



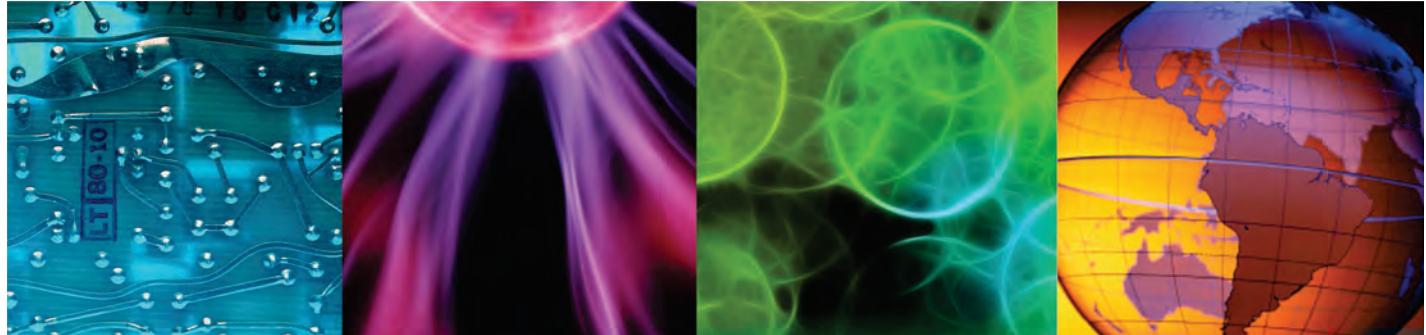
IBM Training

IBM Tivoli Netcool/OMNIbus 8.1 Administration and Maintenance

Student Notebook

Course code TN035 ERC 2.0

May 2016



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

© Copyright IBM Corp. 2016. 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	xiii
About the student	xiv
Learning objectives	xiv
Course agenda	xv
1 Introduction to Netcool/OMNibus Administration.....	1
Objectives	2
Lesson 1 Overview	3
Administration categories	4
Tivoli Netcool/OMNibus software components	5
Administration by component	6
Lesson 2 Stopping and starting components	7
Stopping and starting the ObjectServer	8
Stopping and starting the ObjectServer with process activity	9
Visual process activity	10
Stopping and starting the ObjectServer with visual process activity	11
Stopping and starting the ObjectServer from the command line	12
Controlled ObjectServer shutdown	13
Starting and stopping a probe	14
Starting and stopping a gateway	15
Starting and stopping WebSphere	16
Lesson 3 Backing up and restoring	17
Backup requirements	18
ObjectServer backup and recovery considerations	19
ObjectServer backups	20
Configuration Packager utility	21
ObjectServer Report Generator utility	22
Failure recovery with high availability	23
Desktop backups	24
Probe backups	25
Gateway backups	26
Web GUI backups	27
Web GUI backups, continued	28
Web GUI recovery with load balancing cluster	29
Lesson 4 Applying maintenance	30
Overview	31
Downloading maintenance packages	32
Searching for maintenance on Fix Central	33
Searching for maintenance, continued	34

Installing a maintenance package	35
Installing a maintenance package, continued	36
Lesson 5 Modifying behavior	37
Overview	38
Configuring ObjectServer behavior	39
Configuring probe behavior	40
Configuring gateway behavior	41
Configuring Web GUI behavior	42
Student exercises	43
Summary	44
2 ObjectServer administration	45
Objectives	46
Lesson 1 ObjectServer structure	47
Object Server databases	48
Database and table restrictions	49
ObjectServer structure: alerts.status	50
ObjectServer structure: alerts.details	51
ObjectServer structure: alerts.journal	52
Default ObjectServer fields	53
Default ObjectServer fields, continued	55
Data storage	57
Data storage, continued	58
Region storage and physical checkpoints	59
Region storage, continued	60
Deduplication	61
Deduplication behavior	62
Properties	63
Properties, continued	64
Viewing properties with nco_sql	65
Viewing properties with nco_sql, continued	66
Modifying an ObjectServer property	67
Important properties	68
Important properties, continued	70
ObjectServer OSLC interface overview	72
Important OSLC properties	73
Lesson 2 Modifying the ObjectServer structure	74
Basic ObjectServer administration	75
Connection wizard	76
Tivoli Netcool/OMNIbus Administrator	77
Navigating within the ObjectServer Administrator	78
The User tab	79
The Menu tab	81
The Automation tab	82
The Visual tab	83
The System tab	84
Adding a column to the ObjectServer event record table	85
Defining column conversion values	86

Lesson 3 Creating ObjectServers	87
Creating a default ObjectServer	88
Creating a custom ObjectServer	89
Creating the best practice aggregation ObjectServer	90
Student exercises	91
Summary	92
3 Probes	93
Objectives	94
Lesson 1 MIB Manager	95
MIB Manager overview	96
MIB Manager functions	97
Launching MIB Manager	98
MIB Module view	99
Object Identifier (OID) view	100
Details view	101
Importing a MIB file	102
Generating rules	103
Generating SNMP traps	104
Netcool/OMNIbus Knowledge Library	105
Installed files	106
Installed files, continued	108
Installed files, continued	109
Enabling probe to use vendor rules	110
Adding rules from MIB Manager for a new vendor	111
Adding rules from MIB Manager for an existing vendor	112
Lesson 2 Probe high availability	113
peer-to-peer overview	114
Configuring peer-to-peer mode	115
peer-to-peer operation	116
Example configuration	117
Lesson 3 Remote probe administration	118
Features to facilitate remote administration	119
Rules file caching	120
Bidirectional communication	121
Probe registry	122
Bidirectional probe communication usage	123
Rereading rules: single probe	124
Rereading rules: all probes	125
Rereading rules: configuration	126
Rereading rules: usage	127
Rereading rules: example	128
Student exercises	129
Summary	130
4 Automations	131
Objectives	132
Lesson 1 Basic SQL	133
ObjectServer SQL defined	134

Starting the SQL interactive interface	135
ObjectServer SQL: selecting events	137
Logical operators	138
Comparison operators	139
Comparison operators: LIKE	140
Regular expressions	141
More comparison operators	142
Modifying data with UPDATE	143
Creating data with INSERT	145
Removing data with DELETE	146
ORDER BY clause	147
SQL workbench	148
Lesson 2 ObjectServer automations	149
Trigger applications	150
Trigger types	151
Trigger general settings	152
More trigger general settings	153
Trigger groups	154
WHEN clause	155
Example of a WHEN clause	156
Trigger actions	157
Lesson 3 SQL code blocks	158
SQL code blocks	159
SQL commands that are commonly used	160
SQL code blocks: %user variables	161
SQL code blocks: SET statement	162
Example of a SET statement	163
SQL code blocks: IF THEN ELSE	164
SQL Code Blocks: CASE/WHEN Statement	165
SQL code blocks: FOR EACH ROW	166
SQL code blocks: Example of FOR EACH ROW	167
SQL code blocks: FOR	168
SQL code blocks: Example of FOR	169
Trigger response sequence	170
Lesson 4 Database triggers	171
Database trigger	172
Database trigger: Settings and action	173
Special row variables: Overview	174
Special row variables for operation and timing	175
Pre-levels and post-levels	176
Deduplication automation action	177
Lesson 5 Temporal triggers	178
Temporal trigger	179
Temporal trigger overview	180
Temporal trigger example: mail_on_critical	181
Event correlation: GenericClear	182
GenericClear automation	183
GenericClear	184

