules** and a **.props** file:

Binaries retrieve and tokenise event streams (\$OMNIHOME/probes/nco_p_<probename>)

Properties run time settings of probe (\$OMNIHOME/probes/<arch>/<probename>.props)

Rules are instructions for processing event (\$OMNIHOME/probes/<arch>/<probename>.rules)

Probes can have additional files in \$OMNIHOME/probes/<arch>/

If the probe is on a separate machine from the ObjectServer

- Install common components, process control
- Install the probe itself
- Identify the ObjectServers in the interfaces file (with nco_xigen)

OBJ_SERV hostname 4100

- Edit the properties and rules files of the probe
- Run the probe

\$OMNIHOME/probes/nco_p_probename

© Copyright IBM Corporation 2014

Probe basics

All probes have at least three files: a binary file (**nco_p_<probename>**), a rules file (**<probename>.rules**), and a properties (**<probename>.props**) file.

The properties file sets run time parameters and determines the behavior of a probe. The behavior includes, among other parameters, where to record a log file and how verbose the log file messages are.

The binary collects the event stream and splits it into individual tokens. The binary interprets and applies the rules file, and forwards the processed event to the ObjectServer.

The primary purpose of the rules file is to assign tokens to ObjectServer fields. The rules file can also manipulate data, perform calculations, derive more data, and add it to the event. The rules file can derive more data with lookup tables and other methods.

When installing probes on a remote system, you must install the probes and common files on that system. These common files are installed with the Netcool/OMNibus core software when the Probe Support feature is selected.

Probes require access to the interfaces file to determine how to communicate with the ObjectServer. Use the nco_xigen utility to define the ObjectServer in the interfaces file.

To use failover, you must define a virtual ObjectServer pair. You learn about this topic in the unit on high availability.

Modify the properties, and rules file as necessary, and start the probe by running the following command:

```
$OMNIHOME/probes/nco_p_<probename>
```

Probe properties

You can set properties in these ways:

- With command line switches at probe startup
(**-help** , **-dumpprops** lists switches)
- In the probe **.props** file
- By not specifying either so that the default is used

Property formats:

command-line switch:

-server NCOMS

in the props file:

Server : 'NCOMS'

secure-mode properties:

-authusername root

AuthUserName: 'root'

-authpassword Lkfnjv0d

AuthPassword: 'Lkfnjv0d'

useful debugging properties:

-messagelevel debug

MessageLevel: 'debug'

-messagelog stdout

MessageLog : 'stdout'

© Copyright IBM Corporation 2014

Probe properties

All probes have standard properties, such as **-server** and **-name**. Some probes have extra custom properties that are specific to that probe. To list probe properties, use the following command:

\$OMNIHOME/probes/nco_p_probename -help

or

\$OMNIHOME/probes/nco_p_probename -dumpprops

You can set properties on the command line or in the property file. Property files are found in **\$OMNIHOME/probes/<ARCH>/** directory. Specifying properties on the command line is useful in testing, but for production, use the property file. Any overrides that are added to the property file are reflected when using the **-dumpprops** option.

The **MessageLevel** property sets the verboseness of the message log. In order of decreasing verboseness, the levels are debug, info, warn, error, fatal. The default is warn.

The **MessageLog** property sets the file to which to send debug messages, but you can set it to **STDOUT** to send to the screen. The default is **\$OMNIHOME/log/probename.log**.

You can set the debug for probe from the command line.

nco_p_probename -messagelevel debug -messagelog stdout

You can set the debug probe from the properties file.

MessageLevel: 'debug'

MessageLog: 'stdout'

You can run a probe with the MessageLevel set to debug when starting the probe for the first time or after changes are made. The debug level of messages provides useful information for determining whether the probe is operating correctly. Do not run the probe with this setting continuously. It imposes extra processing requirements on the probe and can potentially affect its performance.

Rules file

Rules files contain program steps, executed for each event, to manipulate incoming data and assign it to alerts.status fields

Found in \$OMNIHOME/probes/<arch>/<probename>.rules

Inbound tokens start with \$, ObjectServer Fields start with @

Major function of rules file is to define Identifier field

Field values of alerts.status can be set by the rules file

Can add additional information using the rules file

Probe can have multiple associated rules files (include files)

Netcool Knowledge Library (NCKL) uses this technique

- One master rules file
- Contains multiple `include` statements for individual vendor or technology rules files

© Copyright IBM Corporation 2014

Rules file

Not all data that a source record contains is relevant to the processing of that event. The rules file defines how the probe rationalizes or adds to the contents of an incoming event to create a meaningful Netcool/OMNibus alert. A key purpose of the rules file is to define the Identifier field that deduplication uses. If the identifier is made too specific, for example, by incorporating the time of the event, little deduplication takes place. However, if the identifier is not specific enough, for example by omitting a card, port, or slot number, the wrong events are deduplicated.

The probe uses the rules file to process the information from the incoming event, and populate the field values in the alerts.status table. You can add extra information to the event in the rules file, such as customer, department, and applications data.

Rules files can selectively update fields in alerts.status, overriding ObjectServer deduplication settings.

Testing and implementing new rules

nco_p_syntax checks rules file syntax

- Valid field names
- Command constructs
- Brace matching

```
$OMNIHOME/probes/nco_p_syntax -server NC0MS -rulesfile  
$OMNIHOME/probes/<arch>/myprobe.rules
```

To apply new rules to a running probe, restart probe process with SIGHUP

```
$kill -HUP <probe PID>
```

Note: **nco_p_syntax** is installed when the *probe support* feature is selected during the Tivoli Netcool/OMNibus core installation

Testing and implementing new rules

Test the syntax of a rules file with the Syntax probe:

```
nco_p_syntax
```

This method is more efficient and less disruptive than running a real probe.

The Syntax probe always runs in debug mode. It connects to the ObjectServer, tests the rules file, and sends error messages to the screen and exits. If no errors are shown, the syntax of the rules file is correct.



Important: The Syntax probe does not produce ObjectServer events. The probe verifies only the syntax of the rules file.

Rules files are read-only at probe start so that changes made to a rules file are not picked up dynamically. To implement changes to a probe that is running, the probe must be restarted or made to re-read the rules file. You use the *re-read the rules file* method because the probe does not lose events. You can force the probe to re-read the rules file by entering a **KILL -HUP PID** command on the probe process ID (PID). If you are making changes to nondeduplicating fields, delete any relevant existing data:

```
kill -HUP PID
```

where PID is the process ID of the probe process.

The kill HUP feature is useful, but you must consider a few issues. First, the kill HUP causes a running probe to re-read the rules file. It does not re-read the property file. Second, when the kill

HUP command is used, the probe re-reads the rules file and validates the syntax. If any errors are detected, the probe ignores the revised rules and uses the old copy that was cached in memory at start. Check the probe log file for any issues after using the kill HUP. Otherwise, the next time the probe stops, it does not restart because of the syntax error in the rules file.

To cause a running probe to re-read its property file, use the following command:

```
kill -USR2 PID
```

Where PID is the process ID of the probe process.

A rules file example

```
@Node=$Node
@Summary=$Summary
switch($Severity) {
    case "Critical":
        @Severity=5
    case "Major":
        @Severity=4
    default:
        @Severity=2
}
if (regmatch(@Summary,"interface.*down")) {
    @AlertKey=extract(@Summary,"interface (.*) is")
}
@Identifier = @Node + @AlertKey + @Summary
```

© Copyright IBM Corporation 2014

A rules file example

The screen capture illustrates a portion of the rules for the Simnet probe. The example is not the complete rules file. The following general statements apply to rules files:

- White space does not matter.

In the example, observe that some lines are indented, which improves readability, but does not affect the syntax of the file.

- Open and close braces can be placed anywhere.

As in other programming languages, the open curly brace ({) indicates the start of a code block. The close curly brace (}) indicates the end of the code block. The open curly brace can appear on the end of a line of code, for example:

```
switch($Severity) {
```

Or it can be placed on a line by itself. Both options are syntactically correct.

Observe the line that sets the value of the identifier column:

```
@Identifier = @Node + @AlertKey + @Summary
```

Setting the value for identifier is one of the primary functions of every probe rules file. The actual command to set the value varies from probe to probe.

Tokens and fields

Probes split the event stream into tokens

Creates all tokens as strings

Denotes tokens in the rules file with a \$ prefix

Example: \$Node is a token holding the node name

Can create tokens immediately `$MyElement=somevalue`

Denotes fields in alerts.status with an @ prefix

To populate fields, you assign values

- Direct assignment: `@Node = $Node`
- Concatenation: `@Summary = $Summary + $Group`
- Concatenation with literals:
`@Summary = $Node + " has problem" + $Summary`

© Copyright IBM Corporation 2014

Tokens and fields

When a probe receives an event stream from the source, it splits the stream into tokens. The set of tokens can change, depending on the event data received. The tokens are identified in the rules file by the dollar sign (\$) character. For example, \$Node is a token that contains the node name.

In the probe rules file, an ObjectServer field value is denoted with an at symbol (@). For example, @Node references the value of the **Node** field. Remember that the ObjectServer is case-sensitive; so all column names must be spelled exactly, including case.

When the probe starts, it connects to the target ObjectServer and verifies that all column names referenced within the rules file exist in the ObjectServer. If a column name is not found in the ObjectServer, the probe generates an error message and quits.

Testing and logic in a rules files

Tests for string values:

- `match(@Node, "router1")` Exact match
- `nmatch(@Node, "router")` Begins with
- `regmatch(@Node, "^router[0-9]")` Full regex matching

if statement incorporates conditional logic:

```
if (<test1>)
  { <action> }
else if (<test2>)
  { <action> }
else
  Example: set Class field based on the value of $EquipType
  { <action> }
if (match($EquipType, "Router")) {
  @Class = 3303
}
else if (nmatch($EquipType, "Switch")) {
  @Class = 3301
}
else if (regmatch($EquipType, "^[Hh]ub.*")) {
  @Class = 3302
}
```

© Copyright IBM Corporation 2014

Testing and logic in a rules files

Every rules file contains some type of testing logic. Probes process a stream of data from some event source. The probe breaks the stream into pieces, called tokens. It then evaluates the tokens to translate the information into ObjectServer columns. A good example is how a probe determines whether an event stream represents a problem or a resolution. In many cases, the probe identifies a problem by testing the contents of a token for the existence of a known string, for example: Up, down, offline, and others.

To facilitate this process, the probe rules language contains a rich set of commands that are used for various types of test logic.

switch/case statements

switch processes statements that exactly match the value of an expression

```
switch($EquipType)
{
    case "framerelay":
        @Class = 3303
    case "hub" | "switch" | "router":
        @Class = 3302
    default:
        @Class = 3300
}
```

default: must always be present, be careful with punctuation and braces

Example: Nesting switch and if

```
switch ($EquipType) {
    case "router":
        @Class=3303
        if (regmatch($Summary,".*offline.*"))
        {
            @Class=3304
        }
    default:
        @Class = 3300
}
```

© Copyright IBM Corporation 2014

switch/case statements

The **switch** statement is used to transfer control to a set of assignment statements, which are based on the value of an expression. The **switch** statement only tests for exact matches. Any other comparison must use an **if** statement. A **switch** statement is preferred over an **if** statement wherever possible. **Switch** statements are more efficient. The **switch** statements must contain the default **case**, even if there are no statements after it.

In the example that is shown, @Class is set to 3303 if \$EquipType is equal to frame-relay; @Class is set to 3302 if \$EquipType is equal to hub; and @Class is set to 3300 for all other \$EquipType values.

Because the **switch** statement is restricted to exact matches, you can combine **case** and **if** statements for greater flexibility.

In the example that is shown, matching is for \$EquipType equal to router, setting Class to 3303 for all cases except when the \$Summary field contains offline.

Control functions: Discard, recover, update, details

If an event is not required, it can be **discarded**

```
if (match($EquipType, "hub"))
{ discard }
```

Note: Be sure to use **discard** inside a test, or all events are discarded

A discard can be negated with a **recover**

```
if (match(@CustSLA, "Platinum"))
{ recover }
```

Event-specific deduplication is enabled with the **update** function

```
update(@Location)
```

You can insert into the details table with the **details** function

```
if (regmatch(@Summary, "Link .* down"))
{details($Card, $Slot)}
```

The **remove** function excludes a token from further consideration

```
if (match(@Node, "newdevice"))
{ remove($Slot)
  details($*) }
```

Note: Using **details(\$*)** impacts performance , use during testing only

© Copyright IBM Corporation 2014

Control functions: Discard, recover, update, details

If an event has no value, you can delete it with the **discard** statement. If the **discard** is not inside a conditional statement, all events are discarded. Discarding all events means that nothing is sent to the ObjectServer.

You can negate a **discard** with the **recover** statement. This statement is useful when the default process is to discard a type of event. In exceptional cases, the event needs to be retained.

In the example, all events with \$EquipType equal to hub are discarded, except for the events that have @CustSLA equal to platinum. Use the **discard** function with caution.

Deduplication is defined on a per-event basis, with the **update** function.

Details are extra data from the probe that is not stored in a field of alerts.status. Detail entries are added at the probe level, and are stored in the alerts.details table. Details are shown by double-clicking an event and selecting the **Details** tab. Details are added with the **details()** function only when an event is first inserted, not if an event is being deduplicated.

In the first example, the tokens \$Card, and \$Slot are stored in the details table when the event is a link down event.

In the second example, all tokens are stored in the details table if the event is from newdevice. The use of the **details(\$*)** function affects performance, and you should use it only during development and testing.

To selectively remove token elements, you can use the **remove()** function. This function is useful with **details(\$*)**.

Extract and exists functions

Sometimes required information is embedded in a string

```
$Summary = "Port is down on Port 1 Board 2"
```

Use the **extract()** function to extract the port value of 1

```
extract($Summary, "Port ([0-9]+)")
```

You can create temporary elements to build statements

```
$temp1=extract($Summary, "Port ([0-9]+)")  
$temp2=extract($Summary, "Board ([0-9]+)")  
@AlertKey = $temp1 + "." + $temp2
```

Use the **exists()** function to test whether a token is created
(not all tokens are created for every event)

```
if(exists($port))
```

© Copyright IBM Corporation 2014

Extract and exists functions

You can use the **extract** function to extract information from a probe token or field.

In the example on the slide, the probe creates the \$Summary token element from the event received. Board and port values are not available in any other token.

```
$Summary="The Port is down on Board 2 Port 1"
```

Board and port values need to be extracted from the token element and put in the @AlertKey field, which is important in deduplication and triggers. You use the **extract()** function. The **extract()** requires two arguments: the string to extract from, which can be a field or token, and a regular expression pattern that indicates what needs to be extracted. A pair of parentheses () denotes the information to extract. Any information in the innermost () parentheses is extracted into the field value. You can create temporary elements to implement the extraction.

```
$temp1=extract ($Summary, "Port ([0-9]+)")  
$temp2=extract ($Summary, "Board ([0-9]+)")  
@AlertKey = $temp1 + "." + $temp2
```

Testing the tokens with the **exists()** function is useful because not all events create all tokens.

When the **extract()** function fails to match the regular expression pattern to the string, it writes an error message to the probe log file. When processing unpredictable data with the **extract()** function,

you should test the validity of the data before using **extract**. A simple way to test is by using the **regmatch()** function:

```
if (regmatch(@Summary, "card.*enabled"))
{ $Card=extract (@Summary, "card (.*) enabled") }
else { log(WARNING, "Unexpected data: " + @Summary) }
```

Lookup tables

At the top of the rules file, place a file reference or the table itself

File Reference: `table name ="$OMNIHOME/probes/<arch>/file"`
`file` should look like: `key[tab]value`
 `key[tab]value`

Table in Rules: `table name = {{"key", "value"}, {"key", "value"}}}`
where `key` - item looked up, `value` - item returned.

Then, to use the table

`@result = lookup(key, tablename)`

where `@result` is assigned by searching for `key` in `tablename`

For example, a rules table would have this definition at the top:

`table DeptTab = {{"batman", "Technical"},
 {"robin", "Finance"} }`

And the lookup occurs when you execute this command:

`@Department=lookup(@Node,DeptTab)`

© Copyright IBM Corporation 2014

Lookup tables

Lookup tables provide a method of making extra information available to a probe, and for inclusion in an event. A lookup table can be defined two ways. You can define it as a reference to an external file that contains the table, or by placing the table in the rules file itself.

For an external file table, at the top of the rules file, specify a pointer to the file table. The top of the rules file is the first uncommented line, above the ProbeWatch section.

`table TabNam="/opt/netcool/omnibus/probes/<arch>/file"`

The file should have the following format:

`key [TAB] value`
`key [TAB] value`

For a lookup table embedded in the rules file, the definition takes the following format:

`table TabNam={{"key", "value"}, {"key", "value"}...}`

In the rules file, the `lookup()` function looks the same for both table types. It uses two arguments, a value to look up, and the name of the table to look up:

`@result=lookup (@Key, TabNam)`

In this example, `lookup()` tries to find the `@Key` data in the key field of the table `Dept` and return the corresponding value to be stored in `@result`. If the key is not found, a null value is returned.



Note: `@Key` and `@result` are not real ObjectServer fields; they are used here as examples.

Multicolumn lookups and defaults

Multicolumn lookup tables return more than one value per key

In a rules table:

```
table xnodes = {{ "mial", "Miami", "Marketing" },
                { "bos7", "Boston", "Finance" } }
```

In a file table:

```
mial [TAB] Miami [TAB] Marketing
bos7 [TAB] Boston [TAB] Finance
```

Field assignment:

```
[@Location, @Department] = lookup(@Node, xnodes)
```

default sets values returned when key is not found

Add default values immediately after the table definition

In a single column lookup table

```
table DeptTab = "/tmp/dept.lookup"
    default = "UNKNOWN"
```

In a multicolumn lookup table

```
table DeptTab = "/tmp/dept.lookup"
    default = { "UNKNOWN", "UNKNOWN" }
```

© Copyright IBM Corporation 2014

Multicolumn lookups and defaults

A lookup table can return more than one value per key:

- **Rules Table:** The table function is extended by adding the additional values for each key. To set both the location and the department, use the following table definition:

```
table deptloc = { { "batman", "London", "Technical" },
                  { "robin", "New York", "Sales" } }
```

- **File Table:** In a file table, more columns are added separated by tabs. To set the location and department fields, use the following table format:

```
batman [TAB] LONDON [TAB] Technical
robin [TAB] New York [TAB] Sales
```

- **Field Assignment:** To set the fields location and department, use the following syntax:

```
[@Department, @Location] = lookup(@Node, tablename)
```

If you define multiple columns in a table, use that many fields in the assignment. The column count must be consistent for each row in the table.

With the **default** option, a value can be returned when an event does not match any of the key values in a table. The default must follow the specific table definition. For a multicolumn table, you must provide a **default** for each column.

Properties and writable properties

You can access the values of probe startup properties in rules

```
%Manager, %MessageLevel, %MessageLog ...
```

Useful for host-specific processing or debugging probe rules

A writable property (persistent variable) is created if it does not exist

Writable properties retain their values across different events and when probe rereads rules, but not when cold started

Example uses: total events counter, discarded events counter

```
if (match(%TotalEvents) , "") { %TotalEvents=1 }
else
{ %TotalEvents=int(%TotalEvents)+1 }
@RunningTotal = %TotalEvents
```

© Copyright IBM Corporation 2014

Properties and writable properties

You can reference probe start properties as values in the probe rules file. This reference is useful when you perform conditional processing in the probe environment. Typical examples might be to perform actions that are based on the host the probe is running on or on the message level the probe is running under. In the example, you are storing all details and modifying the Summary if the probe is running under debug.

You can dynamically create new properties and retain their values through subsequent iterations of events, unlike tokens, which are local to each event. The values are held in memory and retained if the probe is hopped (kill -HUP). They are lost if the probe is restarted. Properties are stored as strings. Type conversion is necessary to perform mathematics on them.

The following example illustrates two scenarios that use writable properties.

- Total events counter

```
if (match(%TotalEvents) , "") { %TotalEvents=1 }
else
{ %TotalEvents=int(%TotalEvents)+1 }
@RunningTotal = %TotalEvents
```

- Discarded event counter

```
if (match(@Node,"testnode")) {  
    discard  
    %DiscardEvents = int(%DiscardEvents)+1  
    if int(%DiscardEvents) > 100 {  
        log(WARN, "Review discard criteria")  
        %DiscardEvents = 0  
    }  
}
```

Associative arrays

Store and use data in dynamic arrays in rules

- Retain values when rules are reread, but not when cold started
- Array elements are keyed by a string; array must be declared

Example: Calculate interval between similar events

```
# Declare array at top of rules file
array prevtime
# First time round initialize array element
if (match(prevtime[@Identifier], ""))
{ prevtime[@Identifier] = 0 }
# Calculate and store interval
@Interval = int(@LastOccurrence) -
            int(prevtime[@Identifier])
# Update array element
prevtime[@Identifier] = @LastOccurrence
```

© Copyright IBM Corporation 2014

Associative arrays

You can store and use data in an associative array in the rules file. Associative arrays are similar to arrays in Perl where the key (or index) field for the array is a string, for example, Node or Identifier. The arrays provide a systematic way of saving state between events even when processing dynamic data. Like writable properties, arrays maintain their values if the probe is hupped (kill -HUP) but are lost on probe restart.

The example that is shown on this slide illustrates the use of an associative array to calculate the interval between similar events. By keying the array with the Identifier field, you create an array element for each type of event. On the first occurrence of the event, the array element is initialized to 0.

You then calculate the interval by subtracting the time of the previous event from the time of the current event. This computation yields the interval value in seconds. After the interval field is populated, you store the current event time in preparation for the next event with this Identifier.

Additional functions

String functions

- Converts format of string functions

Time functions

- Converts UNIX time to time and vice versa
- Get current time

Utility functions

- Obtain platform-specific information such as host name

© Copyright IBM Corporation 2014

Additional functions

The rules file language is essentially a collection of functions that are used to perform various actions, and traditional programming constructs, like *if, then, else*. You saw some examples of functions in the previous slides. The probe rules language contains many more functions.

Refer to the *Probe and Gateway Guide* for a complete description of the language syntax, and available functions.

Events to alternate ObjectServers and tables

Must define all ObjectServers in .props or command

```
-server NY1:LON1:TOK1
```

To select an ObjectServer for all subsequent events

```
setDefaultObjectServer("NY1")
```

To select an ObjectServer for this event

```
setObjectServer("LON1")
```

To send events to alternate tables and ObjectServers

```
Usa = registerTarget("NY1","","  
                      "alerts.status","alerts.details")  
  
Asia = registerTarget("TOK1","LON1",  
                      "alerts.stat_asia","alerts.det_asia")  
  
setDefaultTarget("Usa")  
  
....  
  
if (match(@Location,"Tokyo")) {setTarget("Asia")}
```

© Copyright IBM Corporation 2014

Events to alternate ObjectServers and tables

You can send events to alternative ObjectServers and alternative tables. The probe `server` property must list all possible target ObjectServers. Connections are established when the probe starts up. If the probe fails to connect to any ObjectServer at start, it exits. After start, each connection is independent, going into store and forward and reconnecting if it is lost.

Send this event to an ObjectServer, subsequent events go to default per props file:

```
setObjectServer("NY1")
```

Send all subsequent events to an ObjectServer until the function is used again:

```
setDefaultObjectServer("NY1")
```

Send events to alternative tables and register the new destination table:

```
<targetname> = registerTarget (<server>,           <backupserver>, <alertstable>  
[,detailstable])
```

Example:

```
London = registerTarget("PRIMARY","SECONDARY",  
                      "alerts.london","alerts.detailslondon")
```

generic_clear trigger

generic_clear trigger requires configuration at the probe

- Uses these ObjectServer fields:
AlertGroup, AlertKey, manager, type
- Probe rules identify problem and resolution events by setting Type = 1 for **problems**, and Type = 2 for **resolutions**:

```
if (regmatch(@Summary, "interface.*down"))
{
    @AlertGroup="Interface"
    @Type = 1
}

else if (regmatch($Summary, "interface.*up"))
{
    @AlertGroup="Interface"
    @Type = 2
}
```

© Copyright IBM Corporation 2014

generic_clear trigger

With the **generic_clear** automation, a resolution event is matched with its problem event. The events must be correctly defined at the probe.

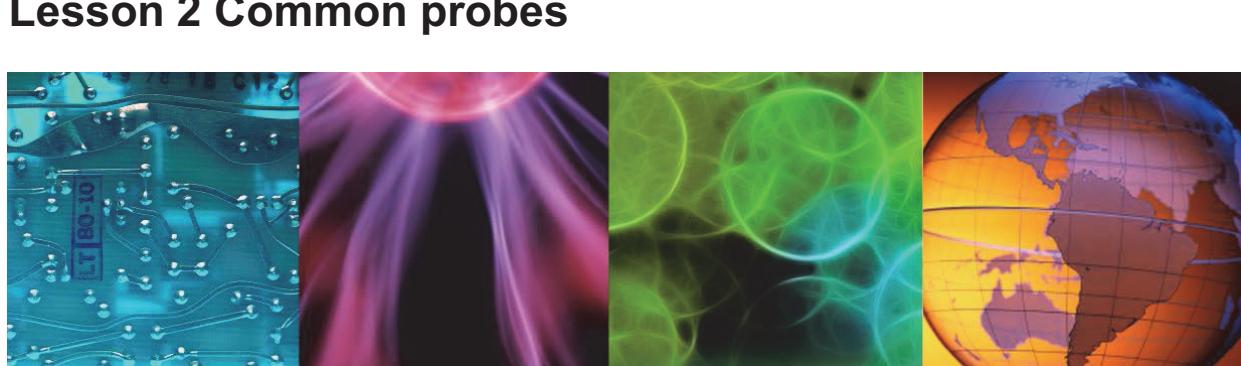
Problem type events are set to Type=1, and resolution events are set to Type=2. The AlertGroup and AlertKey fields correlate the resolution event to the problem event to ensure that the automation matches the correct events.

In this example, you see the rules that populate the required columns for the **generic_clear** automation. For resolution, and problem events, you must configure AlertGroup and Type fields. Many rules files that are provided have the **generic_clear** automation rules defined. The following rules are the **generic_clear** automation rules for the link up and link down events from the Simnet probe:

```
if (nmatch(@Summary, "Link Down")) {
    @AlertGroup="Link Down"
    @Type=1
}
else if (nmatch(@Summary, "Link Up")) {
    @AlertGroup="Link Down"
    @Type=2
}
```



Lesson 2 Common probes



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to install and configure the two most common probes: Syslog, and SNMP. After completing this lesson, you should be able to perform the following tasks:

- Install and configure the Syslog probe
- Install and configure the SNMP probe
- Install the Netcool Knowledge Library
- Describe the function of the MIB Manager

Syslog probe

Install the probe with IBM Installation Manager

The syslog probe can obtain its input from a log file or fifo

- Create either a file or a fifo
 - File: `touch /var/adm/xyz`
 - Fifo: `mkfifo /var/adm/xyz`

Add the following line to `/etc/syslog.conf`

```
*.debug /var/adm/xyz
```

Restart the syslog daemon

Run the probe

```
$OMNIHOME/probes/nco_p_syslog -logfile /var/adm/xyz  
$OMNIHOME/probes/nco_p_syslog -fifo /var/adm/xyz
```

© Copyright IBM Corporation 2014

Syslog probe

The technique that you use to acquire Syslog data involves modifying the behavior of the system Syslog daemon. The Syslog daemon is modified to send selected records to a separate file. The Syslog probe uses either a standard log file or a FIFO (First In First Out) file for its input.

Here is the process for the log file Syslog probe:

1. Edit the `/etc/syslog.conf`, and add the following line:

```
*.debug /var/adm/nco_logfile
```

2. Create the file `/var/adm/nco_logfile` with the `touch` command.

3. Send a SIGHUP signal to the `syslogd` process to reread its configuration file.

4. Start the Syslog probe with the `logfile` command-line option or property.

```
$OMNIHOME/probes/nco_p_syslog -logfile /var/adm/nco_logfile
```

Here is the process for the FIFO Syslog probe:

1. Edit the `/etc/syslog.conf` file, and add the following line:

```
*.debug /var/adm/info
```

2. Create the FIFO file `/var/adm/nco` by running the `mkfifo` command.

3. Send Kill -HUP to the `syslogd` process to re-read its config file.

4. Start the Syslog probe with the FIFO command-line option or property:

```
$OMNIHOME/probes/nco_p_syslog -fifo /var/adm/nco
```

Syslog next generation

SUSE Linux uses Syslog-NG

Configuration for use with the Syslog probe is slightly different

To configure Syslog-NG to send output to a new fifo file

- Create fifo
 - Fifo: `mkfifo /var/adm/xyz`
 - Change ownership: `chown netcool:ncoadmin /var/adm/xyz`
- Add the following lines to `/etc/syslog-NG/syslog-NG.conf`

```
destination netcool { pipe("/var/log/nco" owner(netcool) group(ncoadmin)); };
log { source(src); filter(f_messages); destination(netcool); };
```
- Restart the syslog-NG daemon

© Copyright IBM Corporation 2014

Syslog next generation

Some UNIX or Linux systems use an alternative form of the common Syslog daemon. This daemon is called Syslog Next Generation or Syslog-NG.

The information that is shown on the slide describes how the configuration changes are made for use with Syslog-NG. The configuration of the Syslog probe is the same regardless of whether you use **syslogd** or **syslog-NG**

SNMP probe

Install the probe with IBM Installation Manager

The most common method of integration

Supports SNMP V1, V2 and V3, configured in property settings

Supports UDP or TCP communication, configured in property settings

Probe must run as the root user if using default port 162

- UNIX operating system requirement (port < 1024)
- Possible to configure for non-root user

© Copyright IBM Corporation 2014

SNMP probe

The SNMP probe is the most common probe in use today. The probe can process SNMP v1, v2, or v3 traps. It also supports UDP or TCP communication.

One important point with regards to this probe relates to how Netcool/OMNibus runs in a UNIX or Linux environment. If Netcool/OMNibus is installed and runs as a nonroot user, the SNMP probe must run as the **root** user. This requirement is an operating system requirement. Any component that listens on a port below 1024 must run as **root**. One possible option to work around this requirement is to use an alternative port number for SNMP traps. The default is 162.

It is possible, using some modifications of the operating system, to configure the SNMP probe to use the default port (162), and run as a nonroot user. The steps to accomplish this configuration are detailed in the student exercise.



Hint: The SNMP probe was formerly known as the MTTrapd probe. Many references still use the old name.

Tivoli EIF probe

- Install the probe with IBM Installation Manager
- Event integration facility
 - Used for heritage Tivoli product integration
- Supports
 - IBM Tivoli Monitoring
 - IBM Tivoli Composite Application Manager (ITCAM)
 - Tivoli Enterprise Console
 - Tivoli Event Pump
- Requires ObjectServer modifications
 - New fields (8)
 - Modification to deduplication trigger
- This is a Java-based probe
 - nco_p_nonnaive runs
 - Calls Java and passes **nco_p_tivoli_eif.jar** file

© Copyright IBM Corporation 2014

Tivoli EIF probe

The Tivoli EIF probe provides integration for any Tivoli product that supports the Event Integration Facility (EIF). The probe includes rules that provide integration with several Tivoli products. Other Tivoli products provide their own customized rules that you can install in the probe to provide their integration.

The probe is installed with the IBM Installation Manager utility. However, there are some additional modifications to the ObjectServer that are required. The probe comes with an sql file. That file is read with the nco_sql utility, and all modifications are applied.

Netcool/OMNIbus Knowledge Library

Distributed with Tivoli Netcool/OMNIbus as separate download

Uses the **include** technique to incorporate multiple individual files

- One master rules file contains multiple **include** statements

Predefined rules files for multiple vendors and technologies

- Packaged as individual files, over 2000 in current version
- Easy to configure to add or remove

Rules provided for SNMP and Syslog probes

Installation

- Unpack rules files into target directory - \$NC_RULES_HOME
- Run supplied sql file to add ObjectServer customizations
- Configure probe to use NCKL master rules file
 - \$NC_RULES_HOME/syslog.rules - Syslog Probe
 - \$NC_RULES_HOME/snmptrap.rules - SNMP Probe

© Copyright IBM Corporation 2014

Netcool/OMNIbus Knowledge Library

The Netcool/OMNIbus Knowledge Library (NCKL) is a collection of rules files and lookup tables.

\$NC_RULES_HOME

The NC_HOME_RULES environment variable defines the location of the NCKL rules files (\$NCHOME/etc).

Expand the rules.

The rules are in a compressed file. Expand the file to \$NC_RULES_HOME to create a new subdirectory called rules.

ObjectServer

NCKL requires ObjectServer customizations, and for that reason it is included with an sql file. Use the nco_sql utility to read that file and apply the modifications.

Probe property

By default, the probe is configured to read the standard rules file. Modify the property file to use the NCKL master rules file.

User environment

Make sure that the user that runs the probe has the \$NC_RULES_HOME environment variable defined.

Start the probe.

Start probe as usual.

The Netcool Knowledge Library distribution contains a large collection of prebuilt integrations as delivered. However, event collection can be expanded. Most of the integrations available from the Integrated Service Management Library (ISML) site are packaged so that they can be incorporated into an existing Knowledge Library deployment.

MIB manager

Imports SNMP MIB files and converts to Netcool rules files for use with SNMP probe

Installs as part of Netcool/OMNIbus probe support option

Is a Java GUI application based on the eclipse platform

Ships with most common base and RFC SNMP MIB files for resolving imported MIB dependencies

Functions:

Automatic resolution of MIB dependencies

Allows grouping of MIB files into devices (vendor/model/OS)

Configure traps with preset values for ObjectServer fields such as @Severity and @ExpireTime or custom block of rules code

Generate test traps

Export to multiple format rules files including NCKL and lookup files

© Copyright IBM Corporation 2014

MIB manager

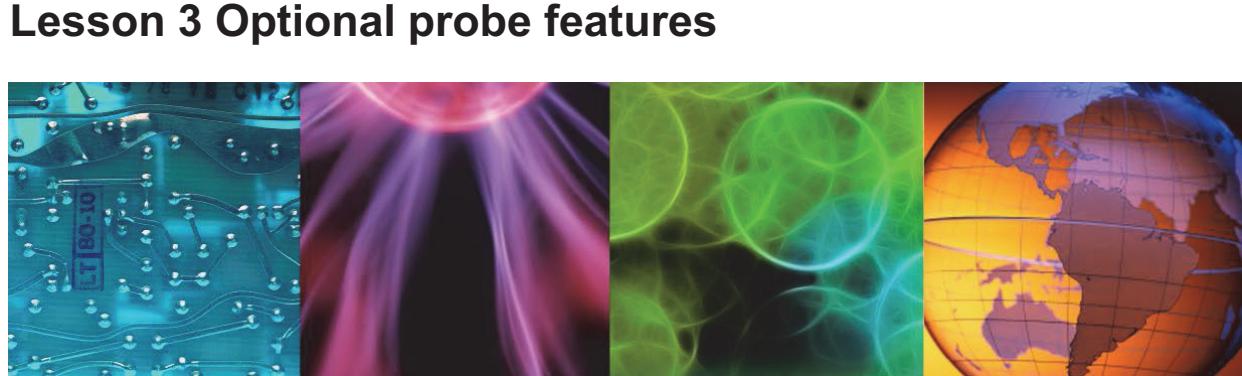
The Netcool/OMNIbus MIB Manager is an IBM Eclipse-based application that you can use to parse Simple Network Management Protocol (SNMP) Management Information Base (MIB) files, from which you can generate SNMP probe rules files. It is intended as a replacement for the `mib2rules` utility.

Netcool/OMNIbus MIB Manager has the following features:

- It imports SNMP MIB files and resolves MIB dependencies to build an Object Identifier (OID) tree.
- It can export some or all of the imported SNMP objects to Netcool rules files and other file formats.
- It includes the base MIB files and RFC MIB files commonly required by other MIB files.
- It includes device definitions to enable device-centric grouping of imported MIB modules.
- It can filter and search MIB modules and the OID tree by object name or OID.
- It can generate SNMP traps to send to the SNMP probe (`nco_p_mttrapd`) or other SNMP agents.



Lesson 3 Optional probe features



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to configure probes to implement two optional behaviors. These behaviors enable the probe to react to event flood conditions. After completing this lesson, you should be able to perform the following tasks:

- Describe the event flood behavior
- Configure a probe to implement this behavior

Optional probe features

Event flood detection

- Detects high event rate based on a preset threshold
- User can divert or discard lower priority events during flood

Anomalous event rate detection

- Detects an unusually low or unusually high rate of receipt of events
- Anomaly detection based on the normal rate of receipt of events
- Anomalous event rate detection reported in a new ObjectServer event

© Copyright IBM Corporation 2014

Optional probe features

Probes react to conditions. The SNMP probe receives an SNMP trap, translates the trap, and generates an ObjectServer event. In this mode of operation, the probe is entirely dependent upon its event source. Without a new source event, the probe has nothing to do. It does not generate an ObjectServer event.

Using custom rules files that come with Netcool/OMNibus, it is possible to configure a probe to implement two extra behaviors.

Event Flood Detection

You can configure the probe to compute event rate statistics and produce an ObjectServer event when an event rate exceeds a user-defined threshold. When the threshold is exceeded, the probe is considered to be in a *flood state*. It also produces a corresponding event when the event rate falls below this threshold. The user can configure how the probe behaves when a flood condition is detected. The user might want the probe to discard specific events and not flood the ObjectServer.

Anomalous Event Rate Detection

When you configure this behavior, the probe determines a *normal* rate of events that are based on a user-defined training period. The user configures conditions that indicate what they consider is less than normal event rate and what is a higher than normal event rate. When the probe detects either of these conditions, it generates corresponding ObjectServer events.

Installation

Two rules file fragments define event flood detection:

- `flood.config.rules`
- `flood.rules`

After installing Tivoli Netcool/OMNibus, rules file fragments are located in
`$OMNIHOME/extensions/eventflood`

© Copyright IBM Corporation 2014

Installation

No additional installation is required. All the components necessary to implement these new features are included with Netcool/OMNibus.

The feature is implemented as two extra rules files:

`flood.config.rules`
`flood.rules`

These rules files are installed during the Netcool/OMNibus core installation process. You can find them in `$OMNIHOME/extensions/eventflood`.

Configuration

Configuration is done in `flood.config.rules`

- Configure the time windows for collecting events for flood and anomalous event rate detection
- Configure the length of the training period during which the typical rate of receipt of events is determined
- Configure the threshold in events per second for determining when an event flood is in progress
- Configure what percentage of the typical event rate constitutes an unusually low or unusually high rate of receipt of events
- Configure remedial behavior to take during the event flood
 - For example, discard all events with severity less than major

Note: `flood.rules` should not be altered in most cases

© Copyright IBM Corporation 2014

Configuration

The user configuration is applied to only one of these files, **`flood.config.rules`**.

The time window for collecting events determines how frequently the probe computes an event rate. For example, if the window is set to 60 seconds, the probe counts events for 60 seconds. It then computes the event rate for that 60-second window.

The training period configures the amount of time that the probe must run to determine the *normal* event rate. The normal event rate is computed when the probe is initially started. The training period is used with the time window. For example, assume that the training period is set for 1 hour, and the time window is set to 600, which is 5 minutes. When the probe starts, it counts events for the first 5 minutes and then computes an event rate. This value is stored in an array. The probe continues to compute event rates for every subsequent 5-minute interval, and saves the values in the array, until 1 hour passes. At the end of this hour, the probe computes an average event rate that is based on the 5-minute values that are saved during the training window. This event rate is now considered the *normal* event rate for the probe. After the training window expires, the probe computes an event rate for every 5-minute interval and compares that rate to the normal rate based on two threshold settings.

The threshold setting that indicates an event flood condition is configured as a number that indicates a rate, $60 = 60 \text{ events/second}$. The low and high settings that are used with the anomalous rate detection are configured as percentages of the *normal* rate. For example, a low setting of 5 indicates that the probe generates a low condition event whenever the computed event rate falls below 5% of *normal*.



Note: These rules use the writable property and associative array features.

Usage

flood.config.rules

- Included at the beginning of the user's set of rules
- Necessary because the rules file fragment defines an array

flood.rules

- Included towards the end of the user's set of rules
- Necessary because the remedial action to take during the event flood might be based on the event severity
- Rules file needs to be processed first to determine this severity

The probe is run as normal

© Copyright IBM Corporation 2014

Usage

To implement this behavior in any probe, you edit the probe rules file and add two *include* statements. The location of those include statements in the probe rules file is important.

The statement to include **flood.config.rules** *must* be at the top of the probe rules file. The **flood.config.rules** file contains an *array* definition, and probe rules syntax requires that all arrays be defined first.

The statement to include the **flood.rules** file can go almost any place in the rules file. However, something to consider is whether the rules perform any optional remedial actions, such as discarding events. If this behavior is enabled, it might be based on specific conditions like the severity of an event. You might configure the probe to discard all low priority events during the flood condition. If that is the case, place the include statement in the rules file in a location where severity is already determined.

Limitations

You might need to adjust the configuration variables for the user's environment

- How long the probe runs before flood detection is enabled to cope with an initial burst of alarms when the probe is started
- How long the training period should be to determine the normal rate of receipt of events

After the training period is complete and the typical rate of receipt of events is determined

- This typical rate is valid only while the probe is running
- Does not persist between subsequent restarts of the probe

© Copyright IBM Corporation 2014

Limitations

As might be expected, all user environments are not the same. You cannot distribute a set of rules that fits all environments. Each environment requires a learning curve that is associated with this feature. This learning involves adjusting the various parameters to produce the wanted result in your environment.

Anything the probe learns while running is not retained. When the probe is started, it uses the various parameter settings to train itself and determine typical behavior. If you stop the probe, none of that information is saved. The next time that the probe starts, it retrains itself.

Example of Syslog probe usage

Edit syslog.rules

- Add to the beginning of the file
 include "flood.config.rules"
- Add to the end of the file
 include "flood.rules"

Edit flood.config.rules

- Adjust the configuration variables to suitable values:
 \$average_event_rate_time_window = 60
 This is the anomaly detection training period in seconds
 \$flood_detection_event_rate_flood_threshold = 50

Run the Syslog probe

© Copyright IBM Corporation 2014

Example of usage of Syslog probe

You implement the flood detection behavior in the Syslog probe as follows:

1. Edit the syslog.rules file and add the include statements for the two files.
2. Edit the flood.config.rules file to set the various conditions and thresholds.
3. Start the probe as usual.

Example of usage, continued

Flood condition

- Probe receives more than 50 events per second
- ObjectServer event indicating that a flood is detected

Flood subsides

- ObjectServer event indicating that an event flood is finished
- Describes number of events received during flood condition and the duration

Low event condition

- Probe receives less than 10% of the expected rate of events
- ObjectServer event indicating low event rate

High event condition

- Probe receives more than five times the expected rate of events
- ObjectServer event indicating high event rate

© Copyright IBM Corporation 2014

Example of usage, continued

Based on the parameter settings that are shown previously, the following behaviors are expected:

- Flood condition

The probe is configured to count events for 60 seconds. At the end of that interval, if the computed event rate exceeds 50 events per second, the probe generates an ObjectServer event that indicates a flood condition. The probe continues to count events for subsequent 60-second windows.

- Flood subsides

During a subsequent 60-second window, if the computed event rate falls below 50 events per second, the probe generates a new ObjectServer event. This event indicates that the flood condition is ended. In that event, the probe includes the number of events that are received during the flood and how long it lasted, based on 60-second windows.

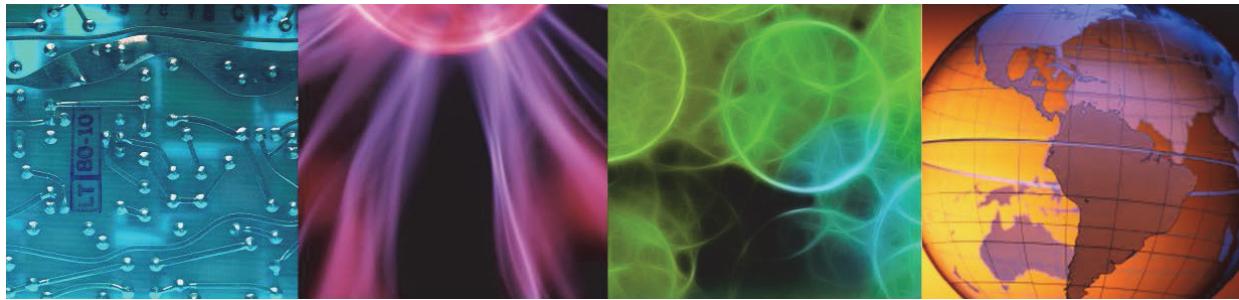
- Low event condition

During those same 60-second windows, if the probe determines that the computed event rate is less than 10% of normal, it generates a new event. This event indicates a low condition.

Lesson 4 Remote probe administration



Lesson 4 Remote probe administration



© Copyright IBM Corporation 2014

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

In this lesson, you learn about the features available to facilitate the administration of probes that are running on remote servers. After completing this lesson, you should be able to perform the following tasks:

- Describe rules file caching
- Describe the probe registry
- Describe how HTTP/HTTPS access is configured for probes

Features to facilitate remote administration

Distributed probe with local rules file

- Probe configured to retrieve rules at startup using HTTP

Local rules file caching

- Used when remote probe retrieves rules file using HTTP
- Probe caches a local copy of rules file

Bi-directional communication

- Communicate with probe using HTTP
- Change property values dynamically
- Instruct probe to reread rules file

Probe registry

- Every running probe registers with ObjectServer
- ObjectServer table contains location of all probes

Process activity

© Copyright IBM Corporation 2014

Features to facilitate remote administration

In most environments, probes are distributed across multiple servers. In some cases, those servers are located remotely, and the administrator has no physical access to the hardware. In this case, all administration must be performed through network access. The administrator might face a significant challenge that is based on the number of probes, and their geographic distribution.

Netcool/OMNIbus provides a number of features that facilitate the remote administration of probes.

This slide contains a list of some of these features. More details are provided on subsequent slides.

Rules file caching

Probes can store their last working rules file in a cache

Stored in a persistent file, in **\$OMNIHOME/var** by default

Provides a backup rules file if the probe restarts and the existing rules file contains syntax errors or is not accessible

If rules caching is enabled, then the following occurs:

- Cache file written out whenever a new valid rules file is read in by the probe, could be at startup or reread by SIGHUP/HTTP
- Upon startup probe attempts to read its normal rules file
- If reading the normal rules file fails because of syntax error or missing or unavailable file, the probe tries the cache file, if present
- If reading the cache file fails, the probe exits as normal

© Copyright IBM Corporation 2014

Rules file caching

A core component of the probe is the rules file. This text file requires modifications whenever the probe is changed to incorporate other functions. When a rules file for a probe that is running on a remote server is modified, there are several challenges:

1. Changing the file on a remote server. In some cases, it might be possible to connect to the remote server and modify the file directly on the server. If the administrator cannot access the remote server, a copy of the file must be retrieved, modified, and then replaced.
2. Causing the probe to reread the modified rules file. After the rules file is modified, the probe must be instructed to reread the file. Again, it might be possible to connect to the server remotely and enter the appropriate command.

One option for the challenge of managing rules files on remote servers is to place all rules files on a central server. Then configure each probe to retrieve a copy of their respective rules file when the probe starts. At start, the probe retrieves a copy of the file, stores the file in cache memory, and runs as normal. There are two options for how the probe retrieves the file, FTP or HTTP. However, both options necessitate network access between the probe, and the rules file server. If the probe cannot access the remote rules server because of network issues, it cannot retrieve a copy of the rules file, and it cannot start.

A recent change to Netcool/OMNibus probes is to provide the option to save a copy of the rules file *locally* when the probe starts. If the probe can access the remote rules file server at least one time, it retrieves the file, stores a copy in a local cache, stores a copy in cache memory, and runs as usual. If the probe restarts, for some reason, and it cannot access the remote rules file server, it uses the *local* copy of the rules file. This feature makes it possible to administer all probe rules files from a central location.