Lesson 6 Signal triggers	185
Signal trigger	186
System signals	187
User-defined signals	188
Signal trigger	189
Raising and dropping signals	190
Lesson 7 ObjectServer procedures	191
ObjectServer procedures	192
SQL procedure syntax	193
Example procedure	194
Parameter declaration	195
Parameter declaration: IN mode	196
Parameter declaration: OUT mode	197
Parameter declaration: IN OUT mode and type	198
Variable declaration	199
External procedures	200
Example External procedures	201
Running a procedure	202
Student exercises	203
Summary	204
5 Web GUI administration.....	205
Objectives	206
Lesson 1 Filters, views, and event grouping	207
Displaying event data	208
Creating a view: Display columns	209
Creating a view: Display columns, continued	211
Creating a view: Sort columns	212
Creating a view: Group columns	213
Creating a filter	214
Creating a filter: Filter condition	215
Using a filter and view	217
Event grouping	218
Lesson 2 Tools, prompts, and menus	219
Web GUI tools	220
Creating a Web GUI tool	221
Command tool example	222
SQL tool example	223
SQL tool journal entry	224
CGI tool example	225
CGI command registration	226
Limiting tool access by group	227
Limiting tool access by class	228
Creating a tool prompt	229
Prompt types	230
Using a prompt in a tool	232
Adding a tool to a menu	233
Lesson 3 Maps	234

Visualizing high-level event information	235
Maps: General information	236
Sample map	237
Creating a map	238
Icon properties	239
Icon associations	240
Icon associations, continued	241
Testing the map	242
Adding a map to a page	243
Adding a map to a page, continued	244
Lesson 4 Gauges	245
Gauges	246
Characteristics of a gauge	247
Characteristics of a gauge, continued	248
Creating a metric	249
Adding a gauge to a page	250
Adding a gauge to a page, continued	251
Lesson 5 Basic dashboard creation	252
Dashboard defined	253
Sample page	254
Creating a page	255
Creating a page, continued	256
Adding a widget to a page	257
Widget properties	258
Adding another widget to a page	259
Accessing the page	260
Lesson 6 Web GUI administrative API	261
Using the WAAPI client	262
Installing the WAAPI client on a remote host	263
Using WAAPI instructions to administer the server	264
Starting the WAAPI client	265
Sample output from webtop_report.xml	266
Student exercises	267
Summary	268
6 User administration	269
Objectives	270
Lesson 1 User administration overview	271
Overview	272
ObjectServer user administration	273
Netcool/OMNIbus Administrator utility	274
Web GUI user administration	275
ObjectServer user synchronization	276
ObjectServer group synchronization	277
Web GUI roles	278
Web GUI group roles	279
LDAP as an authentication source	280
Lesson 2 ObjectServer user administration	281

ObjectServer users	282
User configuration requirements	283
Restriction filters	284
Roles	285
Groups	286
Creating a group	287
Adding new users	288
Configuring the ObjectServer to use LDAP for external authentication	289
Lesson 3 Web GUI user administration	290
Web GUI user administration	291
ObjectServer user repository	292
LDAP user repository	293
Synchronizing LDAP users with the ObjectServer	294
Synchronizing LDAP users with the ObjectServer, continued	295
Dashboard Application Services Hub administrative authority	296
Creating groups and users	297
Creating a group	298
Creating a user	299
Assigning roles to a group	300
Read-only versus read/write access	301
User preferences	302
User preferences explained 1 of 3	303
User preferences explained 2 of 3	304
User preferences explained 3 of 3	305
Lesson 4 Creating Web GUI startup pages	306
Overview	307
Creating a role	308
Assigning role to page	309
Creating a view 1 of 2	310
Creating a view 2 of 2	311
Creating a console preference profile 1 of 2	312
Creating a console preference profile 2 of 2	313
Assigning role to user	314
Result	315
Student exercises	316
Summary	317
7 Customizing Tivoli Common Reporting reports	318
Objectives	319
Lesson 1 Overview	320
Historical event reporting	321
Components	322
REPORTER database configuration	323
REPORTER database configuration, continued	324
AUDIT tables	325
Text conversion tables	327
Modifying the event record	328
Lesson 2 Framework Manager	329

Definition of a data model	330
Why model your data?	331
Framework Manager	332
Installing Framework Manager	333
Configuring Framework Manager	334
Framework Manager user interface	335
Framework Manager terminology	336
Framework Manager terminology, continued	337
Facts and dimensions	338
Identifiers and attributes	339
Framework Manager workflow	340
Lesson 3 Modifying the data model	341
Requirements	342
Accessing report archive database	343
Report package file	344
Copying the original model	345
Modifying the copy, Database View	346
Modifying the copy, Consolidation View	347
Modifying the copy, Testing the changes	348
Creating updated package	349
Publishing updated package	350
Student exercises	351
Summary	352
8 Web GUI high availability.....	353
Objectives	354
Lesson 1 Dashboard Application Services Hub	355
Load balancing environment	356
Load balancing requirements	357
Synchronized data	358
Normal operation	359
Lesson 2 Web GUI	360
Web GUI in a load balancing environment	361
Configuration data	362
Configuration data examples	363
Configuration data changes	364
Detecting configuration data changes	365
Lesson 3 Setting up a load balanced cluster	366
Steps to create a Dashboard Application Services Hub cluster	367
Creating the DB2 database	368
High availability properties	369
Creating the cluster	370
Adding more nodes to the cluster	371
Configuring server-to-server trust, 1 of 3	372
Configuring server-to-server trust, 2 of 3	373
Configuring server-to-server trust, 3 of 3	374
Configuring Web GUI load balancing	375
IBM HTTP Server	376

Lesson 4 Cluster administration	377
Cluster administration for WebSphere	378
Cluster administration for Web GUI	379
Web GUI load balancing best practices	380
Student exercises	381
Summary	382
9 Security	383
Objectives	384
Lesson 1 Netcool/OMNibus security elements	385
Main components	386
User authentication	387
Secure mode authentication	388
Configuring ObjectServer user authentication	389
User authorization	390
Auditing authentication	391
Property value encryption	392
Communication security	393
SSL concepts: Certificates	394
SSL concepts: Keystores	395
Lesson 2 Using SSL for client and server communications	396
Steps for configuring SSL	397
Modifying the interfaces file	398
Configuring cryptographic properties	399
Configuring cryptographic properties, continued	400
Steps for creating, and distributing keys, and certificates	401
Key management database	402
Creating a key management database in FIPS 140-2 mode	403
Creating self-signed certificates	404
Distributing self-signed certificates	405
Creating certificate requests	406
Signing certificate requests	407
Receiving signed certificates	408
Configuring probes for SSL	409
Configuring bidirectional ObjectServer gateways for SSL	410
Configuring JDBC gateways for SSL	411
Lesson 3 Encryption and FIPS compliance	412
ObjectServer configuration for FIPS 140-2	413
Property value encryption	414
Encrypting a string value	415
ObjectServer properties	416
Probe properties	417
Bidirectional ObjectServer gateways	418
Unidirectional ObjectServer gateways	419
Student exercises	420
Summary	421
10 Multitiered architecture	422
Objectives	423

Lesson 1 Overview	424
Multitiered architecture	425
Standard multitiered architecture	426
Collection layer	427
Aggregation layer	428
Display layer	429
Lesson 2 Deploying the architecture	430
Multitiered architecture and configuration	431
Naming conventions	432
Multitiered configuration file locations	433
Multitiered configuration file locations, continued	434
Steps to deploy the architecture manually	435
Steps to deploy the architecture automatically	436
Student exercises	437
Summary	438



About this course



IBM Tivoli Netcool/OMNIbus V8.1 Administration and Maintenance



© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This course focuses on advanced administration of the IBM® Tivoli® Netcool®/OMNIbus solution.

The lab exercises are based on a distributed deployment of Netcool/OMNIbus. You start with two images with Netcool/OMNIbus installed and deployed in a high availability configuration. You use this environment to explore advanced administration topics, and deploy advanced features.

The lab environment for this course uses the 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
Course level	ERC 2.0
This course is a new course.	

Details	
Product and version	Product Name V8.1
Duration	4 days
Skill level	Advanced / Expert

About the student

This course is designed for product administrators and other advanced users.

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

- Linux operating system skills are required
- Basic SQL knowledge is required
- Knowledge of LDAP is beneficial

Before taking this course make sure that you have taken the following courses

- IBM Tivoli Netcool/OMNIbus 8.1 User
- IBM Tivoli Netcool/OMNIbus 8.1 Installation and Configuration

Learning objectives

Course objectives

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

- Describe the major functions of the Netcool/OMNIbus Administrator utility
- Use both basic and advanced SQL
- Configure and run probe integrations
- Configure and use advance probe functions
- Configure and implement Web GUI load balancing
- Customize Tivoli Common Reporting portal reports
- Configure core components and standard integration for SSL and FIPS 140-2 compliance
- Implement and examine the ESF-based, multi-tiered architecture

Course agenda

The course contains the following units:

1. [Introduction to Netcool/OMNIbus Administration](#)

In this unit you learn about some of the basic administrative tasks common to all Netcool/OMNIbus components. In this unit, you learn how to perform some of the basic administrative functions.

2. [ObjectServer administration](#)

In this unit, you learn about the structure of the ObjectServer, and how to use the administrator utility to modify the ObjectServer.

3. [Probes](#)

In this unit, you learn about some of the advanced features for probes.

4. [Automations](#)

In this unit you learn about ObjectServer SQL, and how SQL is used in automations.

5. [Web GUI administration](#)

In this unit, you learn basic Web GUI administration. You learn how to create event filters and views. You learn how to create desktop tools and add them to menus. You learn how to create a map and place the map on a page. You learn how to create a gauge and add the gauge to a page. You also learn how to use the Web GUI administrative API to perform basic administration from the command line.

6. [User administration](#)

In this unit, you learn how to create users and groups.

In this unit, you learn how to configure user access to event information. You configure a user for access to event records, add a restriction to limit the events, and then configure a default startup page for the user.

7. [Customizing Tivoli Common Reporting reports](#)

In this unit, you learn how to modify the Cognos® data model to accommodate extra columns in the Netcool/OMNIbus event record.

The goal of these exercises is to introduce you to using the IBM Cognos® Business Intelligence Modeling tool, Framework Manager. You use Framework Manager to extend an existing Cognos data model for use in Tivoli Common Reporting. In this lab session, you modify the Tivoli Netcool/OMNIbus data model. The model modification supports access to more columns in the Netcool/OMNIbus REPORTER database for use in Tivoli Common Reporting and Cognos reports.

8. [Web GUI high availability](#)

In this unit, you learn how to create a WebSphere cluster, and implement Web GUI high availability.

9. Security

In this unit, you learn about the features of Netcool/OMNibus related to security.

In these exercises, you change the ObjectServer encryption method to use AES instead of DES. You configure the ObjectServer to run in FIPS 140-2 mode. You implement Secure Socket Layer communications.

10. Multitiered architecture

In this unit, you learn how to create multitier architectures.

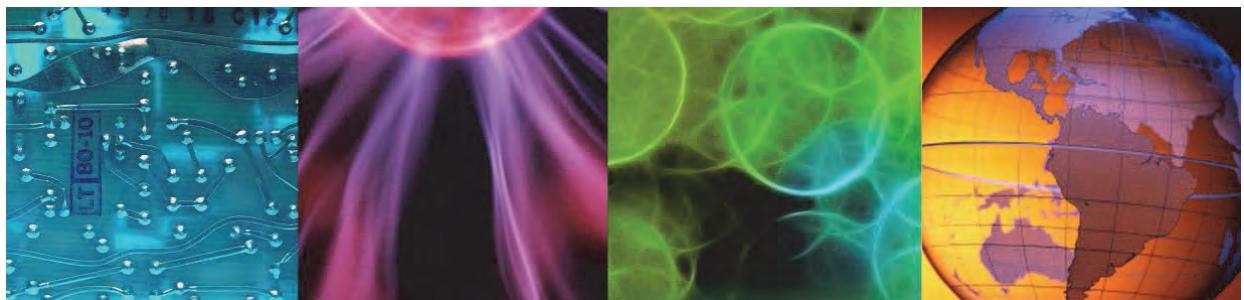
In this unit, you learn how to use the Initial Configuration Wizard to create, and deploy a multi-tiered architecture.



1 Introduction to Netcool/OMNIbus Administration



Introduction to Netcool/OMNIbus Administration



© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit you learn about some of the basic administrative tasks common to all Netcool/OMNIbus components.

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

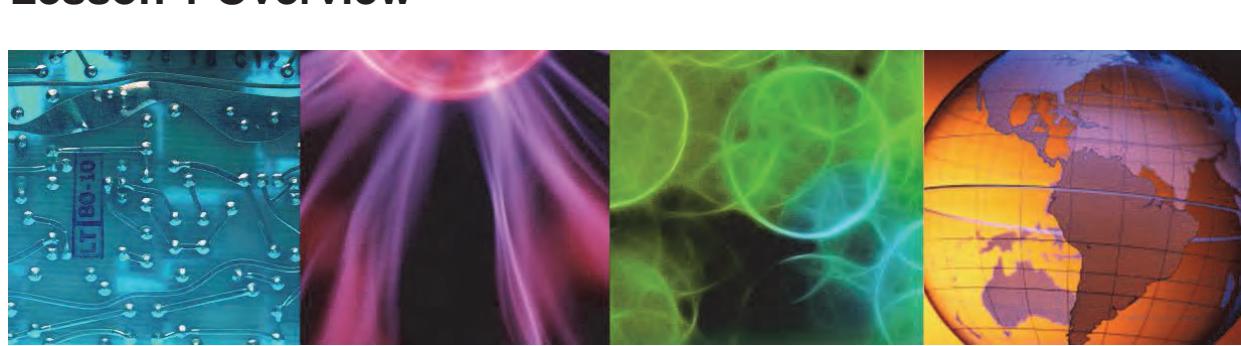
Objectives

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

- Describe basic administration tasks
- Describe how to start and stop Netcool/OMNIbus components
- Describe how to back up and restore Netcool/OMNIbus components
- Describe how behavior is configured within Netcool/OMNIbus components
- Describe how to locate, download, and apply maintenance packages



Lesson 1 Overview



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Describe the Netcool/OMNibus software components
- Describe what administrative functions are relevant for each software component

Administration categories

Starting and stopping

Backing up and restoring

Applying maintenance

Configuring behavior

User administration

Installing and upgrading

- Covered in IBM Tivoli Netcool/OMNIbus 8.1 Installation and Configuration

Administration categories

This slide shows a list of some of the basic administrative functions that are common to all software components.

The material in this class does not cover installation and upgrade. Installation and upgrade are covered in the prerequisite class IBM Tivoli Netcool/OMNIbus 8.1 Installation and Configuration.

Tivoli Netcool/OMNibus software components

ObjectServer

Event Repository

Desktop

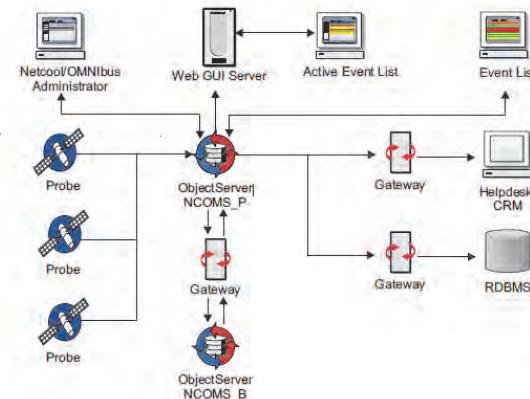
- Event List
- Tools

Web GUI and Dashboard Application Services Hub

Browser-based client

Probes

Gateways



© Copyright IBM Corporation 2015

5

Tivoli Netcool/OMNibus software components

Netcool/OMNibus components are categorized as core or Web GUI. The core components consist of ObjectServer, native desktop, probes, and gateways. The Web GUI component includes Dashboard Application Services Hub.