Bidirectional communication

Provides an HTTP(S) port for communication to probes

Common set of functions that all probes inherit

Provide a base for probes to be extended in the future with individual probe specific function

Not a full REST interface, but quite REST-like

Supported HTTP verbs of the /probe/common URL are:

- GET
- POST
- PATCH

© Copyright IBM Corporation 2014

Bidirectional communication

Any time you change a rules file or a property file, you must instruct the probe to re-read the changed files. If you can connect to the remote server, you can enter the appropriate command locally.

A new feature for Netcool/OMNIbus probes is the option to configure the probe to support remote access. The remote access is provided with HTTP, or HTTPS. Some utilities use the HTTP/HTTPS access to pass commands to the probe. Using these utilities, an administrator can send a command to a remote probe, and command the probe to perform these tasks:

- Reread a rules file.
- Show all property values.
- Change the value of one or more properties.

If a probe is configured for HTTP access and to retrieve its rules file remotely, administration becomes much simpler. The administrator modifies a local copy of the rules file and sends a command to a remote probe to reread its rules file. The probe retrieves a copy of the modified rules, stores a copy in a local cache, and replaces the contents of the in-memory cache with the contents of the modified file. The probe does not stop, and the administrator does not require access to the remote probe server.

Probe registry

ObjectServer database and table: registry.probes												
Data View												
RowID	RowSerial	Name	Hostname	PID	Status	HTTP_port	HTTPS_port	StartTime	ProbeType	ConnectionID	LastUpdate	
20	20	heart	devtest42.hursley.ibm.com	6981	1	0	0	1349367009	heartbeat	1	1349774684	
24	24	simnet	devtest43.hursley.ibm.com	14222	1	0	0	1349367010	simnet	5	1349774684	
23	23	tivolt_ef	devtest44.hursley.ibm.com	15232	1	0	0	1349436051	tivolt_ef	4	1349774684	
21	21	sim1	devtest43.hursley.ibm.com	14252	1	0	4198	1349367016	simnet	2	1349774684	
22	22	sim2	devtest43.hursley.ibm.com	14295	1	4199	0	1349367013	simnet	3	1349774684	
25	25	heartbeat	devtest45.hursley.ibm.com	14384	1	0	0	1349367210	heartbeat	6	1349774684	

ObjectServer database and table:

[registry.probes](#)

Probes automatically write information about themselves when connecting to a V7.4 ObjectServer

Requires no configuration

Function:

Global actions on all probes using the V7.4 Bi-Di probe communication feature

For example: *Flood Control* sample feature

© Copyright IBM Corporation 2014

Probe registry

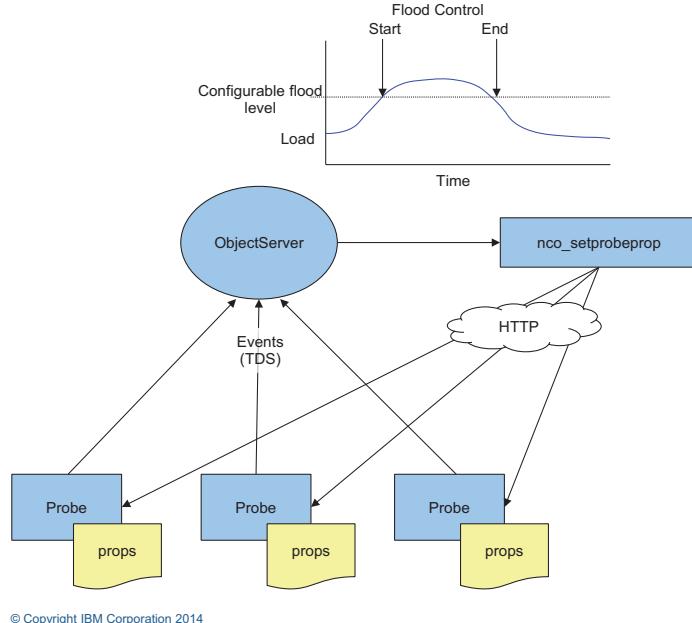
Another new feature is the probe registry. This feature incorporates an ObjectServer database (registry), and table (probes). All probes that connect to the ObjectServer create an entry in the registry.probes table. The entry contains the name of the probe, the server that hosts the probe, and the probe process id. Any probe that is configured to provide remote access through HTTP or HTTPS also includes the port numbers that are used for that access.

This table provides a convenient reference for any administrator that needs to access a remote probe. The contents identify the physical location of the probe, and whether the probe is configured for HTTP or HTTPS access. In addition to being a reference for administrators, this table also provides the option for automating access to one or more remote probes. The following slide illustrates an example of this feature.

Bidirectional probe communication usage

An example of bidirectional probe communication is the ObjectServer centralized flood control feature.

- Flood control trigger detects when ObjectServer load reaches a defined level.
- The trigger then iterates over all active probes and calls nco_setprobeprp program to set a FloodControl property through the probe HTTP interface.
- The FloodControl property causes probes to discard low severity events.
- After the ObjectServer load drops to an acceptable level, the trigger resets the FloodControl property of the probes.



© Copyright IBM Corporation 2014

Bidirectional probe communication usage

In a previous slide, you learned about the optional probe behavior flood detection. When configured with this behavior, the probe performs these tasks:

1. It counts events for some time interval.
2. It computes an event rate.
3. It compares the event rate to a user-configured threshold.
4. If the threshold is exceeded, the probe can discard certain events.

All that behavior is implemented entirely within the probe. In the example that is shown here, too many events came into the ObjectServer too quickly. The assumption is that the events are coming from multiple probes. The administrator wants to protect the ObjectServer from too many events. One solution is to use some of the features that were described previously to implement an automated response to an event flood. The steps in this solution are as follows:

1. An ObjectServer trigger periodically computes an event rate.
2. The trigger compares the event rate to a threshold.
3. If the threshold is exceeded, the trigger runs a procedure.
4. The procedure reads the contents of the registry.probes table.
5. Based on the contents of this table, the procedure sends a command with HTTP or HTTPS to every remote probe.

6. The command instructs the probe to set the FloodControl property, indicating that the *probe* is in a flood condition.
7. The probe discards specific events, which are based on the behavior that you define in the *flood.rules* file.
8. The ObjectServer trigger continues to track the event rate.
9. When the ObjectServer trigger determines that the event rate is below the defined threshold, it runs a procedure.
10. This procedure sends commands to all remote probes to disable the FloodControl property.
11. Remote probes now send all events as usual.

Student exercises



© Copyright IBM Corporation 2014

Student exercises

Perform the exercises for this unit.

Summary

You now should be able to perform the following tasks:

- Implement common integrations in Tivoli Netcool/OMNibus
- Use a probe at the basic level
- Configure and modify probes
- Install and configure the Netcool Knowledge Library (NCKL)
- Install the UNIX Syslog probe and configure it to use the NCKL integration
- Explore the event rate detection capabilities
- Facilitate remote probe administration with available features

© Copyright IBM Corporation 2014

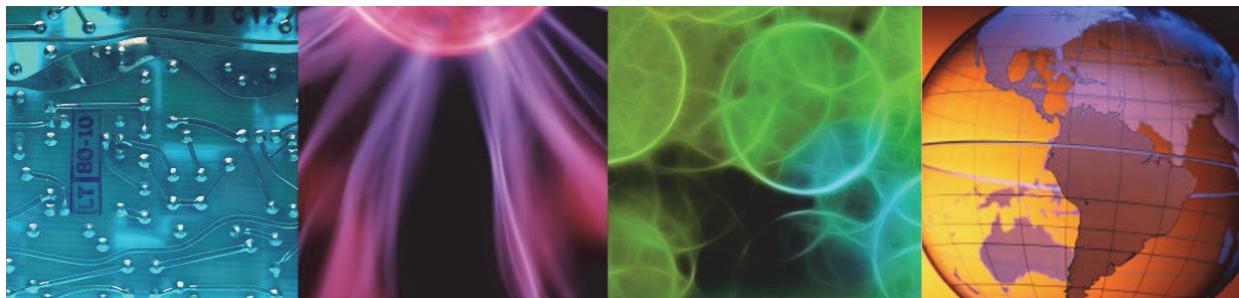
Summary



8 ObjectServer high availability



8 ObjectServer high availability



© Copyright IBM Corporation 2014

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

In this unit, you learn about the high-availability configuration. You learn about the functional components, and the process to create the configuration.

References: SC27-6264-00 *Installation and Deployment Guide*
SC14-7531-00 *Netcool/OMNIbus ObjectServer Gateway Reference Guide*

Objectives

In this unit, you learn to perform the following tasks:

- Describe the functional components of the OMNIbus high availability (HA) configuration
- Use the nco_confpak utility to clone an ObjectServer configuration
- Configure the bidirectional gateway to synchronize the event data content of two ObjectServers
- Configure the automated failover and fallback feature that manages automations on the backup ObjectServer
- Describe options for increased data integrity



Lesson 1 Overview

Lesson 1 Overview



© Copyright IBM Corporation 2014

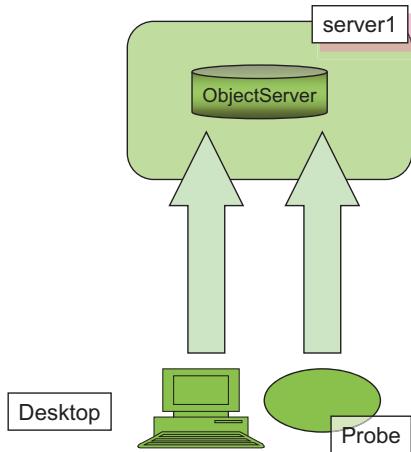
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this lesson, you learn about the architecture, and functional components of the high availability configuration. After completing this lesson, you should be able to describe the functional components of a high availability configuration.

Architecture overview

Without failover

- Probes, desktops, and gateways connect to an ObjectServer
- With a single ObjectServer there is no backup
- If the ObjectServer fails, probes and gateways go into store-and-forward, desktops cannot function



© Copyright IBM Corporation 2014

4

Architecture overview

In a Netcool®/OMNIbus deployment without high availability, an ObjectServer failure is catastrophic. Probes can continue to collect data from their respective sources and maintain the events in local store-and-forward files. They cannot store the data indefinitely because available disk space is the limiting factor. Desktop users have no access to event data.

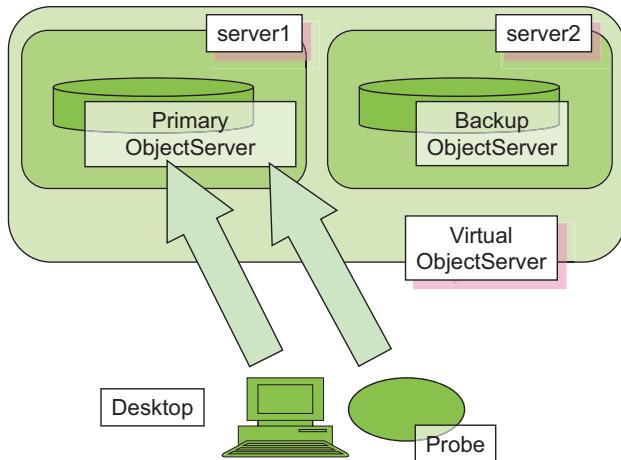
Virtual ObjectServer

Two ObjectServers are configured as a ***virtual*** ObjectServer

Desktops and probes are configured to connect to the ***virtual*** ObjectServer

In actuality, clients are connecting to the real primary ObjectServer

Bidirectional gateway replicates changes from the primary ObjectServer to the backup ObjectServer



© Copyright IBM Corporation 2014

5

Virtual ObjectServer

The Netcool/OMNibus High Availability configuration is based on two ObjectServers that are both running. One of them is designated the primary. All components, desktops, probes, other products, are connected to the primary. New events from probes are inserted into the primary ObjectServer. Desktop users can view and manipulate those events.

The second ObjectServer is designated as the backup. The backup ObjectServer is also active, but no clients are connected.

The HA configuration relies upon a ***virtual*** ObjectServer. As the name implies, the ***virtual*** ObjectServer is not a real component. It is a communication mechanism that is used to identify the two real ObjectServers.

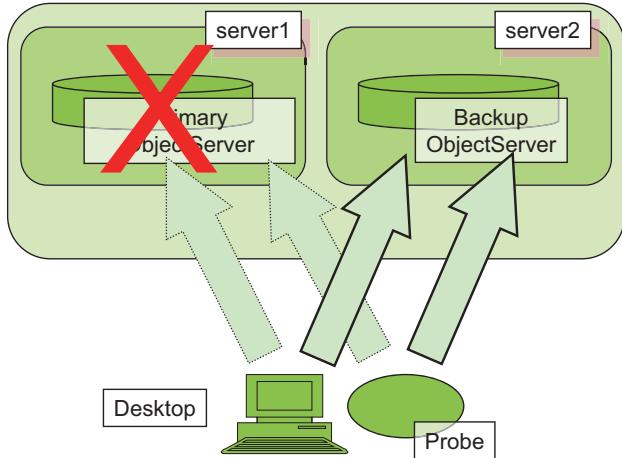
An ObjectServer has a definition in the IDUC interfaces file. Client components use the definition to determine how to communicate with an ObjectServer. The entry contains the name of the server that is hosting the ObjectServer and the port number that the ObjectServer uses. The ***virtual*** ObjectServer is a definition in the interfaces file that has a single ObjectServer name, but two sets of host names and port numbers. One set of data points to the primary ObjectServer, and the other set of data points to the backup ObjectServer.

Another key component in the HA configuration is the bidirectional gateway. The role of the gateway is to replicate all changes from the primary ObjectServer to the backup ObjectServer. This replication ensures that the contents of the backup are the same as the primary.

Failover process

If the primary ObjectServer fails, the clients reconnect to the backup ObjectServer

- Probes switch automatically
- Desktops generate a pop-up window, and user accepts reconnect



© Copyright IBM Corporation 2014

6

Failover process

If the primary ObjectServer fails, or is taken down, the client components detect that they cannot communicate with that ObjectServer. When that happens, the clients refer to the interfaces file to determine whether they have another choice. If the clients are configured to use the *virtual* ObjectServer definition, another choice is the host and port number for the backup ObjectServer. The client components then attempt to connect to the backup. If the connection is successful, the components resume their normal mode of operation with the backup ObjectServer.

Probes detect a primary failure and switch transparently to the backup ObjectServer with no intervention required.

Native Desktop users receive a pop-up message, indicating that their ObjectServer connection failed. The pop-up message has a button to reconnect. When the user selects that button, the desktop connects to the backup ObjectServer and resumes normal operation.

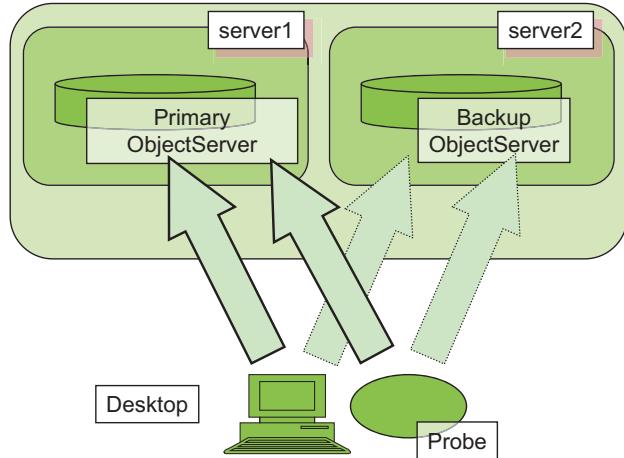
While connected to the primary, there is a box in the lower, right corner of the desktop that contains the text PRI, which indicates that the user is connected to the primary ObjectServer. When the desktop fails over to the backup, that box contains BAK.

Before the failure, the bidirectional gateway is replicating all changes from the primary to the backup. When the client components fail over, the backup looks exactly like the primary.

Fallback process

When the primary ObjectServer recovers, these actions occur:

- The bidirectional gateway synchronizes the primary from the backup
- The clients optionally reconnect to the primary ObjectServer
- Automatic failback is an option, requires additional configuration to enable



© Copyright IBM Corporation 2014

7

Fallback process

When the primary ObjectServer recovers, it no longer contains the same event data as the backup. The primary might be down for a minute or for an hour. While it is down, all event activity is directed to the backup. The first step in the failure recovery is for the bidirectional gateway to synchronize the contents of the primary with the contents of the backup.

The next step in the recovery involves the client components, which are currently connected to the backup ObjectServer. Some users want components to fail back automatically when the ObjectServer recovers. Other users want to control exactly when the failback occurs. Both solutions are possible.

If the system is configured to enable automatic failback, the client components detect that the primary ObjectServer is recovered. Probes fail back to the primary transparently with no intervention required. Desktops disconnect from the backup, and show the same pop-up message that requests permission to reconnect. When the button is selected, they reconnect to the primary.

Configuring interfaces file for failover

Configure the interfaces values for each ObjectServer

1. Enter the primary server name, host name, and port
PRIMARY <yourhost> <port>
2. Click **Add**
3. Enter the backup server name, host name, and port
BACKUP <yourhost> <port>
4. Click **Add**

Configure the virtual ObjectServer pair

1. Enter new name for the virtual ObjectServer plus primary host and port
VIRTUAL <yourhost> <primary_port_num>
2. Click **Add**
3. Select **VIRTUAL**
4. Enter the backup host and port
<yourhost> <backup_port_num>
5. Click **Add** to create a backup entry

Configuring interfaces file for failover

The process to define a *virtual* ObjectServer in the interfaces file is nearly identical to the process you use for a real ObjectServer. You use the nco_xigen utility.

The interfaces file should have an entry for both real ObjectServers. One entry defines the primary, and a second entry defines the backup. Any server that is hosting an ObjectServer must have an entry for the real ObjectServer. The real ObjectServer needs to reference that entry to determine which port number to use. On any server, user workstations or probe servers, you configure only an entry for the *virtual* ObjectServer.

To configure the *virtual* ObjectServer entry, you start with an entry that looks almost like the primary. The ObjectServer name is different, but the host and port are the values that you use for the primary. You click **Add** to add the entry. With that same line selected, you overwrite the host name and port field with the values for the backup ObjectServer. You click **Add** again because you are adding more data to that same definition.



Important: The Initial Configuration Wizard creates the entry for the virtual ObjectServer automatically. When the wizard is used, there is no need to perform this task manually.

How nco_xigen shows failover

Server	Hostname	Port	SSL
AGG	omnihost	4100	
└ Backup1:	omnihost	4101	
AGG_B	omnihost	4101	
AGG_P	omnihost	4100	
NCO_GATE	omnihost	4300	
NCO_PA	omnihost	4200	
NCO_PROXY	omnihost	4400	

© Copyright IBM Corporation 2014

9

How nco_xigen shows failover

This screen capture illustrates the interfaces entries that are required for the high availability pair of aggregation ObjectServers used in the best practices multitier configuration.

AGG_P is the real primary.

AGG_B is the real backup.

AGG is the *virtual*.

Client components connect to AGG. The system determines whether the connection goes to AGG_P or AGG_B.



Important: In this example, both ObjectServers are running on the same host: omnihost. Therefore, they are defined with different port numbers: 4100, and 4101. In a production environment, the ObjectServers run on different hosts. In that configuration, they can both use the same port number.

Configuring client processes for failover

For desktops, select the ***virtual*** server from the ObjectServer list

Must configure probes in the <probename>.props file:

```
Server : 'VIRTUAL'
```

Gateway configurations vary based upon the type of gateway

- Most gateways connect to the ***virtual*** ObjectServer
- Bidirectional gateway uses the real ObjectServer names

Configuring client processes for failover

For the native desktop, the users see a list of ObjectServers when the desktop is started. They pick the ObjectServer from the list. The list is generated from the definitions in the local copy of the interfaces file. For desktop workstations, the interfaces file should contain only an entry for the ***virtual*** ObjectServer. This way the user has only one ObjectServer name to choose.

The ObjectServer used by a probe is configured in the probe property file. Configure the ***virtual*** ObjectServer name.

The configuration for gateways varies by the type of gateway. For most gateways, for example, trouble ticket and database gateways, configure the gateway to use the ***virtual*** ObjectServer name. When configured with the ***virtual*** ObjectServer name, the gateway fails back and forth between the primary and backup ObjectServers. The bidirectional ObjectServer gateway is an exception. This gateway connects two ObjectServers together. When used to connect the primary and backup ObjectServers in a high availability configuration, you want to use the ***real*** ObjectServer names. An example is AGG_P and AGG_B.

Configuring the Web GUI component: Option 1

Manually edit XML configuration file

Restart Dashboard Application Services Hub

```
/opt/IBM/netcool/omnibus_webgui/etc/datasources/ncwDataSourceDefinitions.xml
```

```
<ncwPrimaryServer>
    <ncwOSConnection maxPoolSize="10" port="4100"
host="host1.tivoli.edu" ssl="false" minPoolSize="5"/>
</ncwPrimaryServer>

<!-- The optional failover ObjectServer to connect to -->
<ncwBackUpServer>
    <ncwOSConnection maxPoolSize="10" port="4100"
host="host2.tivoli.edu" ssl="false" minPoolSize="5"/>
</ncwBackUpServer>
</ncwFailOverPairDefinition>
```

1-11

Configuring the Web GUI component: Option 1

The Web GUI component does not use the interfaces file to determine connectivity to ObjectServers. The ObjectServer connectivity information is defined in an XML file. There are several options available for defining ObjectServers to Web GUI.

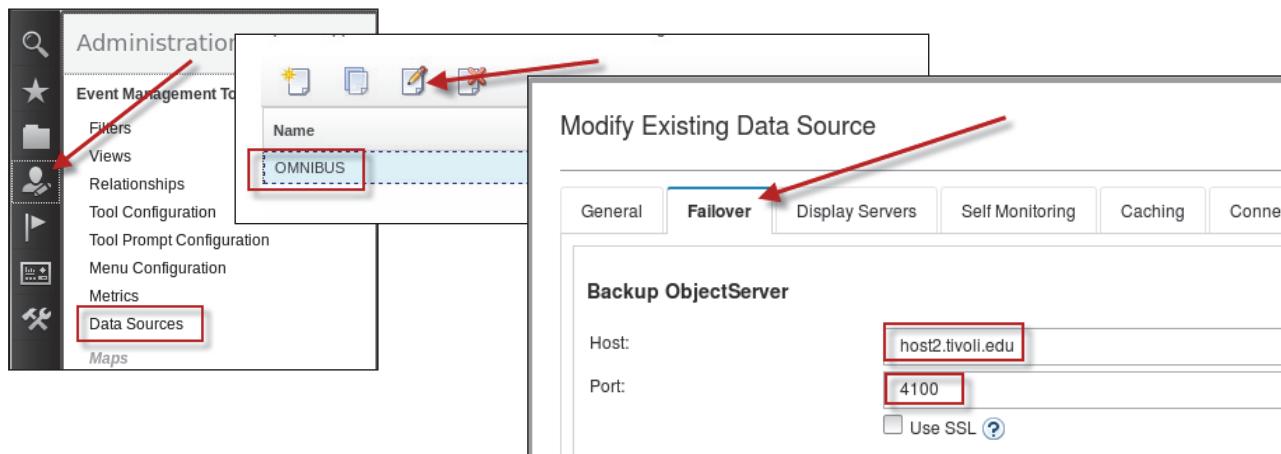
One option is to manually edit the XML file. The screen capture shows a portion of the XML file that contains the commands to define the access information for the primary and backup ObjectServers.

When the file is modified manually, you must restart Dashboard Application Services Hub to activate the changes in the Web GUI component.

Configuring the Web GUI component: Option 2

Define the backup ObjectServer using Dashboard Application Services Hub

- Change takes place immediately with no restart required
- XML file is automatically updated



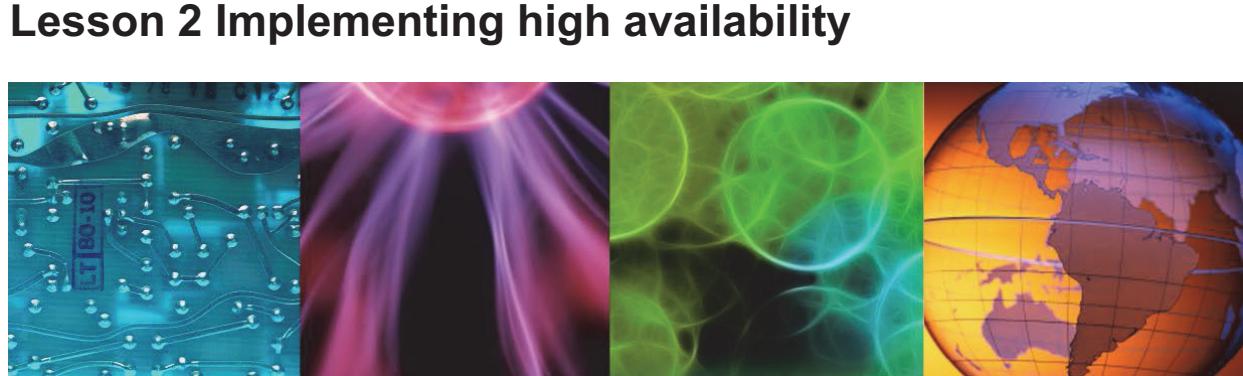
1-12

Configuring the Web GUI component: Option 2

The second option for defining ObjectServer connectivity is available through the Dashboard Applications Services Hub user interface. Any user with the *ncw_admin* role has access to this feature. When this feature is used, the changes take effect immediately and do not require a restart. In addition, the XML file is modified automatically.



Lesson 2 Implementing high availability



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to implement a high availability configuration. After completing this lesson, you should be able to perform the following tasks:

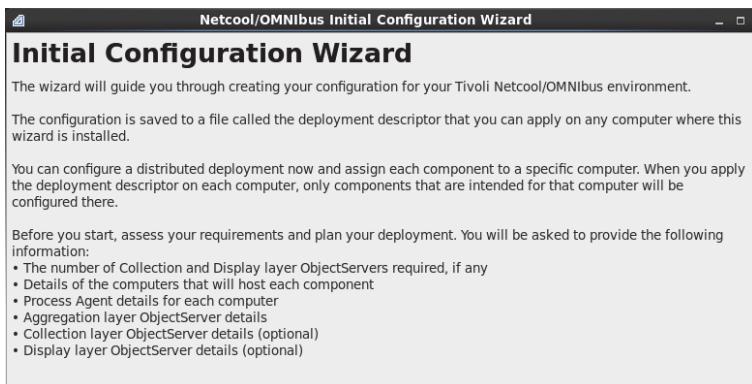
- Describe the steps that are required to implement high availability.
- Replicate an ObjectServer configuration.
- Describe important property settings and their role in the high availability configuration.

Initial Configuration Wizard

Creates Netcool/OMNibus environment

- Primary ObjectServer
- Backup ObjectServer
- Interfaces file
- Gateway configuration
- Process Activity configuration

The wizard creates and defines the backup components automatically



© Copyright IBM Corporation 2014

Initial Configuration Wizard

You can use the Initial Configuration Wizard to create the complete high availability configuration with minimal manual intervention. You can use it to create a single ObjectServer deployment initially. You can then use it again to add the components that are required for high availability.

The student exercises demonstrate how to use the wizard to add high availability to an existing deployment.

Steps for implementing high availability

Update interfaces file **

- Add entry for backup ObjectServer
- Add entry for *virtual* ObjectServer

Create backup ObjectServer

- Create new ObjectServer **
- Replicate primary configuration
- Start the backup ObjectServer

Configure bidirectional gateway **

- Adjust property file
- Adjust table replication files

Configure components to use *virtual* ObjectServer

*** These steps can be performed by the Initial Configuration Wizard*

© Copyright IBM Corporation 2014

15

Steps for implementing high availability

High availability is typically configured after a functioning stand-alone deployment is in place. In this case, the existing ObjectServer typically becomes the primary.

The first step in building the HA configuration involves creating the backup ObjectServer. The initial steps for creating the backup ObjectServer are the same as any other ObjectServer:

1. Add an entry to the interfaces file.
2. Run nco_dbinit to create the basic structure.



Note: The Initial Configuration Wizard can perform these steps.

One important aspect of the backup ObjectServer is that it should look exactly like the existing primary. That means that the same users, menus, tools, automations are available to it. If no customizations or no modifications are made to the primary, then the backup should be the same when you initially create it. Typically, that is not the case. Therefore, the next step is to make the backup look like the primary.

If you make only a few changes to the primary ObjectServer, one option is to perform the same changes to the backup with the Administrator utility. However, if there are considerable changes, then that is not a viable option. The *configuration pack* that is described in this unit offers a solution.



Note: The Initial Configuration Wizard cannot replicate the configuration of the primary ObjectServer into the backup ObjectServer. The replication must be done manually.

The next step is to configure the bidirectional gateway and start it. Then configure client components to use the virtual ObjectServer.



Note: The Initial Configuration Wizard can create the gateway configuration files. The wizard does not modify the client components.

Configuration pack, nco_confpack

Transfers configurations between ObjectServers

Used with ObjectServers online

1. List the existing configuration to a file

```
./nco_confpack -list -server NCOMS -file /tmp/CONF_LIST
```

2. Edit the list file to transfer information you want

3. Export the configuration list to a Java package

```
./nco_confpack -export -file /tmp/CONF_LIST -package /tmp/A_CONF.jar
```

4. Import the package on the destination ObjectServer

```
./nco_confpack -import -server BACKUP -package /tmp/A_CONF.jar
```

© Copyright IBM Corporation 2014

16

Configuration pack, nco_confpack

The **nco_confpack** utility provides a mechanism for exporting the configuration of an ObjectServer. You have the option of selecting which components to export or to export the entire configuration. The utility exports everything from the ObjectServer except the event-related data: alerts.status, alerts.details, and alerts.journal. The utility can export the table definition for the alerts.status table, but it does not export the actual event data. The utility exports ObjectServer configurations, not the event data.

The output of the utility is a JAR file. The file is small enough that you can attach it to an email message.

After the configuration is exported, use it to replicate the configuration, or parts of it, in another ObjectServer. This method is convenient for creating new ObjectServers that look exactly like another one. Use this method to replicate the configuration from the primary into the backup. You can use this method to replicate a production ObjectServer into a test configuration. Because it is possible to selectively export parts of a configuration, you can use it to move configuration changes from a test system into production.

In the following steps, you see how to use nco_confpack to export selected configuration components.

1. List the configuration available in an ObjectServer into a file.
2. Select the information that you want to package by editing this file.
3. Export the package with this file.
4. Import the package into another ObjectServer.

To export and import the entire configuration, use the following steps:

1. Export the configuration with this file:

```
nco_confpack -export -server PRIMARY -user root  
-password object00 -package /tmp/PRIMARY.jar
```

2. Import the package into another ObjectServer.

```
nco_confpack -import -server BACKUP -user root  
-password object00 -package /tmp/PRIMARY.jar
```

This process replicates the entire configuration of the *primary* ObjectServer into the *backup* ObjectServer with two notable exceptions.

All users are replicated from the primary to the backup except the *root* user, which is by design. You do not want to replicate an ObjectServer configuration and discover that you can no longer log in to the replicated server because the *root* user password is changed.

The other components that are not replicated are property values, which are also by design. Some property values define the log file name that contains the ObjectServer name. If these property values are replicated, the updated ObjectServer creates a log file with the other ObjectServer name.

Before you import the configuration from another ObjectServer, create an export of the destination. This export ensures that you can roll back to a working configuration if errors occur.



Important: The utility requires that the ObjectServers are running in to perform an export or import operation.

Important BackupObjectServer property

BackupObjectServer TRUE

Provides for desktops

- Causes the **BAK** text to appear, indicating to the users that they are connected to the backup. Without this property, the desktop always shows PRI
- Enables automatic failback when the primary comes back. Without this property, the desktop stays connected to the backup

Provides for probes

- Indicates to the probe when it is connected to the backup. Without this property, the probe never attempts a failback
- Use in conjunction with the probe properties NetworkTimeout and PollServer. Without this property, the probe never attempts a failback, even with the probe properties configured

** The Initial Configuration Wizard creates the backup ObjectServer with this property set to TRUE

© Copyright IBM Corporation 2014

17

Important BackupObjectServer property

After you create the backup ObjectServer with the nco_dbinit utility, and the configuration from the primary is replicated, you must configure one ObjectServer property on the backup ObjectServer.

BackupObjectServer TRUE

The default for this property is FALSE, and must be manually set to TRUE. This property enables a number of key behaviors.

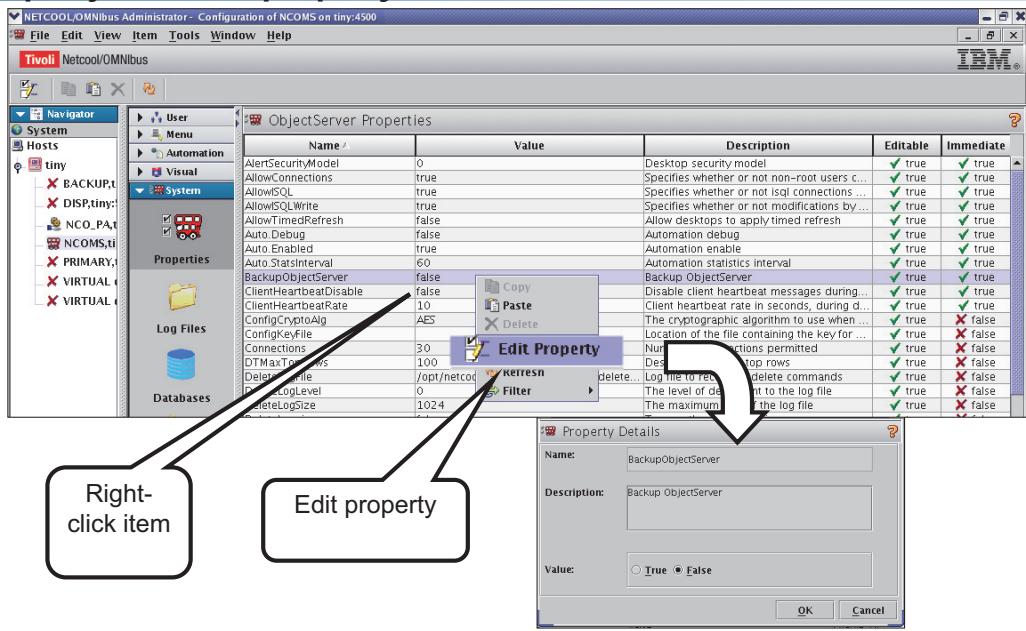
When a native desktop connects to an ObjectServer, it retrieves this property. The value of this property is an indication to the desktop whether it is connected to a primary or backup ObjectServer. This property causes the text to show in the corner of the desktop that indicates PRI or BAK. It also causes a desktop to failback when the primary ObjectServer recovers. If this property is not set, then the desktop is not aware that it is connected to the backup.

When a probe connects to an ObjectServer, it also retrieves this property. The value is an indication to the probe whether it is connected to a primary or backup ObjectServer. The probe uses this value with two other probe property values to enable automatic failback when the primary ObjectServer recovers. When the probe is aware that it is connected to the backup ObjectServer, it uses the values from two property settings to periodically test the availability of the primary. If the BackupObjectServer property is not set to TRUE, the probe is not aware that it is connected to a backup ObjectServer, and never attempts to fail back.



Note: The Initial Configuration Wizard creates the backup ObjectServer with this property already set to the correct value. There is no need to perform this task manually.

BackupObjectServer property



© Copyright IBM Corporation 2014

18

BackupObjectServer property

This screen capture illustrates how to configure the BackupObjectServer property manually.

Configuring properties for failback

In the backup ObjectServer

Use **nco_config** utility to set ObjectServer property

```
BackupObjectServer : TRUE
```

On the probe

Edit the \$OMNIHOME/probes/<arch>/<probe name>.props file

```
NetworkTimeout      : integer
PollServer          : integer
```

19

Configuring properties for failback

Two property settings are required to configure a probe to automatically failback when the primary ObjectServer recovers. These properties are configured in the probe property file:

- **PollServer**, default is 0.

This property defines the frequency, in seconds, at which the probe attempts a connection to the primary ObjectServer.

- **NetworkTimeout**, default is 0.

This property defines how long, in seconds, the probe waits for the primary ObjectServer to respond to the connection request.

Remember, these property values are in addition to the BackupObjectServer property. All of these properties must be set correctly to configure a probe to automatically failback.



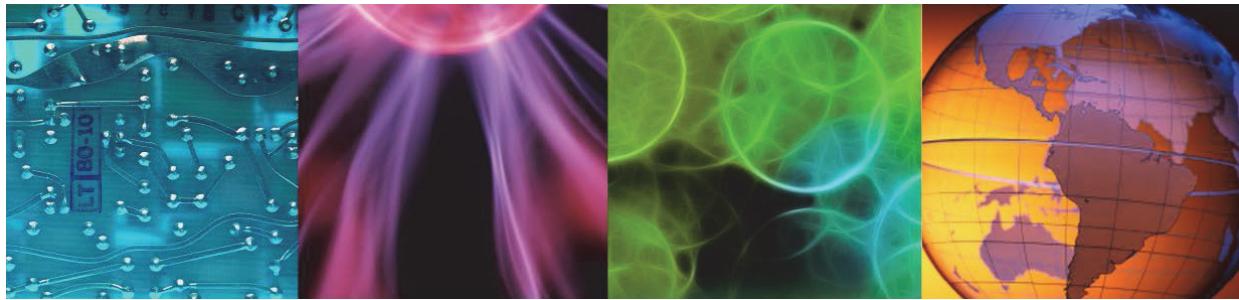
Note: The desktop does not require more configuration. The default behavior is to automatically fail back.



Lesson 3 Gateway configuration



Lesson 3 Gateway configuration



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to configure the bidirectional ObjectServer gateway.

Bidirectional or synchronizer gateway

Critical component of Tivoli Netcool/OMNibus HA configuration

Gateway startup

- Synchronizes contents of primary and backup ObjectServers
- Synchronizes data from source to destination ObjectServers

Typical Operation

- After synchronization is complete
- Gateway replicates changes between ObjectServers

© Copyright IBM Corporation 2014

21

Bidirectional or synchronizer gateway

A critical component in the HA configuration is the bidirectional gateway. The following information provides important details about gateway behavior. You need to understand how the gateway works. The term *default* is used in this lesson because users often use gateways without making many configuration changes. You need to understand what happens when the gateway is operated in that mode.

The gateway behaves differently, based on start or typical operation.

Gateway Startup

When the gateway starts for the first time, or when it reconnects to an ObjectServer after a failure, its primary function is *synchronization*. The synchronization process involves copying of data from one ObjectServer to another. The intent is to ensure that both ObjectServers contain the same event-related data. The gateway designates one ObjectServer as the *source*, where to synchronize from, and designates the other as the *destination*, where to synchronize to. The gateway uses configuration settings to make this determination. The following slide describes exactly how the gateway uses these settings to determine the *source* ObjectServer.

Typical Operation

After synchronization is complete, the gateway switches to typical operation mode. During normal operation, the role of the gateway is to replicate changes between ObjectServers. Consider what is happening in a high availability configuration when components are connected to the primary

ObjectServer. Probes periodically insert new events. The gateway replicates those new events into the backup ObjectServer. Desktop users view events, and they might change one by acknowledging the event. The change in that event is replicated to the backup ObjectServer. An automation that is running on the primary ObjectServer might modify an event. The delete_clears trigger periodically deletes events. The gateway replicates the delete commands to the backup ObjectServer and removes the corresponding events.



Important: The gateway does not replicate every change individually. Any time an event record is modified, the ObjectServer notifies the gateway with the IDUC protocol. The gateway caches a list of event records, and periodically processes all changes in a batch operation. The frequency of this batch processing is configurable. The default frequency is every 60 seconds.

ObjectServer gateway determination of resync direction

1. Check the Gate.Resync.Master property in gateway properties file. Use the ObjectServer as source if defined.
2. If Gate.Resync.Master is not set, check that one of the servers is defined with the BackupObjectServer set to TRUE. If neither server is defined as the backup, skip to step 4.
3. If Gate.Resync.Master is not set and a backup server is defined, check the ActingPrimary property for the backup ObjectServer. Use the backup ObjectServer if ActingPrimary property is TRUE.
4. If no backup server is defined, or the backup ObjectServer ActingPrimary property is FALSE, check each ObjectServer uptime. Use the ObjectServer that has been up the longest as the source.
5. If the ObjectServers uptimes are the same, check Gate.Resync.Preferred property in gateway properties file. Use the ObjectServer as source if defined.
6. If Gate.Resync.Preferred is not set, use the ObjectServer defined as ObjectServerA in the gateway properties file.

ObjectServer gateway determination of resync direction

In older versions of Netcool/OMNIbus, the gateway used the *age* of the ObjectServers to determine the source. The ObjectServer that was older, running longer, was designated as the *source*. For some companies with many events in their ObjectServers, it might take several minutes to synchronize the ObjectServer. If the backup ObjectServer happened to suffer a failure before synchronization was complete, there was the potential for lost events. When the backup ObjectServer recovers, the gateway identifies the primary ObjectServer as the *source* because it ran longer. In that case, the gateway synchronizes from primary to backup, even though the backup has events that were never synchronized to the primary. To eliminate the potential for lost events in this circumstance, a new technique was developed for determining the *source* ObjectServer.

The new technique involves the use of a property setting on the *backup* ObjectServer. The property is called **ActingPrimary**. This property is set to TRUE in the backup ObjectServer whenever the backup ObjectServer assumes the role of primary. Whenever the gateway starts, it checks the property in the backup ObjectServer. If TRUE, the backup is designated the *source*. This property is set to FALSE whenever the backup ObjectServer is not acting in the role of the primary. In that case, when the gateway starts, the primary ObjectServer is designated the *source*. The mechanism for modifying the value of this property is described in detail in subsequent slides.

Gateway data movement techniques

Transfer

- Bulk copy of a table from one ObjectServer to another
- Only happens when the transfer command is run
- Typically configured to run at gateway startup

Replicate

- Changes in the contents of table on one ObjectServer are replicated to a table on another ObjectServer
- Can configure as:
 - ALL
 - INSERTS
 - UPDATES
 - DELETES
- Changes are replicated continuously when the gateway is active
- Frequency is determined by the gateway IDUC setting

Gateway data movement techniques

The following sections have more details about gateway configuration options.

Transfer

The transfer function is used at gateway start. This function is in addition to synchronization. It provides a mechanism for the user to transfer the contents of selected tables from one ObjectServer to another. This feature provides a convenient way to automate administration of multiple ObjectServers. Users can configure this feature to transfer the tables that contain users, groups, roles, menus, and tools. With this configuration in place, the user can perform all administration tasks on the primary ObjectServer.

The bidirectional gateway transfers the corresponding tables to the backup ObjectServer whenever it starts.



Note: The gateway configuration files created by the Initial Configuration Wizard contain the commands necessary to copy the contents of these tables to minimize ObjectServer administration.

Replicate

Transfer moves the entire contents of selected tables. The replicate feature propagates only changes between ObjectServers. The user has the option of configuring not only which tables but what types of changes to replicate. For example, the user can choose not to propagate delete

requests between ObjectServers. The changes are replicated continuously while the gateway is running. The replication does not occur immediately, but rather according to a configurable frequency. The user can also configure the gateway to replicate administrative changes between ObjectServers. With this configuration, a user can add a tool or menu to the primary ObjectServer and the gateway replicates that change on the backup ObjectServer.



Note: The gateway configuration files created by the Initial Configuration Wizard contain the commands necessary to implement this behavior.

Configuring resynchronization

Four options available

Controlled by Gate.Resync.Type

- NORMAL
 - Bulk replace of the entire table
- UPDATE
 - Gateway compares contents of both ObjectServers
 - Rows in the slave that are also in the master are updated with the data from the master, if the data in the master is different from the slave.
 - Rows that are in the master but not in the slave are copied to the slave.
 - Rows in the slave that are not in the master are retained.
- TWOWAYUPDATE
 - Behaves the same way as UPDATE and then
 - The event caches of the master and slave ObjectServers are compared.
 - Any rows that are in the slave but not in the master are written to the master.
- MINIMAL
 - Similar to UPDATE
 - Additionally removes events from the *backup* that are not in the *primary*

Configuring resynchronization

You can set the Gate.Resync.Type property to one of the following values:

- **NORMAL:** For each table, the gateway deletes all the data from the subordinate ObjectServer. Then the gateway transfers the full set of tables from the master to the subordinate. With this type of resynchronization, the master and subordinate are fully synchronized. However, any table row that is in the subordinate but not in the master is lost. Additionally, if table rows are in the master and the subordinate, the copy of the row that is on the master is retained on both the master and the subordinate. Any previous updates to the row on the subordinate are lost.
- **UPDATE:** For each table, the gateway builds a cache that contains all rows in the master and subordinate ObjectServers. Then the gateway examines the contents of the cache for each table and compares the row data from the master with the row data from the subordinate. The data is resynchronized as follows:
 - a. Rows in the subordinate that are also in the master are updated with the data from the master, if the data in the master is different from the subordinate.
 - b. Rows that are in the master but not in the subordinate are copied to the subordinate.
 - c. Rows in the subordinate that are not in the master are retained.With this type of resynchronization, no events are lost, but the master and subordinate ObjectServers might not be fully synchronized.

- **TWOWAYUPDATE:** This option behaves in the same way as UPDATE. Then the gateway behaves as follows:
 - a. The event caches of the master and slave ObjectServers are compared.
 - b. Any rows that are in the slave but not in the master are written to the master.After the TWOWAYUPDATE resynchronization is completed, the master and slave contain identical rows.
- **MINIMAL:** This option behaves in the same way as UPDATE. In addition, events, that is, rows in the alerts.status table, that are in the subordinate but not in the master are marked for deletion. To mark these events for deletion, the gateway behaves as follows:
 - a. For each row of the alerts.status table in the subordinate ObjectServer that is not in the master, the OldRow field is set to 1.
 - b. The pass_deletes trigger runs on the subordinate ObjectServer and deletes all rows in which the OldRow field is set to 1.

The benefit of a MINIMAL resynchronization is that the master and subordinate ObjectServers are fully synchronized but less data is sent during the resynchronization process. MINIMAL resynchronization is less data-intensive because all the rows are not deleted and then recopied, unlike a NORMAL.

Bidirectional ObjectServer gateway configuration

Choose a name for the gateway, for example, **BI_GATE** **

Add **BI_GATE** to interface file using **nco_xigen** **

Verify both ObjectServers are defined in interfaces file **

Copy the files from **\$OMNIHOME/gates/objserv_bi** to **

\$OMNIHOME/etc, renaming them

```
BI_GATE.props  
BI_GATE.map  
BI_GATE.ObjectServera.tblrep.def  
BI_GATE.ObjectServerb.tblrep.def  
BI_GATE.startup.cmd
```

Edit the **\$OMNIHOME/etc/BI_GATE** files as necessary **

Start the gateway

```
$OMNIHOME/bin/nco_g_objserv_bi -name BI_GATE &
```

** These steps can be performed by the Initial Configuration Wizard

© Copyright IBM Corporation 2014

25

Bidirectional ObjectServer gateway configuration

The bidirectional gateway is installed when the Probe Support feature is selected during Netcool/OMNibus core installation. The gateway consists of a binary and five configuration files. These files are considered templates. Best practice is to copy the five files from their default location to another directory, and rename each file. The name of the gateway is used in the new name for each file.

The gateway requires its own entry in the interfaces file. The entry looks like an ObjectServer entry. The user determines the name of the gateway. The same rules apply in terms of how the name is formed. The standard convention for gateway names is to append _GATE to the end of the name.



Note: When the Initial Configuration Wizard is used, the wizard copies the gateway files, renames them and configures them with the best practice settings. The wizard also creates an entry in the interfaces files.