Administration by component

Component	Start / stop	Backup / restore	Maintenance	Behavior	User administration
ObjectServer	Y	Y	Y	Y	Y
Desktop	N (1)	Y	Y	Y (2)	Y
Probe	Y	Y	Y	Y	N (5)
Gateway	Y	Y	Y	Y	N (5)
Web GUI	N (3)	Y	Y	Y	Y
Dashboard Application Services Hub	N (4)	Y	Y	Y	Y
WebSphere	Y	Y	Y	Y	Y

1 Administrator does not start or stop desktops

2 Administrator controls some aspects of desktop behavior

3 Web GUI starts and stops with WebSphere

4 Dashboard Application Services Hub starts and stops with WebSphere

5 Unless the ObjectServer runs in secure mode, a user is not required for a probe or gateway

© Copyright IBM Corporation 2015

6

Administration by component

The table on this slide identifies the administrative categories that are relevant to each software component. A Netcool/OMNIbus administrator typically performs the functions that are shown here. Some administrative functions are not performed for some components.

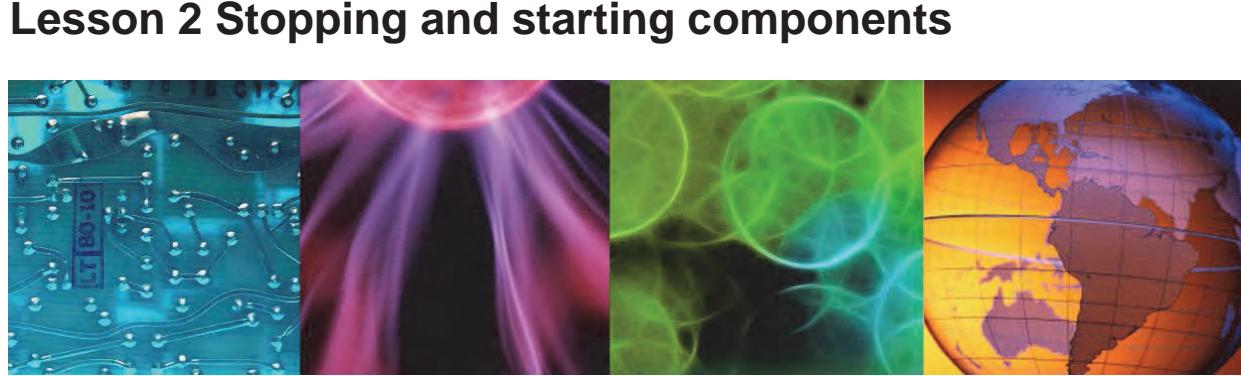
The native desktop is installed on user workstations. The administrator does not stop or start native desktops. In addition, the administrator has little control over the behavior of the desktop. The user typically configures desktop behavior.

Probes and gateways typically do not require a user and password. If the target ObjectServer is configured to run in *secure mode*, probes and gateways must provide a user and password.

Web GUI is another component with varied administrative functions. Web GUI is contained within Dashboard Application Services Hub. Dashboard Application Services Hub is contained within WebSphere®. Therefore, the administrator does not start Web GUI or Dashboard Application Services Hub. Instead, the administrator starts WebSphere.



Lesson 2 Stopping and starting components



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Stop and start the ObjectServer
- Stop and start probes and gateways
- Stop and start Dashboard Application Services Hub

Stopping and starting the ObjectServer

Process Activity

- Suggested best practice for controlling all Netcool/OMNIbus core components
- Process Activity starts when the server starts, and starts the ObjectServer
- Process Activity stops when the server is shut down, and stops the ObjectServer

Starting from the command line

- UNIX or Linux
\$NCHOME/omnibus/bin/nco_objserv -name <ObjectServer name>
- Microsoft Windows
\$NCHOME\omnibus\bin\nco_objserv -name <ObjectServer name>

Stopping from the command line

- Connect to the ObjectServer with the command-line utility
 - UNIX or Linux
\$NCHOME/omnibus/bin/nco_sql -server <ObjectServer name>
 - Microsoft Windows
\$NCHOME\omnibus\bin\isql -S <ObjectServer name>
- Enter SQL commands to gracefully shut down
 - 1> ALTER SYSTEM SHUTDOWN;
 - 2> go

© Copyright IBM Corporation 2016

8

Stopping and starting the ObjectServer

Process activity is the recommended best practice for controlling Netcool/OMNIbus core components. Process activity starts when the server is restarted. Process activity then starts all managed components.

If process activity does not manage the ObjectServer, you can stop it with a command-line utility. You connect to the ObjectServer and then enter the SQL commands to stop the ObjectServer.

Stopping and starting the ObjectServer with process activity

Determine the ObjectServer process name, for example:

```
nco_pa_status -server HOST1_PA -user netcool -password object00
```

Service Name	Process Name	Hostname	User	Status	PID
Core	MasterObjectServer	host1.tivoli.edunetcool		RUNNING	1990
	SyslogProbe	host1.tivoli.edunetcool		RUNNING	1991
	SnmpProbe	host1.tivoli.edunetcool		RUNNING	1992

Stop the process

```
nco_pa_stop -server HOST1_PA -user netcool -password object00 -process MasterObjectServer
```

```
nco_pa_status -server HOST1_PA -user netcool -password object00
```

Service Name	Process Name	Hostname	User	Status	PID
Core	MasterObjectServer	host1.tivoli.edunetcool		DEAD	0
	SyslogProbe	host1.tivoli.edunetcool		RUNNING	1991
	SnmpProbe	host1.tivoli.edunetcool		RUNNING	1992

Start the process

```
nco_pa_start -server HOST1_PA -user netcool -password object00 -process MasterObjectServer
```

© Copyright IBM Corporation 2016

9

Stopping and starting the ObjectServer with process activity

If the ObjectServer is managed with process activity, you determine the process name that is assigned to the ObjectServer component. You use the nco_pa_status command to list the managed processes, their process names, and their current state. Then, you can use the nco_pa_stop command to stop the specific process. You also use the process name to start the process with nco_pa_start.

Visual process activity

Visual Process Activity is a GUI to configure and control process agents and processes

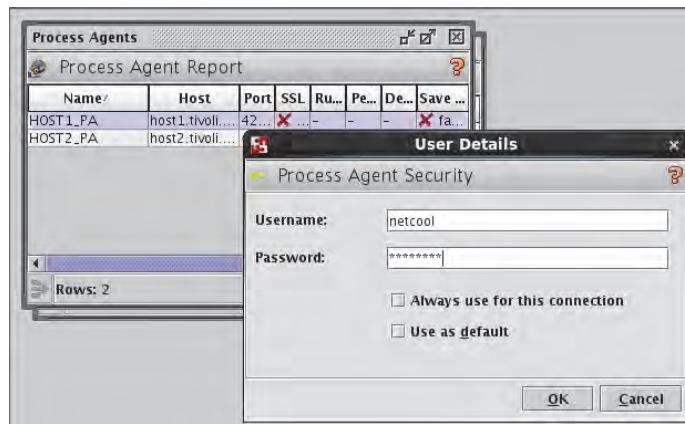
Start from the Netcool/OMNIbus

Administrator

Process Activity must be running

Log in with system user name and

password

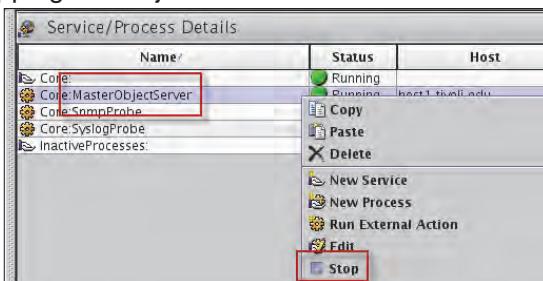


Visual process activity

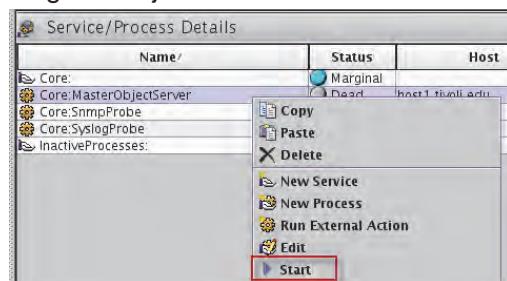
The Visual Process Activity component provides a graphical interface to the process activity daemon. You access visual process activity from the Netcool/OMNIbus Administrator utility.