Bidirectional gateway configuration files

`gatewayname.props`

File locations, gateway name, gateway runtime behavior

`gatewayname.map`

Field mappings for all tables between source and destination

Can specify format conversions and ON INSERT ONLY

`gatewayname.ObjectServera.tblrep.def`

`gatewayname.ObjectServerb.tblrep.def`

Filter statements and resynchronization

Tables replicated during run time

`gatewayname.startup.cmd`

Commands to execute, tables to transfer at startup

© Copyright IBM Corporation 2014

26

Bidirectional gateway configuration files

The gateway property file is where the names of the ObjectServers, the user names, and passwords are defined. Gateways use an ObjectServer user name, and password for their connection. The default is root with no password. The recommendation is to create a separate user name for use by the gateway. You configure that user name and password in the gateway property file. Encrypt the password string with the nco_g_crypt utility.

An important configuration file is the map file. The map file defines how the gateway maps a column from a table in ObjectServer A to the corresponding table in ObjectServer B. There is a separate definition in the map file for every table that the gateway is to manage. The map definition is important for the alerts.status table. You want to verify that every field in this table is mapped to the same field on the other ObjectServer. The bidirectional gateway comes with a map file that contains all of the standard fields for the alerts.status table. Users can add their own custom fields to this table. When modified, the user must edit the map file in the gateway and add the custom fields. Otherwise, the contents of that field are not replicated through the gateway.

The two files that are labeled **tblrep** contain the definitions that tell the gateway which tables to replicate. By default, only three tables are replicated, and those names are in this file. The user can configure more tables for replication. This feature provides another mechanism to facilitate ObjectServer administration. By configuring table replication for users, groups, roles, the gateway replicates any changes to those tables to the other ObjectServer. With this feature enabled, you can add a user to the primary, and the gateway replicates that user to the backup.

The replication files that come with the gateway contain entries for all of the configuration tables. The definitions are commented out in these files. To use these definitions, remove the comment character (#) from the entries in the replication files and map file. Then the automated replication of

ObjectServer configurations is enabled. Replication is configured separately for each direction of transfer. The user has the option of configuring replication from ObjectServerA to ObjectServerB, ObjectServerB to ObjectServerA, or both directions.



Note: The Initial Configuration Wizard creates all of the gateway configuration files automatically.

Automations in HA configurations

Most automations should only be active on the primary ObjectServer

- All event management is taking place in the primary ObjectServer, and the changes are replicated to the backup ObjectServer
- Default gateway behavior is to replicate changes in either direction, primary to backup and backup to primary
- Any event that is changed on the backup ObjectServer by an automation is replicated to the primary ObjectServer
- Automations running on the primary and backup ObjectServers can modify the same event and cause that event to replicate back and forth through the gateway

Automations need to be active on the backup ObjectServer when the primary ObjectServer fails

- The backup ObjectServer is now the primary ObjectServer
- All event management is happening on the backup ObjectServer

Automations in HA configurations

With two ObjectServers in the high availability configuration, there are considerations for the use of automations. Automations, or triggers, perform various event management tasks, for example, deleting clear events. The bidirectional gateway replicates any event changes that occur between ObjectServers. The fact that it is a bidirectional gateway means that replication can occur in both directions: primary to backup and backup to primary. Certain automations should not be active on the backup ObjectServer while the primary ObjectServer is in use.

However, if the primary ObjectServer fails the backup assumes the role of primary. All components are connected to the backup, and all event processing takes place in the backup. In this case, all the automations must be active. A feature exists to automatically disable specific automations on the backup ObjectServer when the primary is active. This feature also enables the automations on the backup when the primary is out of service.

Automated failover and failback

Automated failover and failback feature was added to automate enabling or disabling of automations in backup ObjectServer when primary goes down or comes up

It implements this function using signal-based triggers on the backup ObjectServer

New trigger group *primary_only* has all the triggers that should be running only in primary

Customers should either add their new *primary_only* triggers to this group or extend procedures `automation_enable` and `automation_disable` to include new trigger group

Automated failover and failback

The Automated Failover and Failback feature is implemented with a combination of trigger groups, triggers, signals, and procedures. Netcool/OMNibus includes these items. Users can modify the configuration to include any custom triggers that they develop.

It is this feature that disables certain automations on the backup ObjectServer when the primary is running. And, that enables those same automations on the backup ObjectServer when the primary is down.

This feature also modifies the value of the `ActingPrimary` property on the backup ObjectServer discussed previously.

Initial setup

Primary ObjectServer		Gateway		Backup ObjectServer
Start primary ObjectServer				Start backup ObjectServer
				-Enable triggers in gateway_trigger group -Disable primary_only group
Primary is running	↔	Start bidirectional gateway	↔	Backup is running
Primary is running	↔	Resynchronization begins	↔	Backup is running
Primary is running	↔	Gateway is running	↔	Backup is running

29

Initial setup

The following information outlines how this feature is implemented.

The bidirectional gateway is the key to how this feature works. The gateway generates specific signals in the backup ObjectServer when it connects to the primary and when it loses connection to the primary.

When the gateway generates each of these signals, signal triggers run. Each signal trigger calls a procedure, and the procedure enables or disables the automations.

Three triggers must be enabled on the *backup* ObjectServer:

- `backup_counterpart_down`
- `backup_counterpart_up`
- `backup_startup`

When the gateway starts, and both ObjectServers are available, the gateway generates a signal in the backup ObjectServer. This signal causes a trigger to run. That trigger sets the **ActingPrimary** property in the backup ObjectServer to FALSE, and disables the **primary_only** trigger group.



Important: The Initial Configuration Wizard does not enable the triggers in the backup ObjectServer. They must be manually enabled.

Automated failover: Primary goes down

Primary ObjectServer		Gateway		Backup ObjectServer
Primary goes down	←	Gateway detects primary down	↔	Backup is running
		Gateway sends signal to backup gw_counterpart_down	→	-Execute trigger backup_counterpart_down -Enable primary_only group automations in backup
No response from primary	←	Gateway continues pinging to check whether ObjectServer has come up	↔	Backup ObjectServer starts acting as primary ObjectServer

Automated failover: Primary goes down

If the primary ObjectServer fails, the gateway raises the gw_counterpart_down signal on the backup ObjectServer. This signal causes the backup_counterpart_down trigger to run.

That trigger calls the automation_enable procedure. This procedure enables the **primary_only** trigger group. When that trigger group is enabled, any automation that is a member of that group starts running. By default the primary_only trigger group contains these items:

- generic_clear
- expire
- delete_clears

Users can add their own custom triggers to the primary_only group. The procedure also sets the **ActingPrimary** property to TRUE.

Automated failback: Primary comes up

Primary ObjectServer		Gateway		Backup
Primary comes up and acts as primary	↔	Gateway detects primary has come up	↔	Backup ObjectServer acts as primary ObjectServer
Primary is running		Gateway sends signal to backup gw_counterpart_up	→	-Execute trigger backup_counterpart_up -Disable primary_only group automations in backup
Primary is running	↔	Resynchronization begins	↔	Backup ObjectServer runs as backup
Primary ObjectServer acts as primary	↔	Gateway is running	↔	Backup ObjectServer runs as backup

31

Automated failback: Primary comes up

When the primary ObjectServer recovers, the gateway reconnects and generates the gw_counterpart_up signal. This signal causes the backup_counterpart_up trigger to run. That trigger calls procedure automation_disable, and it disables the **primary_only** trigger group. When the trigger group is disabled, the member triggers no longer run on the backup ObjectServer.

When the gateway starts, it checks the **ActingPrimary** property in the backup ObjectServer and determines that it is TRUE. The gateway starts synchronizing the contents of the backup ObjectServer to primary. If the backup ObjectServer happens to fail, the ActingPrimary property is still set to TRUE. When the backup ObjectServer recovers, the gateway checks ActingPrimary, finds it TRUE, and synchronizes from backup to primary.

When the gateway *completes synchronizing*, it raises a signal in the backup ObjectServer. This signal causes a trigger to run, and that trigger sets the **ActingPrimary** property to FALSE. This behavior ensures that the ActingPrimary property is only set to TRUE until the gateway completes synchronization from the backup to the primary.

The following information is critical to the correct operation of the trigger that sets ActingPrimary to TRUE. The gateway property file *must* contain a description for the gateway with the exact text as shown:

Gate.ObjectServerA.Description: 'failover_gate'
Gate.ObjectServerB.Description: 'failover_gate'

The reason that these values are critical is because the procedure that sets the ActingPrimary property to FALSE tests the description for the exact text **failover_gate**. If that exact text is not found, the procedure does not set the ActingPrimary property to FALSE.



Important: The gateway configuration files created by the Initial Configuration Wizard contain the correct text values.

Automated failover and failback key concepts

Trigger group on the backup ObjectServer (primary_only)

- Identifies which automations should be active when the backup ObjectServer assumes the role of the primary ObjectServer
- Can add custom automations to this group

System signals

- Signals are raised by the gateway
- Signals trigger automations

Automations

- Activated by a signal
- One automation enables trigger group
- Separate automation disables trigger group

Trigger group, signals, and automations bundled with Tivoli Netcool/OMNibus

- Automations are disabled by default

© Copyright IBM Corporation 2014

32

Automated failover and failback key concepts

The following information is a summary of the key concepts:

- The gateway controls the feature by generating signals.
- All of the activity takes place in the backup ObjectServer.
- This feature manipulates the state of the primary_only trigger group, which is enabled or disabled.
- The user can add custom triggers to the primary_only trigger group.
- The ActingPrimary property is set to TRUE or FALSE by this feature.
- Everything that is required: trigger groups, triggers, signals, and procedures, is included.

High availability and data integrity options

You can configure the following options to maximize data integrity:

- Controlled client failback
 - Desktop clients connected to a backup ObjectServer do not fail back to the primary ObjectServer until resynchronization is complete
- Controlled ObjectServer shutdown
 - Clients are disconnected and new connections are blocked
 - Gateways are instructed to process outstanding requests
 - When all requests are complete the ObjectServer is stopped
- Circular Store-and-Forward (SAF) probes
 - Probes are configured with multiple SAF files
 - If the ObjectServer fails, the probe stores events in the first SAF file
 - If one SAF file fills up, the probe moves to another SAF
 - When the ObjectServer recovers, the probe forwards the events in all SAF files

High availability and data integrity options

Other optional features can minimize the potential for event loss during various system transitions. The following information is an overview of three of these features.

Controlled Client Failback

This feature controls when client components attempt to reconnect to a primary ObjectServer that recovers from a failure. The first role of the gateway is synchronization when the primary recovers. In this situation, the gateway is synchronizing the primary ObjectServer. This process might take seconds or several minutes based on the number of event records in the ObjectServer. When the gateway completes synchronization, it raises a signal on the backup ObjectServer. This signal causes a trigger to run, and that trigger starts a procedure. The procedure disconnects all clients from the backup ObjectServer, desktops, and probes. When the clients disconnect, they reconnect to the primary. This behavior ensures that all clients reconnect only after synchronization is complete.

The name of the trigger that implements this behavior is `disconnect_all_clients`.



Note: The student exercises demonstrate how to configure this feature.

Controlled ObjectServer Shutdown

When an ObjectServer is taken down, client components might have events queued to send to the ObjectServer, probes, and gateways. The Controlled ObjectServer Shutdown feature incorporates extra steps. These steps ensure that all outstanding events are processed before the ObjectServer comes down.

Circular Store-and-Forward Probes

All probes can store events in a local disk file if the probe cannot communicate with an ObjectServer. In previous versions, the store-and-forward is limited to a single file with a user-defined maximum size. In the current version that behavior is changed so that the user can now configure a pool of SAF files. Users still define a maximum file size, but they also define a maximum number of files. The probe stores events in one SAF file until the maximum size is reached. It then opens a new file and stores events. The store-and-forward capability continues until it reaches the maximum number of files in the pool. The probe then starts to reuse the files in the pool. When the ObjectServer recovers, the probe forwards the events from all of the SAF files.



© Copyright IBM Corporation 2014

34

Student exercises

Perform the exercises for this unit.

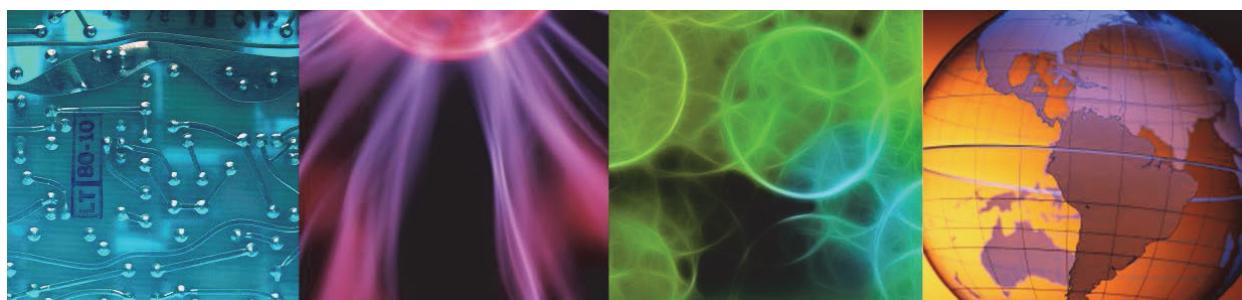
Summary

You now should be able to perform the following tasks:

- Describe the functional components of the OMNIbus high availability (HA) configuration
- Use the nco_confpack utility to clone an ObjectServer configuration
- Configure the bidirectional gateway to synchronize the event data content of two ObjectServers
- Configure the automated failover and fallback feature that manages automations on the backup ObjectServer
- Describe options for increased data integrity



9 Process control



© Copyright IBM Corporation 2014

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

Process control is the Netcool/OMNIbus facility for automatically starting, stopping, and restarting components. You can also use this facility to run commands on a local or remote system and automate remedial actions. In this unit, you learn about the features, functions, and methods to configure process control. Process control and process activity refer to the same capability.

References: SC27-6265-00 *Administration Guide*

Objectives

In this unit, you learn to perform the following tasks:

- Describe the role of *process activity* (PA) in a Tivoli Netcool/OMNibus deployment
- Describe the components in the *process activity* property file
- Configure *process activity* to start and stop Tivoli Netcool/OMNibus components
- Configure the ObjectServer to authenticate with *process activity*
- Use Visual PA to start and stop Tivoli Netcool/OMNibus components

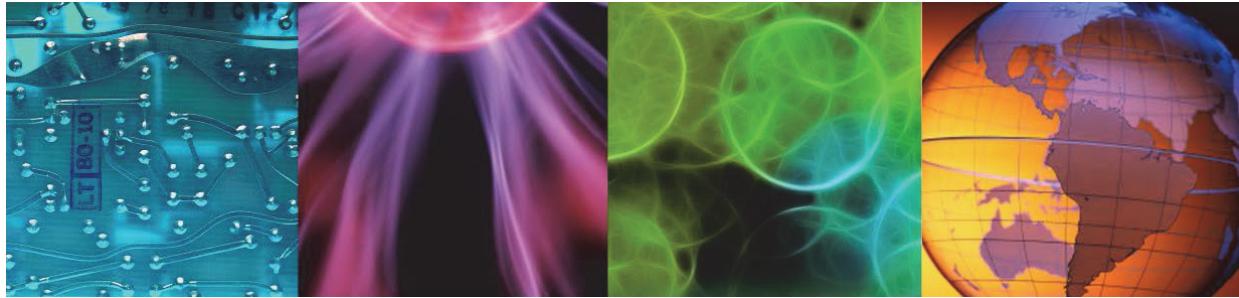
© Copyright IBM Corporation 2014

Objectives

Lesson 1 Overview



Lesson 1 Overview



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to install and configure Process control to manage the core components in the Netcool®/OMNIbus deployment. After completing this lesson, you should be able to perform the following tasks:

- Describe the functions of Process control
- Configure Process control to manage a Netcool/OMNIbus component

Process control

Process control, also known as *process activity*, provides these features:

- A way to start and shut down Tivoli Netcool/OMNibus components
- Automatic restart of failed components
- ObjectServer automations as a means to process external procedures
- Remote management of processes
- Logging of alert and restore messages to syslog

© Copyright IBM Corporation 2014

Process control

Process control provides a simple method of managing Netcool/OMNibus components. Process control is the preferred method for automatically starting Netcool/OMNibus components upon system start. It is also preferred for closing Netcool/OMNibus in a controlled fashion upon system shutdown.

Process control can optionally restart components if they fail or are accidentally stopped, which ensures that the components are kept running.

When using automations within the ObjectServer to run external effects, for example, send an email, within an action, the ObjectServer must be running under process control. The ObjectServer uses process control to route the command to the underlying operating system.

Process control can also generate *alert* and *restore* messages for the components it is managing. These messages are sent to the Syslog of the system where process control is running.

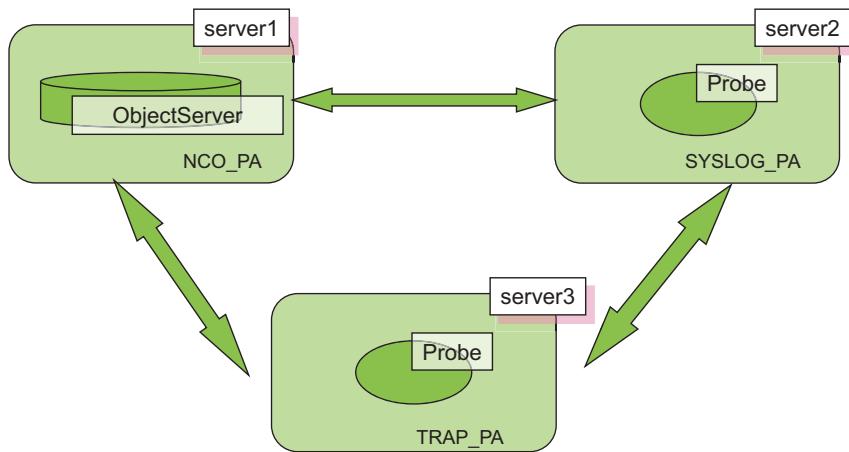
Users that install Netcool/OMNibus on Windows servers have the option of whether to use process control. The user can install Netcool/OMNibus components and use the Windows services features to automate start and shutdown. Or the user can configure process control to automate start and shutdown.

Process control setup example

Process control uses process agents

Process agents on different hosts can communicate with each other

Can remotely start Netcool components



© Copyright IBM Corporation 2014

Process control setup example

Process control consists of one or more process agents, typically one on each host in the architecture. A single-server architecture requires only one process agent.

Because process agents communicate with each other and are considered servers, they must be identified in the interfaces files on each system. Use the default agent name of NCO_PA only once per system.

A process agent can control and manage processes. It can start them automatically, stop them in a controlled fashion, and restart any that fail. You can group multiple processes and define them as services. A service defines the processes to manage and any dependencies between them.

In a multiple process agent architecture, process agents can communicate with each other to start external effects on other systems.

The diagram illustrates how process control agents can be set up to run different components of Netcool/OMNIbus on different servers.

The nco_pa.conf file

Process agents are configured in **\$OMNIHOME/etc/nco_pa.conf**

The file consists of records:

- **nco_process** defines the process (command) to run
- **nco_service** groups together processes and defines dependencies between processes
- **nco_security** (optional) restricts connection to process agent
- **nco_routing** associates host names with process agents

© Copyright IBM Corporation 2014

The nco_pa.conf file

All process agents require a configuration file, which is by default, **\$OMNIHOME/etc/nco_pa.conf**.

The nco_pa.conf file consists of four sections, described as follows:

- **nco_process** section
 - Defines each process to be managed
- **nco_service** section
 - Groups multiple processes to start and configures process dependency
- **nco_security** section
 - Specifies a list of hosts that are allowed to connect to this process agent (optional)
- **nco_routing** section
 - Associates host names with process agents



Note: The Initial Configuration Wizard creates the process activity configuration file with entries for the ObjectServers and gateway. Entries for other components, like probes, must be added manually.

The nco_process section

One *nco_process* per managed process

```
nco_process 'MasterOS'  
{ Command '$OMNIHOME/bin/nco_objserv -name NCOMS  
          -pa NCO_PA' run as 0  
          Host='omnihost'  
          Managed=True  
          RestartMsg='Master ObjectServer has been restarted'  
          AlertMsg='Master ObjectServer has gone down'  
          RetryCount=0  
          ProcessType=PaPA_AWARE }
```

Note: The ObjectServer is started as the root user in this example

© Copyright IBM Corporation 2014

The nco_process section

All *nco_process* entries must have a unique name. In this example, the name is MasterOS. The name identifies the process in the service record.

- **Command**

The command line requires the full executable and arguments within single quotation marks. You must specify the user UID the process runs under. In this example, the process runs as the UNIX root user (run as 0). Set this value to run components as non-root users.

- **Host**

The host is the one on which the process runs.

- **Managed**

True: Process is restarted if it exits.

False: Process is not restarted if it exits.

- **RestartMsg/AlertMsg**

Messages are sent to Syslog to indicate when a process starts and stops.

- **RetryCount**

The number of restart attempts to make if the process exits. A value of 0 means the process restarts indefinitely.

- **ProcessType**

Indicates whether a process is PaPA_AWARE, for example, it can communicate its status to process activity, or PaNOT_PA_AWARE. Allows process dependency, for example, not starting a probe until the ObjectServer can accept connections.

Second nco_process section

```
nco_process 'MasterOS'  
{ Command '$OMNIHOME/bin/nco_objserv -name NCMS  
          -pa NCO_PA' run as 0  
          Host='omnihost'  
          Managed=True  
          RestartMsg='Master ObjectServer has been restarted'  
          AlertMsg='Master ObjectServer has gone down'  
          RetryCount=0  
          ProcessType=PaPA_AWARE }
```

```
nco_process 'SimnetP'  
{ Command '$OMNIHOME/probes/nco_p_simnet' run as 0  
          Host='omnihost'  
          Managed=True  
          RestartMsg='Simnet Probe has restarted'  
          AlertMsg='Simnet Probe has gone down'  
          RetryCount=0  
          ProcessType=PaPA_AWARE }
```

© Copyright IBM Corporation 2014

Second nco_process section

This example shows the addition of a second process record to manage a Simnet probe. The user can add as many entries as required.

The nco_service section

The nco_service aggregates processes and sets dependency

```
nco_service 'Core'  
{  ServiceType=Master  
  ServiceStart=Auto  
  process 'MasterOS' NONE  
  process 'SimnetP' 'MasterOS' }
```

Three types of dependency

- Process is not dependent on anything
 - process 'MasterOS' NONE
- Process has a time dependency (seconds)
 - process 'MasterOS' 20
- Process is dependent on another process
 - process 'Probe' 'MasterOS'
 - Processes must be in the same service record
 - Parent process must be PA-aware

© Copyright IBM Corporation 2014

The nco_service section

Each service record that is defined must have a unique name. A service record is composed of a ServiceType, ServiceStart, and multiple process entries.

- ServiceType (Master |Non-Master)

A Master service is started before other services.

- ServiceStart (Auto|Non-Auto)

Auto indicates that the service should be started when the process agent starts. If Non-Auto, you can activate the service can manually after the process agent is started.

- Process

Specifies process record names that are started when the service starts. This name is used in the nco_process definition. A process can have a dependency that is defined. There are three types of dependency:

- No Dependency (NONE): The process is not dependent on another process or on time. The process is started automatically.
- Time Dependency: The process has a time delay that is associated with it. Delay is measured from the start of the service in seconds.
- Process Dependency: The process depends on another process. Both processes must be in the same service, and the parent process is PA-aware.

The nco_security and nco_routing sections

```
nco_security
{
    host 'hostname or IP address'
}
```

Standard routing syntax

```
nco_routing
{
    host 'omnihost' 'NAME_PA'
}
```

Secure mode routing syntax

```
nco_routing
{
    host 'omnihost' 'NAME_PA' 'username' 'password'
}
```

Note: The user name is a UNIX user, not an ObjectServer user
The password string is encrypted using nco_pa_crypt

© Copyright IBM Corporation 2014

The nco_security and nco_routing sections

Security

The optional nco_security section restricts which hosts are allowed to connect to the process agent. If no nco_security record is specified, all hosts can connect to this process agent. If an empty nco_security record is specified, for example, nco_security {}, only the current host and the hosts that are defined in the nco_routing record can connect.

To allow specific hosts, specify host names or Internet Protocol (IP) addresses in the nco_security record. These hosts are in addition to the hosts that are defined in the nco_routing record.

```
nco_security {
    host 'darkstar'
    host '192.168.0.34'
    host '193.37.192.*' }
```

In this example, connections are allowed from hosts darkstar, 192.168.0.34, and any device on subnet 193.37.192.0.

 **Note:** If the server has more than one active network interface, the nco_security entry must identify all active interfaces.

Routing

For the process agent to communicate with other hosts, the agent and host names must be defined. The process agent uses this record to map the host entry in an automation external effect to the process agent. The process agent name that you specify here must exist in the interfaces file. The name follows the same rules as an ObjectServer name.

The nco_routing record is configured differently, depending upon whether the process agent is running in secure mode. When not in secure mode, the entry contains only the host name and process control agent name:

```
host 'omnithost' 'NAME_PA'
```

When running the process agent in secure mode, you must also specify a valid UNIX user name and password for the remote system.

```
host 'omnithost' 'NAME_PA' 'username' 'password'
```

Passwords are encrypted with the utility nco_pa_crypt.



Important: If you use the process control agent to run external commands, it *must* run in secure mode and *must* have user authentication that is configured.

The process control agent can run commands (start processes) as another user. On UNIX or Linux systems, that capability requires *root* user authority. The process control agent must run as the *root* user. It can start other Netcool/OMNIbus components, such as ObjectServers, probes, and gateways, as non-root users, but it must run as the *root* user.

If the process control agent starts only processes as a single, non-root user, then the process control agent can run as that user. It is only when the process control agent starts processes as various users, that the agent must run as the *root* user.

You run the process agent as a non-root user in the student exercises.

Configuring authentication

UNIX user **netcool**, password **netcool**

```
nco_pa_crypt netcool  
ECEDBJAGBFHGD
```

PA config file (nco_pa.conf)

```
nco_routing  
{  
    host 'server1' 'NCO_PA' 'netcool' 'ECEDBJAGBFHGD'  
}
```

ObjectServer properties

```
PA.Name: 'NCO_PA'  
PA.Password: 'ECEDBJAGBFHGD'  
PA.Username: 'netcool'
```

© Copyright IBM Corporation 2014

Configuring authentication

Process control authentication implements a level of security to ensure that only authorized components can connect to the agent. You must configure authorization in the process control agent *and* the ObjectServer:

- Process Control Agent

A valid UNIX or Linux user is the basis for the authentication. You must encrypt the password with the nco_pa_crypt utility. Take the output from that utility and paste the string into the nco_routing definition.

- ObjectServer

When a trigger runs a procedure that contains a request to run an external command, the ObjectServer passes that request to the process control agent. The ObjectServer connects to the process control agent, and the agent, when running in secure mode, requests authentication. The ObjectServer must know the user name and password that authentication requires. You must configure three ObjectServer properties:

- PA.Name

The name of the process control agent that references the interfaces file.

- PA.Password

Contains the encrypted string for the password, which nco_pa_crypt creates.

- PA.Username

The UNIX or Linux user name

This configuration is important if you want to run external commands with triggers, for example, sending an email. If the configuration is not correct, the trigger runs and runs the procedure, but the external command does not run. If you look in the log file for the ObjectServer, you see a message that indicates an authentication failure, connection refused. This message is an indication that the authentication parameters are not configured correctly.

You implement this configuration in the student exercise.

Process agent daemon

Machines requiring process control must run a process agent daemon

Perform these tasks on each process agent daemon host:

- Edit **\$OMNIHOME/etc/nco_pa.conf**
- Make an entry for the process agent daemon (_PA) via **nco_xigen**
- Run the process agent daemon as the root user
 - Solaris syntax
`$OMNIHOME/bin/nco_pad -name NAME_PA`
 - Linux syntax
`$OMNIHOME/bin/nco_pad -name NAME_PA -authenticate PAM`

© Copyright IBM Corporation 2014

Process agent daemon

A process agent daemon must run on all systems to enable starting of processes. This daemon manages process control hosts, services, and processes and the dependencies, restarts, and timing of processes.

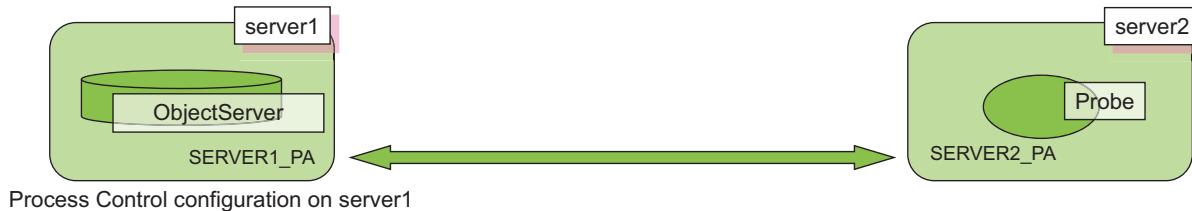
On the process agent host, edit the nco_pa.conf file. Using nco_xigen ensures that there is a valid entry for the process agent daemon. The entry should end in _PA, should be unique, and should reference a unique host and port combination.

Use this command to run the process agent daemon:

```
$OMNIHOME/bin/nco_pad -name NAME_PA
```

Linux requires **-authenticate PAM** if it requires interaction with process activity. Also, under Linux, in the **/etc/pam.d** directory, copy the file system-auth to netcool.

Running external procedure on server2



Process Control configuration on server1

```
nco_routing
{
    host 'server1' 'SERVER1_PA' 'netcool' 'ECEDBJAGBJFHGD'
    host 'server2' 'SERVER2_PA' 'netcool' 'ECEDBJAGBJFHGD'
}
```

Interfaces file on server1

```
[SERVER1_PA]
{
    Primary:      server1 4200
}
[SERVER2_PA]
{
    Primary:      server2 4200
}
```

© Copyright IBM Corporation 2014

Running external procedure on server2

This slide depicts two servers: server1 and server2. The following section describes what you must configure on each system if you want to be able to configure a trigger on server1 to run a command on server2.

Configure server1 in this way:

1. Process control agent, SERVER1_PA, is configured to run in secure mode.
2. The nco_routing section contains an entry for host server1, with the process agent SERVER1_PA, and requires a user name of netcool, and password object00 (encrypted).
3. The nco_routing section contains an entry for host server2, with the process agent SERVER2_PA, and requires a user name of **netcool** and password **object00** (encrypted).
4. The interfaces file contains an entry for SERVER1_PA, host name server1, and port number 4200.
5. The interfaces file contains an entry for SERVER2_PA, host name server2 and port number 4200.
6. The ObjectServer has the following properties configured:
 - PA.Name: SERVER1_PA
 - PA.Password: object00 (encrypted)
 - PA.Username: netcool

Configure server2 in this way:

1. Process control agent, SERVER2_PA, is configured to run in secure mode.
2. The nco_routing section contains an entry for host server1, with the process agent SERVER1_PA, and requires a user name of **netcool** and password **object00** (encrypted).
3. The nco_routing section contains an entry for host server2, with the process agent SERVER2_PA, and requires a user name of **netcool** and password **object00** (encrypted).
4. The interfaces file contains an entry for SERVER1_PA, host name server1, and port number 4200.
5. The interfaces file contains an entry for SERVER2_PA, host name server2, and port number 4200.

Assume that a procedure is configured in the ObjectServer that runs a command on host server2. The following section describes what happens when that procedure runs:

1. When the ObjectServer starts, it connects to the process control agent. The ObjectServer property PA.Name: SERVER1_PA tells the ObjectServer the name of the process control agent. The ObjectServer queries the interfaces file to determine that SERVER1_PA is running on host server1 and listening on port 4200. The ObjectServer sends a request to server1 over port 4200.
2. SERVER1_PA on server1 receives the connection request from the ObjectServer and requests authentication. The ObjectServer uses the following properties:
 - PA.Password: object00 (encrypted)
 - PA.Username: netcool, to determine the user name and password to sendSERVER1_PA uses the information in the nco_routing entry for server1 to verify that the ObjectServer provides the correct user name and password. If the information is correct, the connection is established.
3. The procedure runs inside the ObjectServer and specifies that a command is to run on host server2. The ObjectServer passes the request to SERVER1_PA on server1.
4. SERVER1_PA on server1 receives the request to run a command on host server2. SERVER1_PA uses the information in nco_routing to determine that the process control agent name associated with host server2 is SERVER2_PA.
5. SERVER1_PA on server1 queries the interfaces file and determines that SERVER2_PA is running on host server2, and listening on port 4200. SERVER1_PA sends a request to host server2 over port 4200.
6. SERVER2_PA running on host server2 receives the request from server1, and requests authentication.
7. SERVER1_PA running uses the information in the nco_routing entry to determine that the correct user name is **netcool** and password is **object00** (encrypted) when communicating with host server2.

8. SERVER2_PA receives the credentials, and uses the information in nco_routing to verify the credentials. If the authentication is correct, SERVER2_PA runs the requested command on host server2 as the requested user.

Process control command line utilities

These utilities require UNIX user to be in **ncoadmin** group

Each command requires PA agent name, UNIX user, and password

Service status	<code>\$OMNIHOME/bin/nco_pa_status</code>
Stop everything	<code>\$OMNIHOME/bin/nco_pa_shutdown</code>
Stop a service or process	<code>\$OMNIHOME/bin/nco_pa_stop</code>
Start a service or process	<code>\$OMNIHOME/bin/nco_pa_start</code>
Add a service or process to process control dynamically	<code>\$OMNIHOME/bin/nco_pa_addentry</code>

© Copyright IBM Corporation 2014

Process control command line utilities

Command-line functions are provided to manage processes that the process agents control. To use these functions, you create the UNIX group *ncoadmin* and add users that require access to the process agent. When accessing these functions, you supply a UNIX user name and password for authentication. The UNIX user name must be a member of the *ncoadmin* group.

- Service Status

Retrieves the status of any service in the process agent:

```
$OMNIHOME/bin/nco_pa_status
- server Name of the process agent.
- user Name of the UNIX user to run the command.
- password Password of the UNIX user.
```

Example:

```
$OMNIHOME/bin/nco_pa_status -server NCO_PA -user netcool -password netcool
```

- Stop process control and optionally stop all processes that are controlled by process control.

```
$OMNIHOME/bin/nco_pa_shutdown
```

Example:

```
$OMNIHOME/bin/nco_pa_shutdown -server NCO_PA
-user netcool -password netcool
```

- Start a service or process.

```
$OMNIHOME/bin/nco_pa_start
  - service Service_Name Name of the service to stop
  - process Process_Name Name of the process to stop
```

Example:

```
$OMNIHOME/bin/nco_pa_start -server NCO_PA -user netcool -password netcool
  -process MasterOS
```

- Stop a service or process.

```
$OMNIHOME/bin/nco_pa_stop
```

Example:

```
$OMNIHOME/bin/nco_pa_stop -server NCO_PA -user netcool -password netcool
  -process MasterOS
```

- Add a service or process.

```
$OMNIHOME/bin/nco_pa_addentry
```

You must supply all process details on the command line. To obtain help, specify the –help option.

Configuring automated startup

Verify that process control starts whenever the machine restarts

- Configure, test, and run all processes under process control
- Install the startup scripts provided in
\$OMNIHOME/install/startup/<ARCH>install
 - Run the installation script for the required architecture as root
 - Enter the name of the process agent, user name, and password
 - Specify whether the process agent is to run in secure mode

Configuring automated start

After all of the components are configured, tested, and run under process control, verify that the Netcool/OMNIBus processes start automatically when the system restarts. Netcool/OMNIBus uses standard start scripts, which are found in the **\$OMNIHOME/install/startup** directory.

Run the installation script that is relevant to the system. You must provide the process agent name, user name, and password, and whether the process agent must run in secure mode.

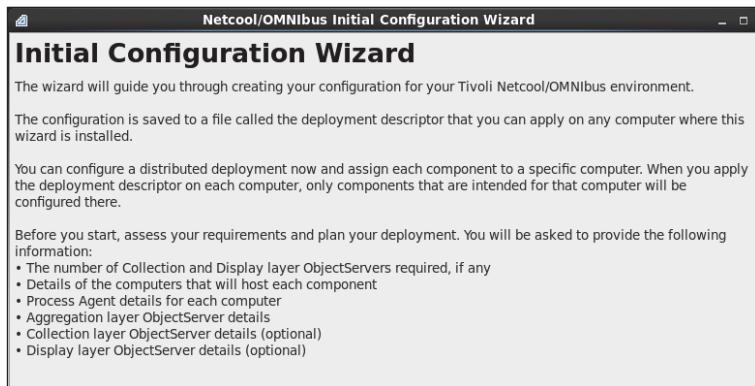


Note: You must run the script as the *root* user because the utility modifies files in the **/etc** directory.

Initial Configuration Wizard

The wizard creates the process agent configuration file automatically with entries for these items:

- Primary ObjectServer
- Backup ObjectServer
- ObjectServer gateway



© Copyright IBM Corporation 2014

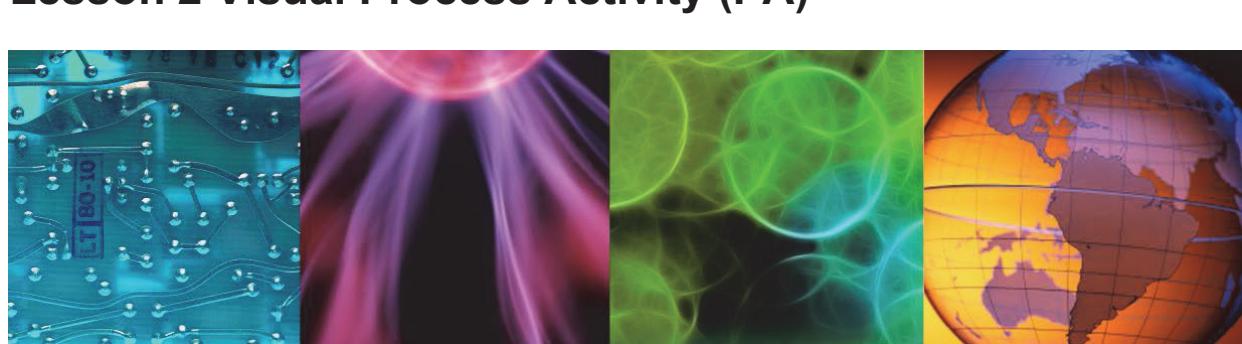
Initial Configuration Wizard

The Initial Configuration Wizard can perform most of the configuration steps for the use of process activity, as in the following examples:

- The wizard can create entries in the interfaces file to identify the process agent, for example, SERVER1_PA.
- The wizard can create the process agent configuration file with entries for ObjectServers and gateways.
- When high availability is configured, the wizard can create interface file entries for multiple process agents, for example, SERVER1_PA and SERVER2_PA. It can also create the individual process agent configuration files with the respective components.



Lesson 2 Visual Process Activity (PA)



© Copyright IBM Corporation 2014

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

In this lesson, you learn about the graphical user interface that is used to configure and control the process control component. After completing this lesson, you should be able to perform the following tasks:

- Describe the features of Visual PA
- Connect to Visual PA
- Create a process
- Start a new process

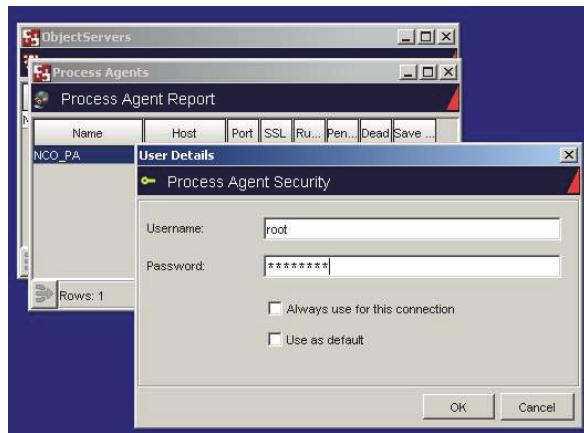
Visual PA: Logging in

Visual PA is a GUI where you configure and control process agents and processes

Start from the Administration GUI

PA must already be running

Log in with system (not Tivoli Netcool/OMNibus)
user name and password



© Copyright IBM Corporation 2014

Visual PA: Logging in

Visual PA is a graphical user interface to configure, and control process agents, and processes. Visual PA is started from the Netcool/OMNibus Administrator main login screen. To connect to a process control agent, it must be configured in the interfaces file.

Visual PA can connect only to a running process agent, which means that you must create a minimal text-based configuration in the **nco_pa.conf** file. Thereafter, you can use the graphical Visual PA to do the remainder of your configuration.

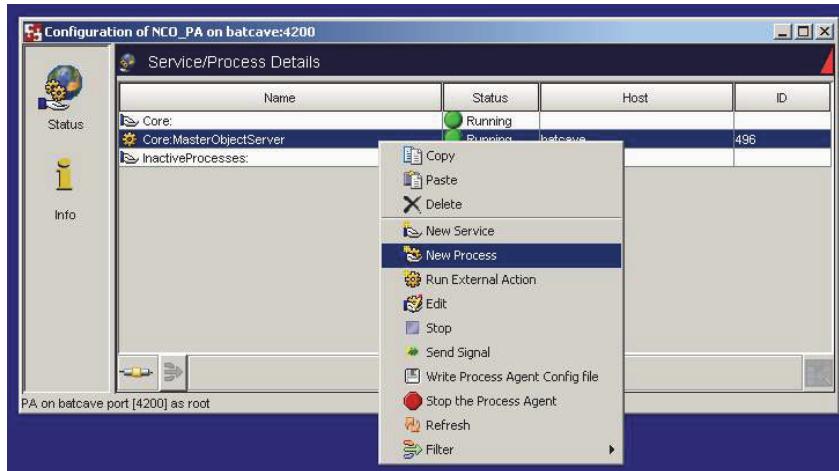
As with all process control utilities, you must log in with the system (not ObjectServer) user name and password.



Important: If your deployment contains multiple servers, and a process agent is running on each server, you can connect to any one of those process agents with the Administrator utility.

Visual PA: Adding a process

You can add processes and services through the visual PA GUI



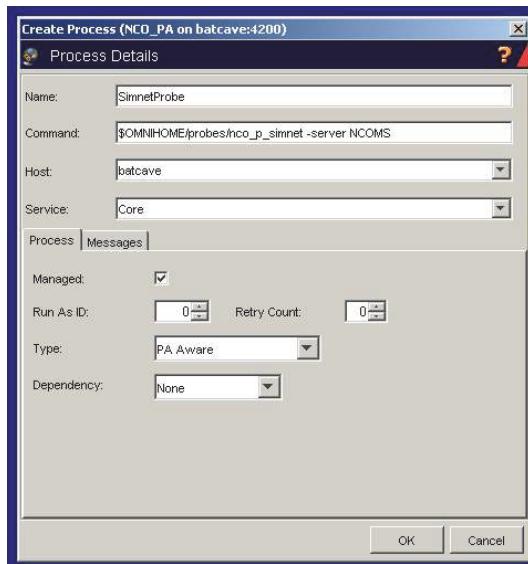
© Copyright IBM Corporation 2014

Visual PA: Adding a process

The Visual PA configuration screen shows all configured process and service records. You right-click a record to see a submenu with options to add new records or modify existing ones. You also have the option of reusing a definition by copying and pasting.

Visual PA: New process configuration

Configure same information as found in the `nco_pa.conf` process record



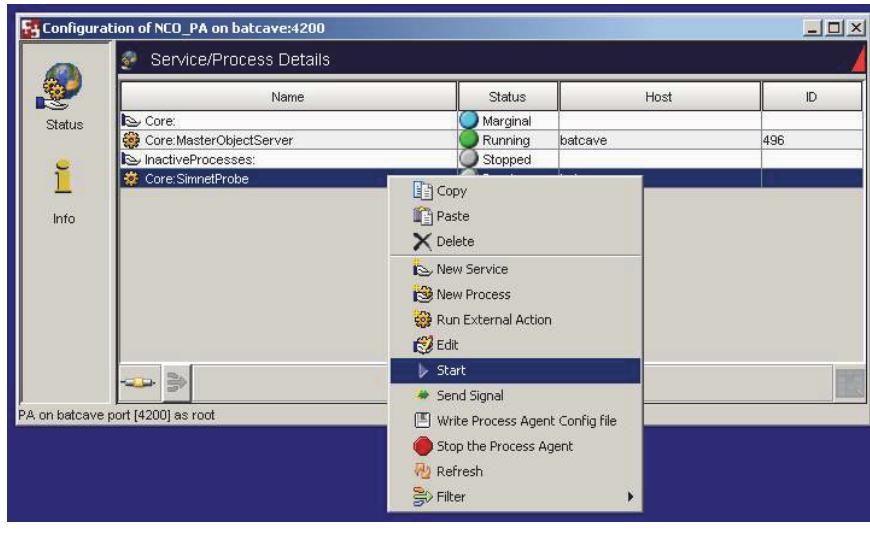
© Copyright IBM Corporation 2014

Visual PA: New process configuration

The configuration window that opens when adding or modifying a process record contains the same information as in the text-based `nco_pa.conf` file. Complete the fields as appropriate and click **OK** to apply.

Visual PA: Starting the process

After you add processes, you can control them through the visual PA GUI



© Copyright IBM Corporation 2014

Visual PA: Starting the process

When you add a process record, it is not automatically started. Right-click the record, and select **Start** from the submenu.

You can also stop the process from this submenu. However, do not use the red stop sign icon. Selecting the stop sign stops *all* process control and optionally, all the processes that it started. Instead, use the gray box labeled **Stop**.

When adding an entry through Visual PA, you are doing the equivalent of the text-based nco_addentry command. You are adding the process record dynamically to the already running PA, and not saving it to the configuration file.

After you configure and test all your processes, click **Write Process Agent Config File** option to save your configuration to the nco_pa.conf file. The Visual PA GUI also shows the status of all running PA processes.

Student exercises



© Copyright IBM Corporation 2014

Student exercises

Summary

You now should be able to perform the following tasks:

- Describe the role of *process activity* (PA) in a Tivoli Netcool/OMNibus deployment
- Describe the components in the *process activity* property file
- Configure *process activity* to start and stop Tivoli Netcool/OMNibus components
- Configure the ObjectServer to authenticate with *process activity*
- Use visual PA to start and stop Tivoli Netcool/OMNibus components

© Copyright IBM Corporation 2014

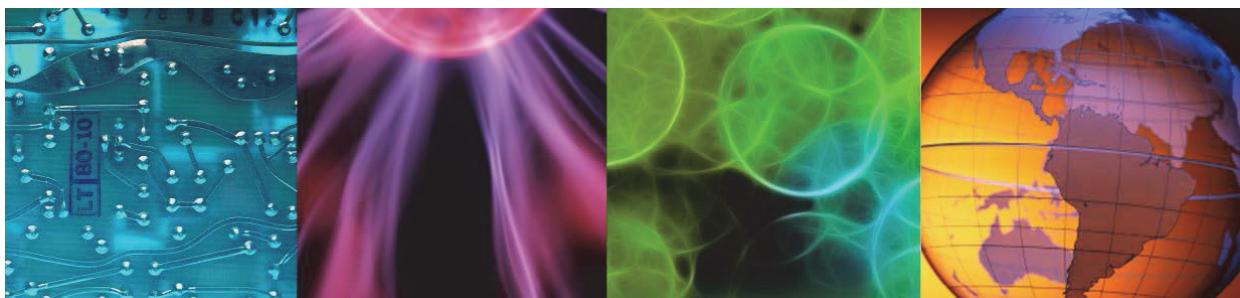
Summary



10 Event archiving



10 Event archiving



© Copyright IBM Corporation 2014

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

Netcool/OMNIbus includes a package of prebuilt event reports. In this unit, you learn how to configure the Netcool/OMNIbus solution to implement historical event retention and reporting.