Stopping and starting the ObjectServer with visual process activity

Stopping the ObjectServer



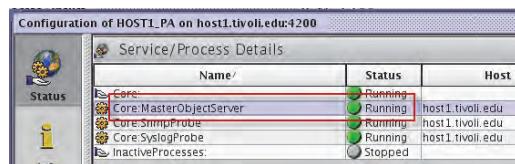
Starting the ObjectServer



Stopped



Started



© Copyright IBM Corporation 2015

11

Stopping and starting the ObjectServer with visual process activity

This slide shows how to stop and start a process with visual process activity. You right-click a process name and select stop or start.

Stopping and starting the ObjectServer from the command line

Stopping the ObjectServer with the SQL command-line utility

1. Connect to the running ObjectServer

```
nco_sql -server NYC_AGG_P -user root -password object00
```

-The command requires an ObjectServer user ID and password

2. Enter the SQL commands to shut down the ObjectServer

```
1> ALTER SYSTEM SHUTDOWN;
```

```
2> go
```

The ObjectServer comes down gracefully

Starting the ObjectServer from the command line

```
nco_objserv -name NYC_AGG_P &
```

Important:

If process activity is configured to control the ObjectServer, you cannot stop the ObjectServer from the command line.

If you stop the ObjectServer, process activity restarts it automatically.

© Copyright IBM Corporation 2016

12

Stopping and starting the ObjectServer from the command line

The ObjectServer is started with the following command:

```
$OMNIHOME/bin/nco_objserv -name <ObjectServer name> &
```

You can stop an ObjectServer with the following commands:

```
$OMNIHOME/bin/nco_sql -server <ObjectServer name> -user <ObjectServer user>
-pw <ObjectServer password>
1> ALTER SYSTEM SHUTDOWN;
2> go
```

This command sequence causes the ObjectServer to shut down gracefully.

You cannot stop and start the ObjectServer from the command line if the ObjectServer is managed with process activity.

Controlled ObjectServer shutdown

Stopping the ObjectServer with the SQL command-line utility

- Causes the ObjectServer to shut down immediately
- Probes and gateways might have events that are buffered for the ObjectServer
- You can configure a controlled shutdown of any ObjectServer such that pending changes are forwarded to IDUC clients before the ObjectServer shuts down

Controlled ObjectServer shutdown

- Implemented as automations and procedures in target ObjectServer
- When the procedure is executed to shut down the ObjectServer
 - The ObjectServer is brought to a restricted state
 - Connections that are identified for non-IDUC clients (such as **nco_sql** and **nco_config**) are dropped
 - An IDUC FLUSH command is initiated to send pending changes to all identified IDUC clients (such as gateways and event lists)
 - Any new connection requests to the ObjectServer are blocked
- When the data retrieval is completed for the IDUC clients, the **nco_pa_stop** utility is used to shut down the ObjectServer process that is running under process control

© Copyright IBM Corporation 2016

13

Controlled ObjectServer shutdown

When the ObjectServer is shut down with the ALTER SYSTEM SHUTDOWN command, the ObjectServer comes down immediately. Client connections are terminated and the ObjectServer creates a checkpoint file. In many cases, IDUC updates are pending for one or more clients. These updates are lost.

You can optionally configure controlled ObjectServer shutdown. The behavior is configured with an SQL file. The file is included with Netcool/OMNIbus in the following location:

```
$OMNIHOME/extensions/control_shutdown/control_shutdown.sql
```

You import the file into an ObjectServer with the **nco_sql** utility. The file creates some triggers and procedures in the ObjectServer. The behavior depends on process activity. Process activity must manage the ObjectServer. Process activity must be configured to run *external* procedures.

You modify the SQL file before it is imported into the ObjectServer. You configure various properties that are related to process activity.

After the SQL file is modified and imported, you shut down the ObjectServer as follows:

```
$OMNIHOME/bin/nco_sql -server <ObjectServer name> -user <ObjectServer user>
-pwd <ObjectServer password>
1> execute procedure control_shutdown;
2> go
```

Starting and stopping a probe

Process Activity

- Suggested best practice for controlling probes
- Process Activity starts when the server starts, and starts the probe
- Process Activity stops when the server is shut down, and stops the probe

Starting from the command line

- UNIX or Linux
\$NCHOME/omnibus/probes/nco_p_probename -server <ObjectServer name> &
- Microsoft Windows
\$NCHOME\omnibus\probes\ncop_p_probename -server <ObjectServer name>

Stopping from the command line

- There is no mechanism to stop a probe
- The only option is to kill the process

Starting and stopping a probe

A probe is stopped or started with process activity just like the ObjectServer. You can use the command-line utilities or visual process activity.

You start a probe from the command line as follows:

```
$OMNIHOME/probes/nco_p_<probe name> -server <ObjectServer name> &
```

There is no option for stopping a probe from the command line. The only option is to kill the process.

Starting and stopping a gateway

Process Activity

- Suggested best practice for controlling gateways
- Process Activity starts when the server starts, and starts the gateway
- Process Activity stops when the server is shut down, and stops the gateway

Starting from the command line

- UNIX or Linux
`$NCHOME/omnibus/bin/nco_g_gateway &`
- Microsoft Windows
`$NCHOME\omnibus\bin\nco_g_gateway`

Stopping from the command line

- There is no mechanism to stop a probe
- The only option is to kill the process

© Copyright IBM Corporation 2016

15

Starting and stopping a gateway

Gateways are managed just like probes. They can be stopped and started with process activity.

You start a gateway from the command line as follows:

```
$OMNIHOME/bin/nco_g_<gateway name> &
```

There is no option for stopping a gateway from the command line. The only option is to kill the process.

Starting and stopping WebSphere

Web GUI is not controlled separately

- Web GUI starts and stops with WebSphere

Dashboard Application Services Hub is not controlled separately

- Dashboard Application Services Hub starts and stops with WebSphere

Process Activity is not used to control WebSphere

A custom script can be used on UNIX or Linux to start and stop WebSphere with the server

- The student exercises include a sample script

Starting from the command line

- UNIX or Linux

```
$JazzSM_HOME/profile/bin/startServer.sh server1
```

- Microsoft Windows

```
$JazzSM_HOME\profile\bin\startServer.bat -name server1
```

Stopping from the command line

- UNIX or Linux

```
$JazzSM_HOME/profile/bin/stopServer.sh server1 -username <Admin user> -password <Admin password>
```

- Microsoft Windows

```
$JazzSM_HOME\profile\bin\stopServer.bat server1 -username <Admin user> -password <Admin password>
```

© Copyright IBM Corporation 2016

16

Starting and stopping WebSphere

Web GUI is a collection of applications that are accessed from Dashboard Application Services Hub. The web engine for Dashboard Application Services Hub is WebSphere. When you stop or start Web GUI or Dashboard Application Services Hub, you stop and start WebSphere.

WebSphere is not controlled with process activity. It must be started and stopped from the command line. You manage WebSphere with three scripts. One starts, one stops, and one is used to query the status.

You start WebSphere as follows:

```
JazzSM_HOME/profile/bin/startServer.sh server1
```

You stop WebSphere as follows:

```
JazzSM_HOME/profile/bin/stopServer.sh server1 -user <admin user> -password  
<admin password>
```

You verify the status of WebSphere as follows:

```
JazzSM_HOME/profile/bin/serverStatus.sh server1 -user <admin user> -password  
<admin password>
```

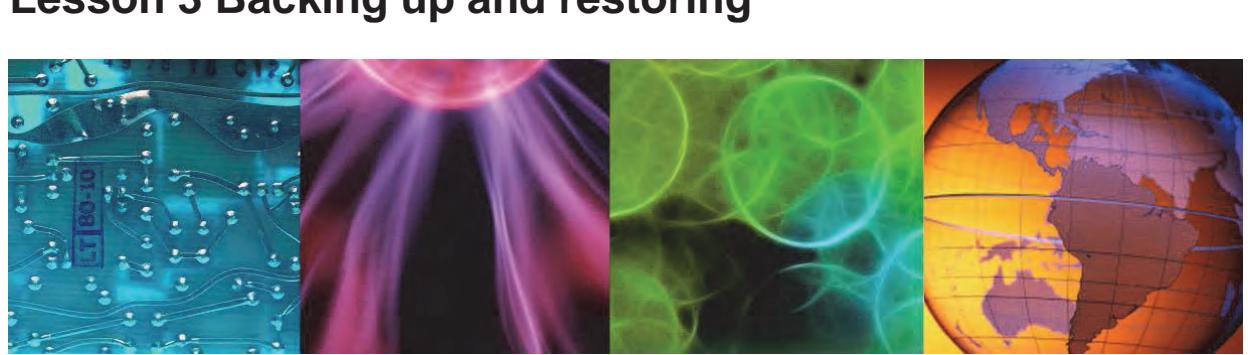
The WebSphere administrator user name and password are required to stop or status WebSphere.



Hint: JazzSM_HOME is typically /opt/IBM/JazzSM.



Lesson 3 Backing up and restoring



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Create application-level backups for components
- Use the backups to restore components

Backup requirements

Disk-level backups

- Full or complete backup
 - Generally used for recovery after a catastrophic failure
 - Typically done weekly
- Partial or incremental backup
 - Used to capture all changes between full backups
 - Used with a full backup when recovering from a catastrophic failure
 - Generally done on a more frequent basis, possibly daily
- Done with special purpose software, such as IBM Storage Manager

Application-level backups

- Manually copy the contents of a directory or file-system
- Use application-specific utility to export configurations and data

The following slides focus on application-level backups

Backup requirements

Disk backups are a standard process in all data centers. They are required to prevent the loss of critical data due to a hardware failure, or accidental deletion. Many companies perform full disk backups on a standard schedule. Full backups might occur weekly. Between full backups, companies perform incremental backups. An incremental backup captures only changed data. Incremental backups might occur daily.

Application-level backups are typically focused on just one application and not an entire disk. You create an application level backup because a disk backup might not capture everything that is required to restore the application.

ObjectServer backup and recovery considerations

Two aspects that are related to the ObjectServer

- Structure
 - Databases and tables
 - Column definitions
 - Others
- Contents
 - Event data
 - Menus, tools, automations
 - Others

Structure remains static unless modified by an administrator

- Structure is preserved in the database checkpoint files

Contents are changing continuously

- Contents are preserved in the database checkpoint files
- However, the checkpoint represents a point in time

Restoring the contents is the biggest challenge

© Copyright IBM Corporation 2016

19

ObjectServer backup and recovery considerations

The ObjectServer is a database. The database has a structure and it contains data. The structure of the ObjectServer is relatively static. The only time the structure changes is when a modification is made by an administrator. However, the contents of the ObjectServer change continuously. Preserving the contents and restoring them without any loss is the biggest challenge.

ObjectServer backups

Key directories

- Netcool/OMNIbus core files

\$NCHOME – typically /opt/IBM/tivoli/netcool

- IBM Installation Manager

<home directory>/IBM – the home directory of the user that installed Netcool/OMNIbus

Export/Import utilities

- ObjectServer configuration, but no event records

\$OMNIHOME/bin/nco_confpack

- ObjectServer configuration and event records

\$OMNIHOME/bin/nco_osreport

ObjectServer backups

All Netcool/OMNIbus core software resides in a single directory:

/opt/IBM/tivoli/netcool

This directory holds the executable files, configuration files, log files, and the checkpoint files for the ObjectServer. Create a backup copy of this directory.

IBM Installation Manager is used to install the software components and for applying maintenance. The IBM Installation Manager software resides in the home directory of the user who installs Netcool/OMNIbus. Create a backup of this directory.

The backups can be used to recover from a catastrophic hardware failure. In addition to these backups, you can create an export of the ObjectServer. The ObjectServer export is smaller than the backup of the directory and more portable.

There are two utilities you use to create an export of the ObjectServer. One utility, nco_confpack, creates an export with only the structure of the ObjectServer. The second utility, nco_osreport, creates an export with the complete contents of the ObjectServer.

Configuration Packager utility

Creates and manages configuration packages for transferring configuration objects between ObjectServers

- Deploy duplicate ObjectServers
- Back up ObjectServer configurations
- Does not export event data

Export the configuration of the existing ObjectServer

```
$OMNIHOME/bin/nco_confpack -export -server <ObjectServer name> -package /tmp/OS1.jar
```

Steps to recreate the ObjectServer:

1. Create an ObjectServer

```
$OMNIHOME/bin/nco_dbinit -name <new ObjectServer>
```

2. Start the ObjectServer

```
$OMNIHOME/bin/nco_objserv -server <new ObjectServer>
```

3. Import the configuration into the new ObjectServer

```
$OMNIHOME/bin/nco_confpack -import -server <ObjectServer name> -package /tmp/OS1.jar
```

The event data must still be recovered with some other technique

© Copyright IBM Corporation 2016

21

Configuration Packager utility

The configuration packager utility, nco_confpack, is used to extract the structure of the ObjectServer, and the contents of some tables. The utility extracts everything except the event data. The IBM Support desk might ask you to use this utility when you call with an error or issue. IBM Support asks you to export the contents of your ObjectServer. You upload the export to an IBM FTP site. The support engineer uses your export to create an ObjectServer. The ObjectServer contains the same structure as your ObjectServer. However, it does not have any event data.

The utility includes an option to selectively export the contents of an ObjectServer. You can use this option to configure the utility to not export things like user names and passwords.

ObjectServer Report Generator utility

Writes the contents of ObjectServer tables to an HTML or XML file

- Use the files to document the contents of the ObjectServer

The utility can also be used to export an ObjectServer configuration to a set of SQL files

- Use the SQL files to create the initial contents of a new ObjectServer
- The new ObjectServer contains the same structure and content, including event data

Export the configuration of the existing ObjectServer

```
$OMNIHOME/bin/nco_osreport -dbinit -server <ObjectServer name>
- The utility creates SQL files that are used to create the structure and content of a new ObjectServer
```

Create an ObjectServer

```
$OMNIHOME/bin/nco_dbinit -server <new ObjectServer> -applicationfile application.sql
-automationfile automation.sql -desktopfile desktop.sql -securityfile security.sql
-systemfile system.sql -alertsdata -alertsdatafile alertsdata.sql
```

- The new ObjectServer has the same structure as the old ObjectServer
- The new ObjectServer has the same content as the old ObjectServer including event data

However, even though the event data is recovered, it is still old

© Copyright IBM Corporation 2016

22

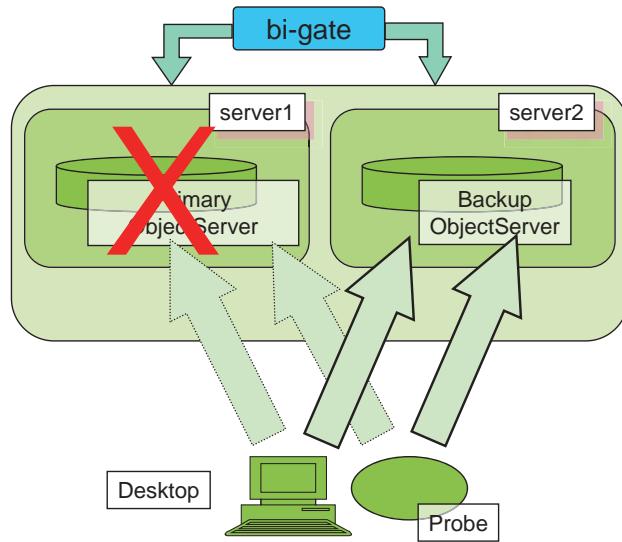
ObjectServer Report Generator utility

The ObjectServer report generator utility serves two purposes. The utility can be used to create an HTML file of the contents of the ObjectServer. The HTML file is a convenient technique for documenting the contents of the ObjectServer. The second purpose is to create an export of the complete ObjectServer, including the event data. The output from this option can be used to re-create the ObjectServer. The new ObjectServer is a duplicate of the original, including the event data. However, because the event data changes continuously in the original ObjectServer, the new ObjectServer does not contain the same event data.