References: SC22-5408 *Tivoli Netcool/OMNIbus Gateway for JDBC Reference Guide*

Objectives

After completing this unit, you should be able to perform the following tasks:

- Describe functional architecture
- Install and configure the JDBC gateway
- Install and configure Tivoli Common Reporting
- Import bundled event reports into Tivoli Common Reporting

© Copyright IBM Corporation 2014

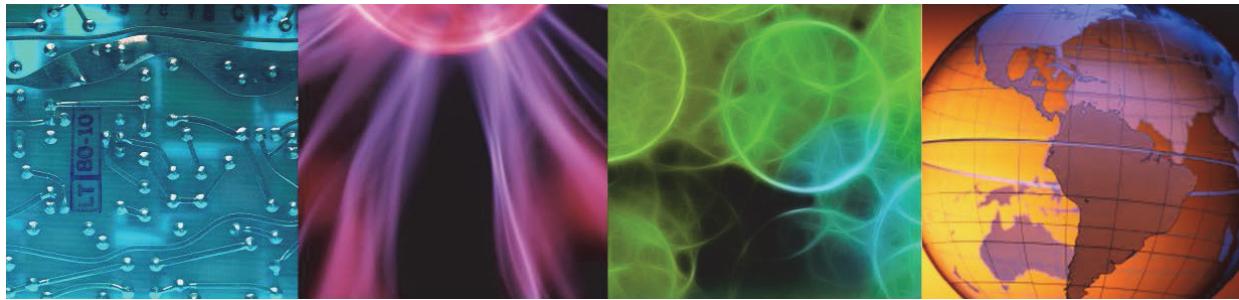
Objectives



Lesson 1 Overview



Lesson 1 Overview



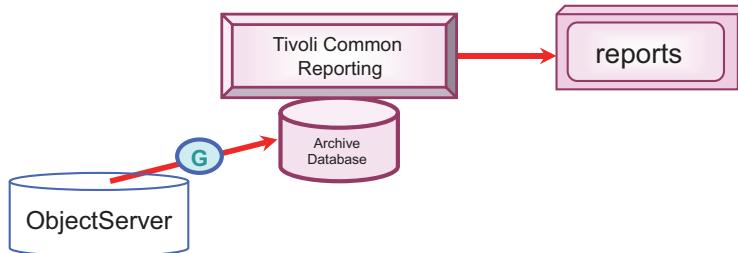
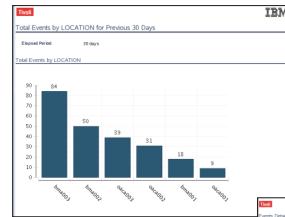
© Copyright IBM Corporation 2014

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

In this lesson, you learn what components are required for historical event retention and reporting. After completing this lesson, you should be able to describe the functional components of the event archive solution.

Historical event reporting

- Event records archived to database
- Gateway for JDBC
 - Oracle, Sybase, DB2, MSSQL
- Tivoli Common Reporting
- Standardized reporting tool
 - Prebuilt reports



© Copyright IBM Corporation 2014

Historical event reporting

Tivoli® Common Reporting Event Reporting is based on the historical event records that the Netcool®/OMNIbus gateway collects. These records are archived to a user-configured database. The archive gateway supports various databases, including: DB2®, Oracle, Sybase, MS SQL, and others.

Components

Archive database

- Customer provided
- DB2, Oracle, Sybase, MS SQL, Informix*, MySQL*

Gateway for JDBC

- Installed separately
 - Netcool/OMNIBus JDBC Gateway Configuration Scripts
 - Netcool/OMNIBus Gateway for JDBC

Tivoli Common Reporting

- Installed separately
- Not bundled with Netcool/OMNIBus

Netcool/OMNIBus Tivoli Common Reporting reports

- Bundled with Netcool/OMNIBus

* Audit mode only

© Copyright IBM Corporation 2014

Components

The database that retains event records is customer-provided. The following databases are supported:

- DB2
- Oracle
- Sybase
- MS SQL
- Sybase
- Informix
- MySQL

ObjectServer event records are written to the archive database by the Netcool/OMNIBus Gateway for JDBC.

Tivoli Common Reporting generates *event reports*. Netcool/OMNIBus includes a package of prebuilt reports. You can modify these reports, and create new reports with Tivoli Common Reporting.

Implementation

Expected installation time: 1 ½ hours, not including installation of Netcool/OMNibus or back-end database

Typical sequence of installation steps:

1. Archive database
 - Install Netcool/OMNibus JDBC Gateway Configuration Scripts
 - Define REPORTER database and tables
2. Gateway for JDBC
 - Install Netcool/OMNibus Gateway for JDBC
 - Configure, start, and verify gateway operation
3. Tivoli Common Reporting
 - Install
 - Configure
4. Event reports
 - Import report package
 - Configure data source
 - Verify reports work correctly

© Copyright IBM Corporation 2014

Implementation

This slide illustrates the steps that are required to implement event archiving, and reporting. The slide assumes that Netcool/OMNibus is installed and running. The customer-provided database is installed and running. The steps that are shown here build on those components.

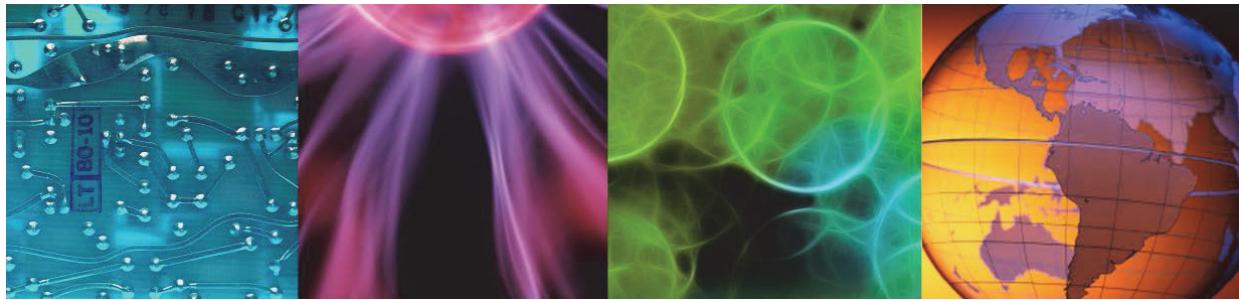
Subsequent lessons in this unit provide more detail on each of these steps.



Lesson 2 Tivoli Common Reporting



Lesson 2 Tivoli Common Reporting

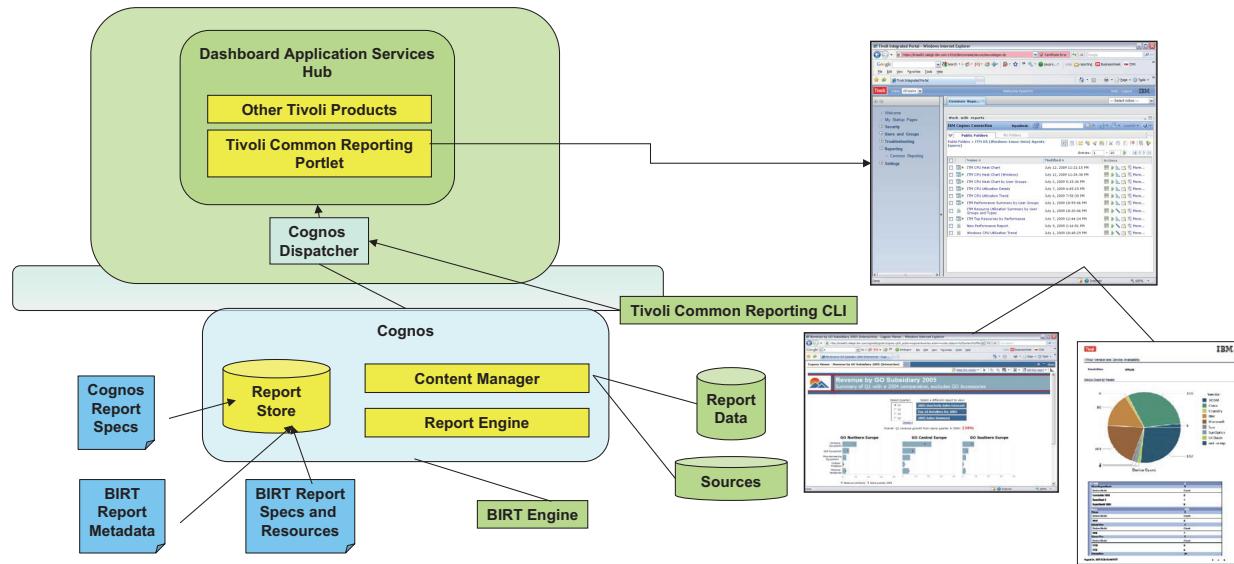


© Copyright IBM Corporation 2014

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

In this lesson, you learn how to install and initialize Tivoli Common Reporting.

Tivoli Common Reporting architecture



© Copyright IBM Corporation 2014

Tivoli Common Reporting architecture

There are two major functional components in the Tivoli Common Reporting architecture. The first is the user interface, which is Dashboard Application Services Hub. You have the option when installing Tivoli Common Reporting of installing a complete copy of Dashboard Application Services Hub, or reusing an existing copy. With Netcool/OMNIbus, the installation of Web GUI creates an installation of Dashboard Application Services Hub. So when you install Tivoli Common Reporting, you reuse this copy of Dashboard Application Services Hub.

The second functional component is the reporting engine. The reporting engine for Tivoli Common Reporting is Cognos®. If a copy of the required version of Cognos exists, you can install Tivoli Common Reporting and reuse the existing copy of Cognos.

In addition to the Cognos report engine, Tivoli Common Reporting also includes a copy of Business Intelligence Reporting Tool (BIRT). BIRT is included for compatibility with older versions of Tivoli Common Reporting.



Important: Tivoli Common Reporting requires a database that is called the Report Store. The only supported database type is DB2®. This database is used to store all of the Tivoli Common Reporting artifacts. The customer data that is used for reporting can be of various database types, not just DB2.

Installing Tivoli Common Reporting

Currently not bundled with Netcool/OMNibus

Requires Jazz for Service Management

- Dashboard Application Services Hub

Requires DB2

- Report Content Store database

Installation options:

- Install Jazz for Service Management and Tivoli Common Reporting
- Install Tivoli Common Reporting into existing Jazz for Service Management

Installation runs approximately 1 hour

© Copyright IBM Corporation 2014

Installing Tivoli Common Reporting

The installation software for Tivoli Common Reporting is not included with Netcool/OMNibus. You must download and install the installation file separately.

There are two options for installation of Tivoli Common Reporting. If you are installing Tivoli Common Reporting on a clean server, one option is to install Jazz™ for Service Management and Tivoli Common Reporting at the same time. The installation utility for Jazz for Service Management provides the option to install both components at one time.

If a copy of Jazz for Service Management exists, the second option is to install Tivoli Common Reporting into that existing deployment.

Installing Jazz and Tivoli Common Reporting

Installed with IBM Installation Manager

- Expand the installation files
 - Jazz for Service Management
 - WebSphere
 - DB2
 - Tivoli Common Reporting
- Start the installer
`./launchpad.sh`

The screenshot shows the 'Specify Source Locations' tab where 'DB2, WebSphere Application Server, and Tivoli Common Reporting source locations' are listed. A callout bubble points to the 'installation files detected automatically' section. The 'Select Operations' tab shows a grid of components and their current, target, and operation status. A callout bubble points to the 'disable components not wanted' section.

Component	Current	Target	Operation
Administration Services	1.1.0.3	1.1.0.3	<input type="button" value="None"/>
Service provider	1.1.0.3	1.1.0.3	<input type="button" value="None"/>
User Interface	1.1.0.3	1.1.0.3	<input type="button" value="None"/>
Registry Services	3.1.0.0	1.1.0.3	<input type="button" value="Install"/>
Reporting Services	3.1.0.0	1.1.0.3	<input type="button" value="None"/>
Security Services	3.1.0.0	3.1.0.3	<input type="button" value="Install"/>
Visualization Services	3.1.0.3	3.1.0.3	<input type="button" value="Install"/>
IBM DB2 Enterprise Server Edition	10.5.0.3	10.5.0.3	<input type="button" value="Install"/>
IBM WebSphere Application Server	8.5.0.1	8.5.0.1	<input type="button" value="Install"/>

© Copyright IBM Corporation 2014

Installing Jazz and Tivoli Common Reporting

Installing Tivoli Common Reporting into an existing Jazz deployment

Tivoli Common Reporting does not use IBM Installation Manager

A custom installation utility is included with the installation files

DB2 must be installed and running

Required for the Report Store database



© Copyright IBM Corporation 2014

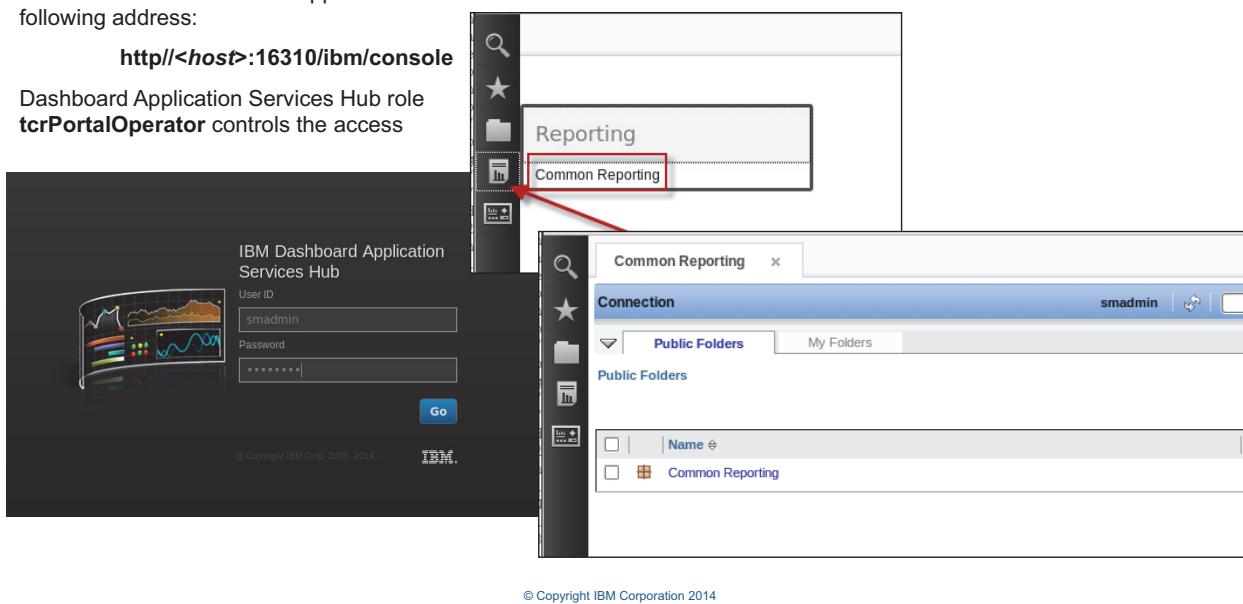
Installing Tivoli Common Reporting into an existing Jazz deployment

Accessing Tivoli Common Reporting

You can access Dashboard Application Services Hub at the following address:

<http://<host>:16310/ibm/console>

Dashboard Application Services Hub role **tcrPortalOperator** controls the access



Accessing Tivoli Common Reporting

Access to Tivoli Common Reporting requires the **tcrPortalOperator** role. The installation adds this role to the **smadmin** user automatically. For all other users, you must manually add the role.

Log in to Dashboard Application Services Hub, expand **Reporting**, and click **Common Reporting**. The installation creates the Common Reporting package. This package contains a single report that lists the contents of the Cognos Report Store database. At the conclusion of the installation, this report is the only report in the database.

Student exercises



© Copyright IBM Corporation 2014

Student exercises

The installation of Tivoli Common Reporting runs for approximately 1 hour. To make the best use of time, you start the installation process, and then the instructor continues with the lecture material. After the lectures are complete, you return to this exercise to complete the remaining steps.



Lesson 3 Gateway for JDBC

Lesson 3 Gateway for JDBC



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to install and configure the Netcool/OMNIbus Gateway for JDBC. You also learn how to use the supplied SQL files to create the archive database structure. After completing this lesson, you should be able to perform the following tasks:

- Describe the components in the archive database structure.
- Create the archive database.
- Install and configure the Netcool/OMNIbus Gateway for JDBC.

Gateway for JDBC audit mode of operation

Creates a new row in the database table for every new alert and alert update

The target database can contain multiple rows for each alert, depending on its update history

In audit mode, existing data in the target database is never updated or deleted

Database can grow large quickly

© Copyright IBM Corporation 2014

Gateway for JDBC audit mode of operation

The Netcool/OMNIbus Gateway for JDBC supports two modes of operations: Audit and Reporting. The primary difference between the two modes is the volume of data that is written to the archive database.

Recall that the Netcool/OMNIbus ObjectServer provides an event reduction feature called deduplication. This feature results in a single ObjectServer event, regardless of how many times the same problem occurs. If a device generates the same alarm repeatedly, link down for example, the ObjectServer contains a single link down event. Some users want to be able to produce a report that itemizes every link-down occurrence. This type of report is possible by running the gateway in Audit mode.

In Audit mode, the gateway writes a new record to the archive database for every event occurrence. For example, if a device generates 100 link-down alarms, the ObjectServer contains a single link-down event. With Audit mode, the gateway writes 100 link-down event records to the archive database. You must be careful to ensure that the volume of data that is sent to the archive database is not too large.

Gateway for JDBC *reporting* mode of operation

The target database contains one row for each new alert and that row is updated whenever the alert is updated

The gateway essentially replicates the source status table from the ObjectServer into the target database

Inserts, updates, and deletes are mapped as follows:

- New alerts cause a new row, unique to the alert, to be inserted into the target database
- Alert updates cause the existing alert row to be updated with the new alert values
- Alert deletes cause the existing alert row to be updated with a deletion timestamp.

This activity is implemented with database triggers

Event reports assume that data is archived using *reporting* mode

© Copyright IBM Corporation 2014

Gateway for JDBC reporting mode of operation

When the gateway runs in Reporting mode, the data sent to the archive database essentially mimics the data in the ObjectServer. For example, if a device generates 100 link-down alarms, the ObjectServer contains a single event record. The gateway creates a single corresponding record in the archive database. In this case, the volume of data that is sent to the archive database is considerably less than in Audit mode.

In addition to the differences in the quantity of data that is saved in the archive database, there are structural differences in the database between Audit mode and Reporting mode.



Important: The reports that are delivered with Netcool/OMNIbus are created to use the event archive data that is produced by running the gateway in **Reporting** mode.

Archive database configuration

Install Netcool/OMNIbus JDBC Gateway Configuration Scripts

For DB2, import: db2.reporter.sql

- Additional SQL files available for other database types
- Creates:
 - REPORTER database
 - Tables
 - Views
 - Stored procedures

Supplied SQL creates database table with all standard ObjectServer event record columns

- You must modify supplied SQL if ObjectServer event record is modified

© Copyright IBM Corporation 2014

Archive database configuration

The gateway configuration files include SQL files that create the database structure on the archive database. There are different files for each of the supported database types. There are also different files for Audit mode and Reporting mode.

Because the supplied reports are based on the data that is stored with Reporting mode, the remainder of the information in this lesson assumes Reporting mode operation.

The SQL file that is provided creates the database REPORTER. That database consists of tables, views, and stored procedures.

REPORTER database configuration

Tables

- REPORTER_DETAILS
- REPORTER_JOURNAL
- REPORTER_STATUS
 - Typically requires modifications to add custom ObjectServer fields
- AUDIT Tables – used to track changes
 - REP_AUDIT_OWNERUID
 - REP_AUDIT_OWNERGID
 - REP_AUDIT_SEVERITY
 - REP_AUDIT_ACK
- Tables used to convert integers to text
 - REPORTER_NAMES
 - REPORTER_GROUPS
 - REPORTER_MEMBERS
 - REPORTER_CLASSES
 - REPORTER_CONVERSIONS
- Tables used for prompts during report generation
 - REP_SEVERITY_TYPES
 - REP_TIME_PERIODS

© Copyright IBM Corporation 2014

REPORTER database configuration

The supplied SQL file creates 14 tables in the REPORTER database. The first three tables that are shown here, REPORTER_DETAILS, REPORTER_JOURNAL, and REPORTER_STATUS, store the event data from the corresponding ObjectServer tables: alerts.details, alerts.journal, and alerts.status.

The next nine tables on the slide are described in subsequent slides.

The last two tables, REP_SEVERITY_TYPES, and REP_TIME_PERIODS, provide static values for prompts when a user runs a report.

REPORTER database configuration, continued

Database triggers

- REP_AUDIT_INSERT
- REP_AUDIT_UPDATE
- REP_AUDIT_ACK

Stored procedures

- Acknowledged
- Deletedat
- Ownergid
- Owneruid
- Severity

Views

- REP_REFERENCE_DATE
- REP_AUDIT
- STATUS_VW

© Copyright IBM Corporation 2014

REPORTER database configuration, continued

The supplied SQL file creates database triggers, stored procedures, and database views.

The triggers, and stored procedures are used with the AUDIT tables shown previously. Details on their use are provided in a subsequent slide.

The views are a convenience for reporting. The REPORTER database contains numerous tables. The views that are shown here implement SQL JOIN statements to combine various tables into meaningful relationships for reporting purposes.

AUDIT tables

In reporting mode, there is only one archive database event record (REPORTER_STATUS) for every ObjectServer event record (alerts.status)

To record the history of changes for certain event record columns, the REPORTER database maintains separate tables:

- REP_AUDIT_OWNERUID
- REP_AUDIT_OWNERGID
- REP_AUDIT_SEVERITY
- REP_AUDIT_ACK

Each table records the old and new values for a specific column, with a time stamp

Stored procedures trigger based upon column changes and update the respective AUDIT table

© Copyright IBM Corporation 2014

AUDIT tables

Do not confuse the term AUDIT as used here with the Audit mode of gateway operation. They are not the same.

When the gateway runs in Reporting mode, data that is stored in the archive database mimics the ObjectServer deduplication function. For every event record in the alerts.status ObjectServer table, there is a corresponding record in the REPORTER_STATUS table on the archive database. For reporting, you want to be able to show when the event changed in certain ways. For example, a link-down event is created with a Severity of 5 (critical). When the issue is resolved, a link-up event is generated. The ObjectServer generic_clear trigger correlates the link-down event with the link up event and changes the Severity to 0 (clear). At that point, the gateway updates the corresponding record in the REPORTER_STATUS table and sets Severity to 0. Based on this single record, it is not possible to produce a report that details when the link went down and when the link came up. The AUDIT tables help provide this type of report.

Four tables in the archive database capture changes to some ObjectServer event columns, specifically Severity, OwnerUID, OwnerGID, and Acknowledged. These tables contain a column that contains the previous value and a column that contains the current value. The tables also contain a column that contains the time stamp of when the change occurred and columns that provide a database link to the corresponding record in the REPORTER_STATUS table. Database triggers and stored procedures populate the AUDIT tables.

The following simple example shows how this process works.

Assume that a device generates a link-down alarm, and this alarm produces a link-down event with Severity of 5. The gateway creates a record in REPORTER_STATUS with all of the same

information, except that there is an extra column in the table to store the *original* severity, in this case, 5. When the link issue is resolved, the device generates a link up alarm, and this alarm produces a link up event in the ObjectServer. The generic_clear trigger correlates the two events, and sets Severity to 0. The gateway detects this change, and updates the Severity column in the REPORTER_STATUS table to 0. The column that holds the *original* severity still contains a 5. When the REPORTER_STATUS table changes, a database trigger is run. This trigger adds a record to the REP_AUDIT_SEVERITY table. The record contains the old severity (5), the new severity (0), the time stamp of the change, and a link to the corresponding record in REPORTER_STATUS. In this simple example, the event changes Severity once: Critical to Clear. If the link goes up and down multiple times, or flaps, the Severity changes numerous times. The contents of the REP_AUDIT_SEVERITY table details every change in Severity.

A database view joins the REPORTER_STATUS table to the REP_AUDIT_SEVERITY table. You can use this view in a report to detail every change in the Severity column for any event record. Use the same technique for the OwnerUID, OwnerGID, and Acknowledge fields.