Failure recovery with high availability

Assume server1 suffers a catastrophic disk failure

1. When the primary ObjectServer fails, the clients reconnect to the backup ObjectServer
 - The backup ObjectServer has the same content, and structure as the primary
2. Replace the failed disk hardware on server1
3. Restore the contents of the disk from backups
4. Restart the primary ObjectServer
 - The structure of the primary ObjectServer is restored
 - However, the contents are restored they are not the same as the backup ObjectServer
5. Bi-directional gateway synchronizes the contents of the ObjectServers automatically
6. Clients reconnect to the primary ObjectServer



© Copyright IBM Corporation 2016

23

Failure recovery with high availability

The recommendation for preserving the event data is with high availability. In a high availability configuration, the gateway replicates all event data between the primary and backup ObjectServers. If one of the systems sustains a catastrophic failure, the recovery is simple.

If a disk failure takes down the primary ObjectServer, all Netcool/OMNibus components connect to the backup ObjectServer. The backup ObjectServer contains the same event data as the primary due to the gateway.

You repair or replace the faulty hardware. You restore the disks, if necessary, from the full and incremental backup files. You restart the primary ObjectServer. The structure of the ObjectServer is restored based on the checkpoint file. The contents are also restored based on the checkpoint file. After the ObjectServer starts, the gateway synchronizes the contents of the primary and backup ObjectServers automatically. The primary ObjectServer now contains the same event data as the backup.

Desktop backups

Desktops are installed on end-user workstations

Desktops require Netcool/OMNIbus core files

Key directories

- Netcool/OMNIbus core files
\$NCHOME – typically /opt/IBM/tivoli/netcool
- IBM Installation Manager
<home directory>/IBM – the home directory of the user that installed Netcool/OMNIbus
- User-specific customizations
<home directory>/ .netcool
<home directory>/ .omnibus

Export/Import utilities

- None

Desktop backups

Native desktops are installed on user workstations. The software is stored in the same directory:

\$NCHOME

IBM Installation Manager is also used to install the native desktop.

The user-specific customizations for the native desktop are saved in the home directory of the user. If you create backups of the Netcool/OMNIbus software and the home directory of the user, you have everything that you need to restore the user environment.

Probe backups

Probes require Netcool/OMNibus core files

Key directories

- Netcool/OMNibus core files
\$NCHOME – typically /opt/IBM/tivoli/netcool
- IBM Installation Manager
<home directory>/IBM – the home directory of the user that installed Netcool/OMNibus

Export/Import utilities

- No specific utilities
- Probe behavior is configured through two text files
\$OMNIHOME/probes/<arch>/probename.props
\$OMNIHOME/probes/<arch>/probename.rules
- These files can be maintained with any version control software, like TortoiseSVN

© Copyright IBM Corporation 2016

25

Probe backups

Probes are typically installed on remote servers. The software is stored in the same directory as other Netcool/OMNibus components:

\$NCHOME

IBM Installation Manager is also used to install probes.

The configuration of the probe is controlled with two or more text files. The contents of these text files can be managed with any type of software version control system.

You create a backup of the Netcool/OMNibus directory and the IBM Installation Manager directory.

Gateway backups

Gateways require Netcool/OMNIbus core files

Key directories

- Netcool/OMNIbus core files
\$NCHOME – typically /opt/IBM/tivoli/netcool
- IBM Installation Manager
<home directory>/IBM – the home directory of the user that installed Netcool/OMNIbus

Export/Import utilities

- No specific utilities
- Gateway behavior is configured through various text files

\$OMNIHOME/etc/gateway_name.props
\$OMNIHOME/etc/gateway_name.map
\$OMNIHOME/etc/gateway_name.startup.cmd
\$OMNIHOME/etc/gateway_name.rdrwtr.tblrep.def

- These files can be maintained with any version control software, like TortoiseSVN

© Copyright IBM Corporation 2016

26

Gateway backups

Gateways are typically installed on remote servers. The software is stored in the same directory as other Netcool/OMNIbus components:

\$NCHOME

IBM Installation Manager is also used to install gateways.

The configuration of the gateway is controlled with various text files. The contents of these text files can be managed with any type of software version control system.

You create a backup of the Netcool/OMNIbus directory and the IBM Installation Manager directory.

Web GUI backups

Web GUI requires Dashboard Application Services Hub and WebSphere

Key directories

- Netcool/OMNIbus Web GUI files
`/opt/IBM/netcool/omnibus_webgui`
- Dashboard Application Services Hub files
`/opt/IBM/JazzSM`
- WebSphere files
`/opt/IBM/WebSphere`
- IBM Installation Manager
`<home directory>/IBM` – the home directory of the user that installed Netcool/OMNIbus

© Copyright IBM Corporation 2016

27

Web GUI backups

The software required by Web GUI is contained in three directories:

- WebSphere
`/opt/IBM/WebSphere`
- Dashboard Application Services Hub
`/opt/IBM/JazzSM`
- Web GUI
`/opt/IBM/netcool/omnibus_webgui`

IBM Installation Manager is used to install all three components.

You create backups for each application directory and for IBM Installation Manager.

There are other utilities to export the contents Dashboard Application Services Hub and Web GUI.

Web GUI backups, continued

Export/Import utilities

- Export parts of Web GUI configuration
 - Can be used to copy modifications from a test server to a production server
- Export the complete configuration
 - Referred to as cloning
 - Copies Web GUI data, and Dashboard Application Services Hub data
- The same utility is used for both
 - \$JazzSM_HOME/ui/bin/consolecli.sh – UNIX/Linux
 - \$JazzSM_HOME/ui/bin/consolecli.bat – Windows
- Using the utility
 - Features are controlled through command-line parameters
 - Export the source configuration to data.zip file
 - Copy the file to the target server
 - Import the configuration

© Copyright IBM Corporation 2016

28

Web GUI backups, continued

The consolecli utility is used to export the contents of Web GUI or Dashboard Application Services Hub. The utility is controlled with a property file. The contents of the property file determine whether the export is for Web GUI or Dashboard Application Services Hub. The utility can be used to export selected components or to export all components.

The contents of Dashboard Application Services Hub are fairly static. The contents change when an administrator implements a change such as creating a new page. A user requires a specific authority in order to change Dashboard Application Services Hub, so changes can be controlled. With a stable change control procedure in place, you control when a change takes place. When a change occurs, you can create an export to capture the revised configuration.

The contents of the Web GUI are also fairly static. However, most users can make changes. Therefore, changes to Web GUI cannot be controlled. You can implement an automated procedure to create an export at some standard frequency, like daily, to capture changes.

Web GUI recovery with load balancing cluster

In a cluster, a DB2 database holds the configuration data for the entire cluster

- The database is the master copy of the data shared by all the cluster nodes
- A single set of configuration data means that each node is configured identically
- No configuration data is specific to a cluster node

Although the database holds the master copy, each node also has a copy in its local file system

- The local copy is for fault tolerance reasons and allows the cluster to continue operation should the configuration database become unavailable during operation
- When a node starts, it reads a complete set of configuration data from the database into the local file system and loads it into memory to improve performance

Steps to recover from a catastrophic failure of one node

1. Recover the hardware
2. Restore disk from backups
3. Restart the WebSphere node

© Copyright IBM Corporation 2016

29

Web GUI recovery with load balancing cluster

There is a high availability option for the Web GUI. The high availability option for Web GUI is based on a WebSphere cluster. In a WebSphere cluster, there are two or more instances of WebSphere. The instances are configured to store their configuration data in a DB2® database. During normal operation, when a change is made to one member of the cluster, the change is written to the DB2 database. After the change is committed to the database, a notification is sent to the other members of the cluster. When a member receives the notification, the change is retrieved from the database and applied to the member. The cluster ensures that all members have the same contents. The information that is maintained across cluster members includes WebSphere, Dashboard Application Services Hub, and Web GUI content.