Text conversion tables

The ObjectServer can automatically associate text with an integer column value, for example, severity.

To replicate this same behavior for reporting, the REPORTER database maintains separate tables:

- REPORTER_NAMES
- REPORTER_GROUPS
- REPORTER_MEMBERS
- REPORTER_CLASSES
- REPORTER_CONVERSIONS

The database replicates the performance using a JOIN to combine the REPORTER_STATUS table with the respective conversion table or tables.

The contents of these database tables must be maintained to preserve the integer-to-text relationships.

© Copyright IBM Corporation 2014

Text conversion tables

Several columns in the ObjectServer event record contain integer data, for example, Severity. The ObjectServer can associate text to the integer through a conversion. This feature provides a convenient way to minimize the volume of data that is stored and still provide readable text. The event records that are written to the archive database contain the same integer values as the records in the ObjectServer. To provide a similar textual conversion facility for reporting purposes, the REPORTER database contains a number of *conversion* tables. The text that is stored in one of these conversion tables is associated with the corresponding column in the REPORTER_STATUS table with an SQL join statement in a database view. The result is that a report contains the text *Critical*, *Major*, or *Minor*, instead of 5, 4, or 3.

These tables in the REPORTER database must be maintained to preserve their integer-to-text relationships. Fortunately, the corresponding ObjectServer tables do not change frequently. One technique for maintaining the tables is to configure the gateway to transfer the contents of the ObjectServer tables to the corresponding archive database tables. This configuration is demonstrated in the student exercise for this unit.

Gateway installation and configuration

Install Netcool/OMNIbus Gateway for JDBC

- This installation creates \$OMNIHOME/bin/hco_g_jdbc

Define a gateway name

- Same rules as ObjectServer name
- Add to interfaces file

Copy gateway configuration files and rename

```
cd $OMNIHOME/gates/jdbc  
cp reporting.jdbc.map $OMNIHOME/etc/JDBC_GATE.map  
cp reporting.G_JDBC.props $OMNIHOME/etc/JDBC_GATE.props  
cp jdbc.rdrwtr.tblrep.def $OMNIHOME/etc/JDBC_GATE.rdrwtr.tblrep.def  
cp jdbc.startup.cmd $OMNIHOME/etc/JDBC_GATE.startup.cmd
```

Install vendor JDBC drivers

© Copyright IBM Corporation 2014

Gateway installation and configuration

The gateway is installed with the IBM Installation Manager utility. Two packages must be installed to use the gateway. One package contains the gateway configuration files. The other package contains the gateway itself.

The gateway is configured like the bidirectional ObjectServer gateway. The best practice is to copy the configuration files from the \$OMNIHOME/gates/jdbc directory to \$OMNIHOME/etc, and rename each file so that it contains the gateway name. You must define the gateway name in the interfaces file.

One more step is to install the vendor JDBC drivers. These drivers vary based on the database type: DB2®, Oracle, Sybase, or MS SQL.

JDBC_GATE.map file

Provides the mapping between table/column in ObjectServer database to the corresponding table or column in the archive database

Default file contains all of the standard ObjectServer column names

- Must be revised if ObjectServer is modified

Contains mapping statements for conversion tables:

- REPORTER_NAMES
- REPORTER_GROUPS
- REPORTER_MEMBERS
- REPORTER_CLASSES
- REPORTER_CONVERSIONS

With this mapping, you can configure the gateway to update these tables

© Copyright IBM Corporation 2014

JDBC_GATE.map file

The title of this slide assumes that the gateway is named JDBC_GATE.

The map file provides the mapping between columns in the ObjectServer tables to the corresponding columns in the archive database tables. In many cases, you must modify the mapping for the REPORTER_STATUS table because the user adds more columns to the ObjectServer event record.



Note: In addition to modifying the map file to include the new columns, you must also modify the REPORTER_STATUS table in the archive database and add corresponding columns.

JDBC_GATE.props file

Define access to the ObjectServer:

Gate.RdrWtr.Server *string* ObjectServer name [NCOMS]

Gate.RdrWtr.Password *string* ObjectServer password

Gate.RdrWtr.Username *string* ObjectServer user name

Note: user name and password only required if ObjectServer running in secure mode

Define access to archive database, for DB2:

Gate.Jdbc.Driver: '**com.ibm.db2.jcc.DB2Driver**'

Gate.Jdbc.Url: 'jdbc:db2://omnithost:50001/reporter'

Gate.Jdbc.Username: 'db2inst1'

Gate.Jdbc.Password: '**object00**'

Gate.Jdbc.ReconnectTimeout: 30

© Copyright IBM Corporation 2014

JDBC_GATE.props file

The title of this slide assumes that the gateway is named JDBC_GATE.

The property file controls the behavior of the gateway. You must modify the file to define the access criteria for the archive database, and you must modify it to define the access criteria for the ObjectServer.



Note: The gateway uses the interfaces file to locate the ObjectServer. In a high-availability configuration, you use the *virtual* ObjectServer name in the property file. By using the virtual name, you ensure that the gateway can fail over and fail back between the two ObjectServers.

JDBC_GATE.rdrwtr.tblrep.def file

Defines what types of changes to which ObjectServer tables are *replicated* to the corresponding archive database table

Default configuration is to replicate *all* changes to alerts.status, and alerts.journals to the archive database

Option exists to configure the gateway to update one or more **alerts.status** columns after the gateway updates the archive database

```
REPLICATE ALL FROM TABLE 'alerts.status'  
    USING MAP 'StatusMap'  
    AFTER IDUC DO 'Archived_Flag=1';
```

Can use to *flag* a record as having been archived

© Copyright IBM Corporation 2014

JDBC_GATE.rdrwtr.tblrep.def file

The title of this slide assumes that the gateway is named JDBC_GATE.

This file configures the gateway to replicate changes from specific tables in the ObjectServer to the corresponding tables in the archive database. The default is alerts.status, and alerts.journals.

An optional configuration parameter is AFTER IDUC DO. When you use this parameter, the gateway updates a column in the ObjectServer alerts.status table after the corresponding event record is written to the archive database. By using a user-defined column in the alerts.status table, this parameter can modify the column in a way that indicates that the record is archived.

This configuration is demonstrated in the student exercise for this unit.

JDBC_GATE.startup.cmd file

Can use to configure the gateway to perform some operation whenever the gateway starts

Convenient way to configure the gateway to copy conversion tables to archive database

```
TRANSFER FROM 'alerts.conversions' TO 'REPORTER_CONVERSIONS' DELETE USING TRANSFER_MAP  
ConversionsMap;  
TRANSFER FROM 'alerts.objclass' TO 'REPORTER_CLASSES' DELETE USING TRANSFER_MAP ObjectClassesMap;  
TRANSFER FROM 'master.groups' TO 'REPORTER_GROUPS' DELETE USING TRANSFER_MAP GroupsMap;  
TRANSFER FROM 'master.members' TO 'REPORTER_MEMBERS' DELETE USING TRANSFER_MAP MembersMap;  
TRANSFER FROM 'master.names' TO 'REPORTER_NAMES' DELETE USING TRANSFER_MAP NamesMap;
```

© Copyright IBM Corporation 2014

JDBC_GATE.startup.cmd file

The title of this slide assumes that the gateway is named JDBC_GATE.

This file can contain commands that the gateway runs only when the gateway starts. The file contains a series of TRANSFER statements that are commented out. These TRANSFER statements correspond to the conversion table that was described previously. Removing the comment character from these commands configures the gateway to copy the contents of each table to the archive database whenever the gateway starts.

Gateway operation

Start the gateway

```
$OMNIHOME/bin/nco_g_jdbc -name JDBC_GATE
```

Check the log file for issues

```
$OMNIHOME/log JDBC_GATE.log
```

- Most common issue is database connection
- Examine property file and verify access information

Verify that the archive database is being populated

Add gateway to *process activity*

© Copyright IBM Corporation 2014

Gateway operation

You start the gateway by running the following command:

```
$OMNIHOME/bin/nco_g_jdbc -name JDBC_GATE &
```

Check the log file for any issues. The most common issues involve connecting to the archive database, or connecting to the ObjectServer.

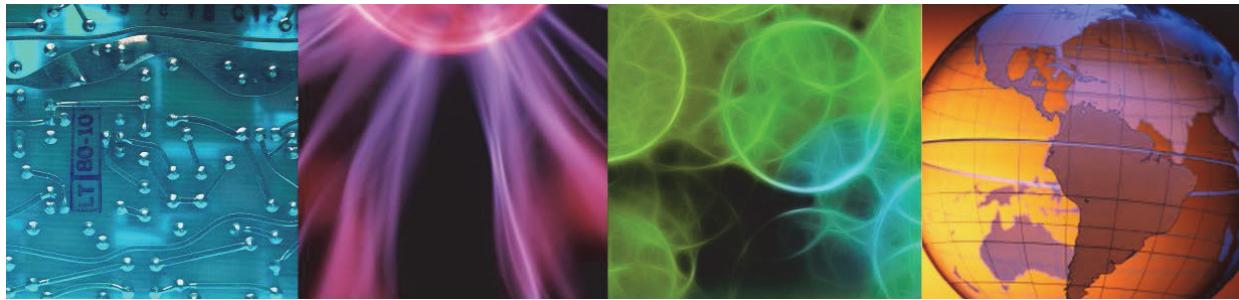
To verify that the gateway is functioning correctly, examine the REPORTER_STATUS table in the archive database. If there are records in this table, the gateway is working. If you configured AFTER IDUC DO, then check the corresponding column in the alerts.status table to verify that the gateway is modifying the column as defined.

In a production environment, you add the gateway to process activity to ensure that it starts automatically.

Lesson 4 Event reports



Lesson 4 Event reports



© Copyright IBM Corporation 2014

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

In this lesson, you learn how to import the Netcool/OMNIbus report package into Tivoli Common Reporting. After completing this lesson, you should be able to perform the following tasks:

- Import the Netcool/OMNIbus report package.
- Create a data source that maps to the REPORTER database.
- Run a Netcool/OMNIbus report.

Event reports

Bundled with Netcool/OMNibus

\$OMNIHOME/extensions/tcr_event_reports/Netcool_OMNibus.zip

Copy to import directory

/opt/IBM/JazzSM/reporting/cognos/deployment/

Import the package

- Log in to Dashboard Application Services Hub
- Select Common Reporting
- Run import wizard

Create data source definition

Verify reports

© Copyright IBM Corporation 2014

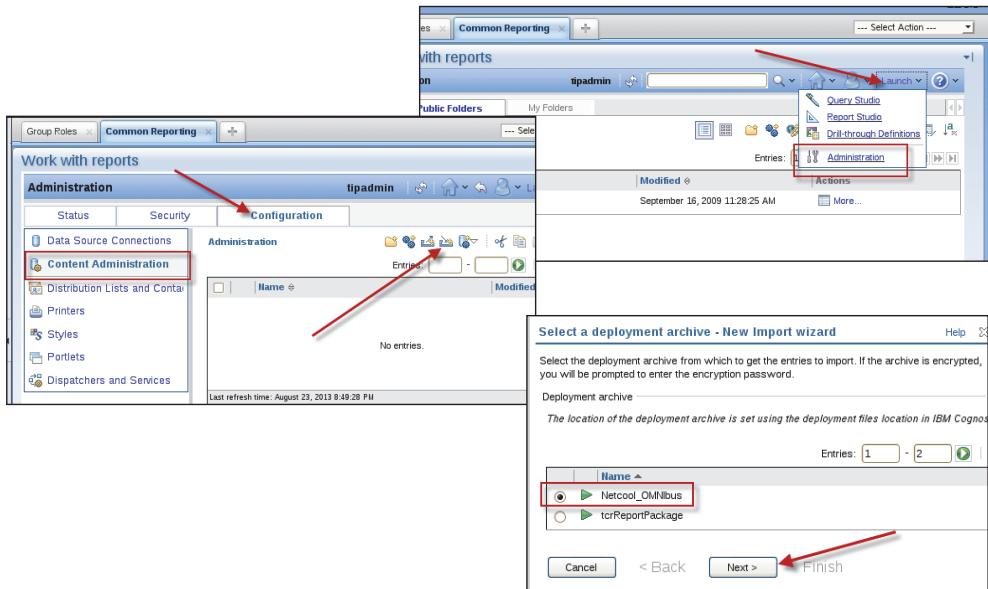
Event reports

The Netcool/OMNibus reports are distributed as a compressed file that you find in the following location:

\$OMNIHOME/extensions/tcr_event_reports

To import the reports, you copy the report package file to a specific directory, log in to Dashboard Application Services Hub, and run the package import wizard. After the package is imported, you must create a data source definition. This definition contains the information that is required to access the REPORTER database. After the data source is defined, you can use the reports.

Import report package

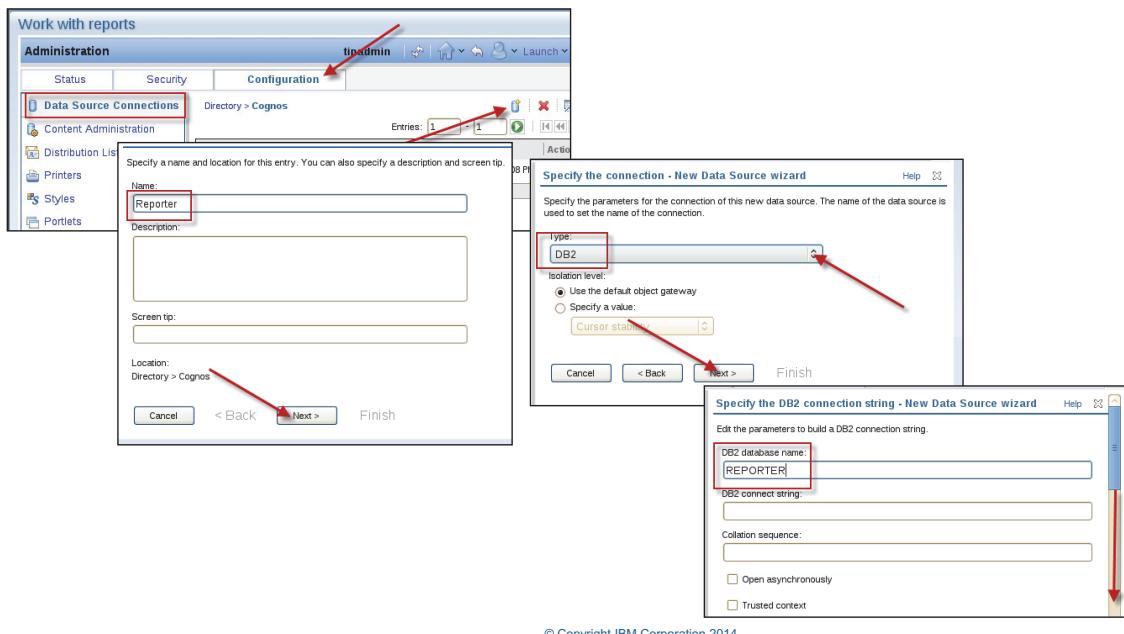


© Copyright IBM Corporation 2014

Import report package

You must copy the report package file to a specific directory. Then you log in to Dashboard Application Services Hub, expand **Reporting**, and select **Common Reporting**. The Tivoli Common Reporting home page opens. Next, you click the arrow next to **Launch**, and select **Administration**. Click the **Configuration** tab, and select **Content Administration**. Click the icon that is shown in the screen capture to import a report package, which starts the wizard. If the package is in the correct directory, the name is shown in the list of packages. Click the box to select the package, and click **Next**. There are several more screens in the report wizard. You keep the default settings on each screen, and click **Next**. When the wizard is complete, the reports are available.

Create data source

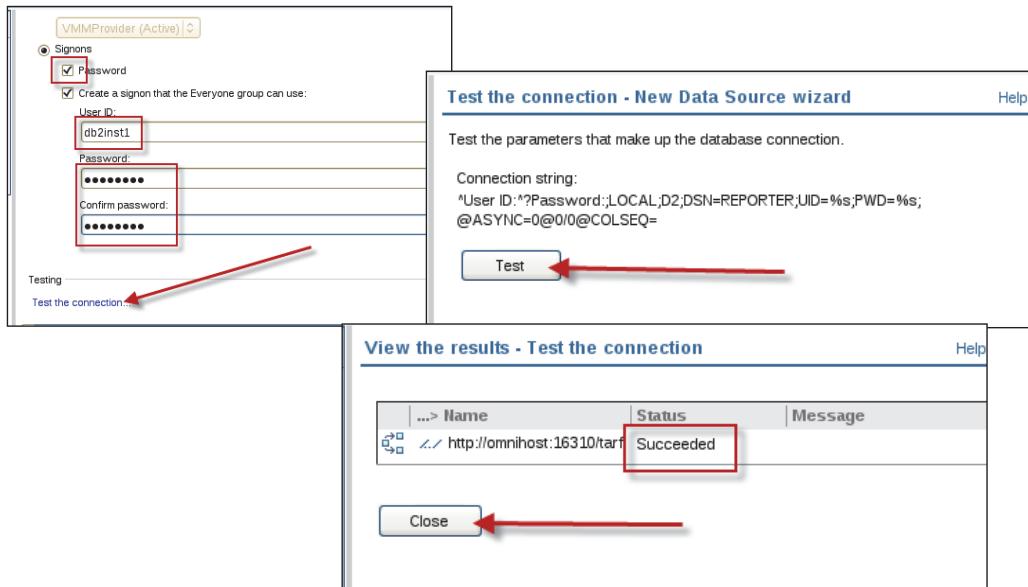


Create data source

A Tivoli Common Reporting data source defines the access to a database. For the Netcool/OMNIbus reports, that database is the REPORTER database. The data source name must be REPORTER because that name is configured in the Netcool/OMNIbus reports.

From the **Configuration** tab, you select **Data Source Connections**. Click the icon that is shown in the screen capture to create a new data source. You enter REPORTER for the data source name, then click **Next**. You select the type of database, DB2 in the screen capture, and click **Next**. You enter REPORTER again. This value is the name of the database. Then scroll to the bottom of the page.

Test data source



© Copyright IBM Corporation 2014

Test data source

Enter the database user name and password, and then click the line that says **Test the connection**. On the next screen, click **Test**, and verify that the test is successful.

Close the test screens, and click **Finish** to create the data source definition.

Verify reports

The screenshot shows the 'Common Reporting' interface. In the left sidebar, under the 'Reporting' category, there is a link to 'Common Reporting'. The main area is titled 'Work with reports' and shows a list of public folders. One folder, 'Netcool_OMNibus', is highlighted with a red box and has a red arrow pointing to it. The second screenshot shows a detailed view of the 'Netcool_OMNibus' folder, listing various reports like 'Acknowledgement Details', 'Event Distribution' (which is also highlighted with a red box), and 'Event Severity'. Below these screenshots is a third one showing a 'Grouping Criteria' dialog box. It includes fields for 'End Date' (set to Aug 27, 2013, 11:59 PM), 'Group by' (set to 'Node'), and 'Number of groups to include' (set to 10). A red arrow points from this dialog to the 'Finish' button at the bottom.

© Copyright IBM Corporation 2014

Verify reports

When you return to the report package page, the Netcool_OMNibus package is listed. Click the name, which is a hyperlink. The list of Netcool/OMNibus reports opens. To run a report, click the report name, which is a hyperlink. A page opens that provides the option to select various prompt values, such as a start date and end date for the event records included in the report. Scroll to the bottom of the page, and click **Finish** to run the report. The report request passes to the Cognos report engine, and the engine generates the report.

Verify reports



Verify reports

The report output opens in a new browser window.

Reports included with product

Event_Distribution

Use this report to view the entities, probes, locations, and so on, that generate the most events over a defined period of time, to identify which parts of your system require attention or remedial action

Event_Selection

Use this report to view the number of events by severity and day, over a given date range, for particular criteria

Event_Details

Use this report to show the full details of a single event to determine the source of problems in your system

Event_Severity

Use this report to view the events that have a severity greater than or equal to a particular severity, and a first occurrence by day, over a defined period of time

© Copyright IBM Corporation 2014

Reports included with the product

The report package contains six reports. Those reports are described on this slide and the following slide.

Reports included with product

Acknowledgement_Summary

Use this report to view the average time that it takes operators to acknowledge a new event

Acknowledgement_Details

Use this report to view the sum of acknowledgement times for events over time, selected by various criteria

© Copyright IBM Corporation 2014

Reports included with the product

Student exercises



© Copyright IBM Corporation 2014

Student exercises

Make sure you complete the exercises related to installing Tivoli Common Reporting. Then complete the remaining exercises for this unit.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Describe functional architecture
- Install and configure the JDBC gateway
- Install and configure Tivoli Common Reporting
- Import bundled event reports into Tivoli Common Reporting

© Copyright IBM Corporation 2014

Summary

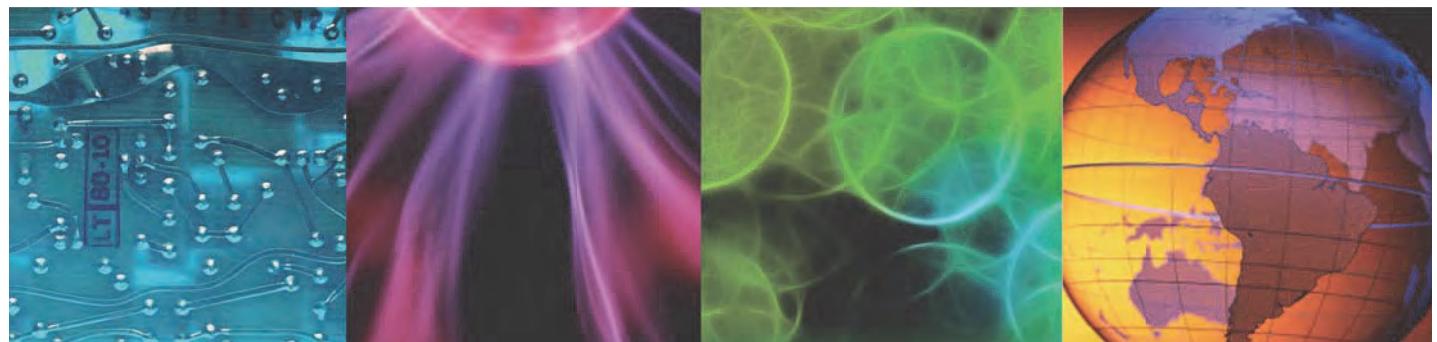


APPENDIX A Software installation files

The following is a list of the eAssembly names for the product installation files that are used in this course.

- IBM Tivoli Netcool OMNIbus 8.1 Core – Linux 64bit Multilingual (CN05EML)
- IBM Tivoli Netcool OMNIbus 8.1 WebGUI – Linux 64bit Multilingual (CN05FML)
- IBM DB2 Server V10.5 for Linux on AMD64 and Intel EM64T systems (x64) Multilingual (CIXV0ML)
- Netcool/OMNIbus 8 Plus Probe for SNMP (nco-p-mttrapd 18_0) Multiplatform English (CN02JEN)
- Netcool Knowledge Library (NcKL) Enhanced Probe Rules V4.2 Multiplatform English (CN09KEN)
- Netcool/OMNIbus 8 Plus Probe for Syslog (nco-p-syslog 7_0) Multiplatform English (CN02EEN)
- Netcool/OMNIbus 8 Plus JDBC Gateway Configuration Scripts (Reporting Mode: nco-g-jdbc-reporting-scripts 1_0) Multiplatform English (CIZZ2EN)
- Netcool/OMNIbus 8 Plus Gateway for JDBC (nco-g-jdbc 5_0) Multiplatform English (CIZZ4EN)
- Jazz for Service Management V1.1.0.3 for Linux Multilingual (CIXA2ML)
- IBM Tivoli Common Reporting V3.1.0.2 for Linux Multilingual (CIXA6ML)
- IBM WebSphere Application Server V8.5.0.1 for Jazz for Service Management for Linux Multilingual (CIES6ML)

TN025 1.0



ibm.com/training

Authorized
IBM | Training



Printed in Ireland