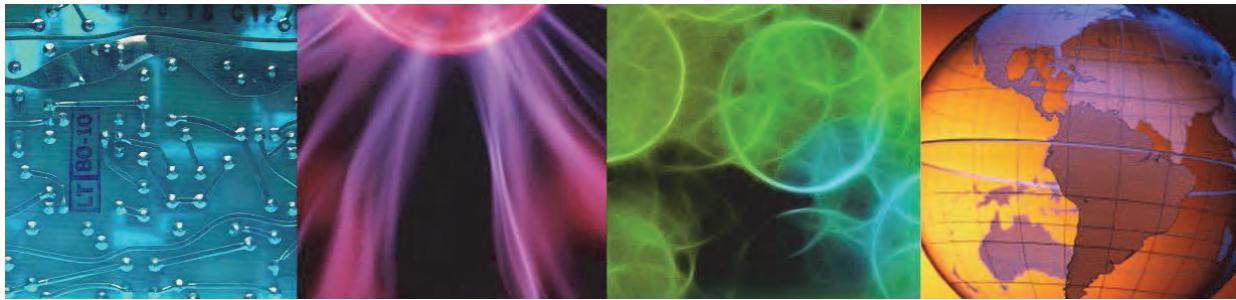
If Web GUI is operating in a clustered configuration, there is no need to maintain any other backups for the configuration data. You still need a backup in order to restore the software itself if there is a catastrophic failure. However, a separate backup for the configuration content is not required.



Lesson 4 Applying maintenance



Lesson 4 Applying maintenance



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Find maintenance packages
- Download maintenance packages
- Install maintenance packages

Overview

IBM distributes two types of maintenance packages

- Interim Fix
 - A tested and verified fix
 - It can contain fixes for one or more product defects (APARs)
 - If a fix is listed in a fix pack for the affected product, it is recommended that the fix pack is applied, rather than the interim fix
 - However, there can be cases where your situation calls for the installation of an individual interim fix
- Fix Pack
 - A cumulative collection of all fixes available to registered customers since the last release of the product
 - It can include fixes that were not previously released and can span multiple products or components
 - Usually, a fix pack can be applied on top of any earlier maintenance

Overview

IBM provides maintenance as either a fix pack or interim fix. An interim fix typically contains a few fixes. Interim fixes are typically released fairly regularly. A fix pack combines interim fixes, and, optionally, adds more fixes. The schedule of releases might look something like the following scenario.

When IBM releases a software version it is designated as *generally available*, or GA. The software is available for download from the Passport Advantage website. After the software is released, IBM releases an interim fix. The interim fix is available from a separate website that is called Fix Central. After some time, IBM releases another interim fix. The second interim fix typically contains the contents of the first interim fix, plus more fixes. This process repeats with subsequent interim fixes.

At some point, IBM combines a number of interim fixes into a fix pack. The fix pack contains all of the previous interim fixes, and possibly more fixes. The fix pack is also available from Fix Central.

Downloading maintenance packages

All maintenance packages are found on Fix Central

- Fix Central provides fixes and updates for your system's software
- Find fixes by searching by product, by fix ID, or by APAR
- Fix Central also helps identify any prerequisite or co-requisite fixes that are associated with the fixes you download
- A user ID and password are required

<http://www-933.ibm.com/support/fixcentral/>

Downloading maintenance packages

The Fix Central website is where all maintenance packages are found. You can search the contents of the website by product name, version, operating system, and others.

You need a valid IBM user ID and password to access the Fix Central website.

Searching for maintenance on Fix Central

The screenshot shows a search interface for Fix Central. At the top, there are two tabs: "Find product" and "Select product". Below them, a message says "Select the product below." and "When using the keyboard to navigate selection lists, use the left arrow key to move up and the right arrow key to move down, and the down arrow keys to navigate the list." A "Product Group*" dropdown menu is open, showing "Tivoli" as the selected item. A callout bubble points to it with the text "Select software pillar". Below it, another dropdown menu is open under "Select from Tivoli*", showing "IBM Tivoli Netcool OMNibus" as the selected item. A callout bubble points to it with the text "Select product". Further down, an "Installed Version*" dropdown menu is open, showing "8.1.0.0" as the selected item. A callout bubble points to it with the text "Select version". At the bottom of the form, there is a "Platform*" dropdown menu with "All" selected. A red arrow points from the "Continue" button at the bottom to the "Platform" dropdown. The "Continue" button is located at the bottom right of the form.

© Copyright IBM Corporation 2016

33

Searching for maintenance on Fix Central

To locate the available maintenance for Netcool/OMNibus, the easiest technique is to search by product. Drop-down lists guide you through the search process.

Searching for maintenance, continued

1-4 of 4 results

- 1. fix pack: [8.1.0-TIV-NCOMNibus-ZLinux-FP0001](#) →
8.1.0-TIV-NCOMNibus-ZLinux-FP0001
[More Information](#) [OMNIbus GUI](#)
- 2. fix pack: [8.1.0-TIV-NCOMNibus-Windows-FP0001](#) →
8.1.0-TIV-NCOMNibus-Windows-FP0001
[More Information](#) [OMNIbus GUI](#)
- 3. fix pack: [8.1.0-TIV-NCOMNibus-Linux-FP0001](#) →
8.1.0-TIV-NCOMNibus-Linux-FP0001
[More Information](#) [OMNIbus GUI](#)
- 4. fix pack: [8.1.0-TIV-NCOMNibus-AIX-FP0001](#) →
8.1.0-TIV-NCOMNibus-AIX-FP0001
[More Information](#) [OMNIbus GUI](#)

1-4 of 4 results

Click to download

Continue Clear selections Back

34

Searching for maintenance, continued

After you search by product, the list of available maintenance opens. The sample screen capture shows the list of fix packs available for Netcool/OMNIbus. In this example, the list contains a separate fix pack for each supported operating system.

To download a package, you select the file and click Continue. You specify the location to save the maintenance package. Maintenance packages are distributed in compressed form.

The file that is downloaded is referred to as an Electronic Software Distribution or ESD.

Installing a maintenance package

Maintenance packages are typically installed with IBM Installation Manager

- Varies by product and version
- Package contains a separate installation utility if IBM Installation Manager is not used

Each package contains a README file

- Describes what defects are corrected
- Describes how to apply the maintenance
- Generally describes how to roll-back or uninstall the maintenance if required
- Might also provide post-install instructions, like whether a restart is required

Two options for installing maintenance with IBM Installation Manager

- Connect to Passport Advantage and retrieve packages automatically
- Download packages manually as described previously

© Copyright IBM Corporation 2016

35

Installing a maintenance package

The process to install a maintenance package is the same whether for an interim fix or fix pack. You start by expanding the package file. The package contains a textual readme file. The readme file contains instructions for installing the package, and a list of the fixes. The file might also contain any postinstallation instructions if any are required.

The maintenance is installed with IBM Installation Manager. The package includes a utility script that is used to start the installation. The utility script starts IBM Installation Manager, and configures it to use the installation package as the software repository. In most cases, you run the utility script, which starts IBM Installation Manager. You click Next several times, and the maintenance is installed.

IBM Installation Manager provides several key features when installing maintenance packages. It ensures that all prerequisite and corequisite packages are installed. If the installation fails for some reason, IBM Installation Manager removes the maintenance and restores the software to the previous state. IBM Installation Manager also records what maintenance packages are installed and when they are installed.

You can also use IBM Installation Manager to retrieve a maintenance package and install it in one operation. If you elect to use this option, you must start IBM Installation Manager manually, and configure the fix central website as a remote user repository. The readme file describes how to use a remote repository.

Installing a maintenance package, continued

1. Extract the maintenance package into some temporary directory

The package is referred to as an Electronic Software Delivery (ESD) file

2. Start the installation utility

<maintenance directory>/update_gui.sh

The installation utility starts IBM Installation Manager

3. Select the package group



The maintenance is applied



© Copyright IBM Corporation 2016

36

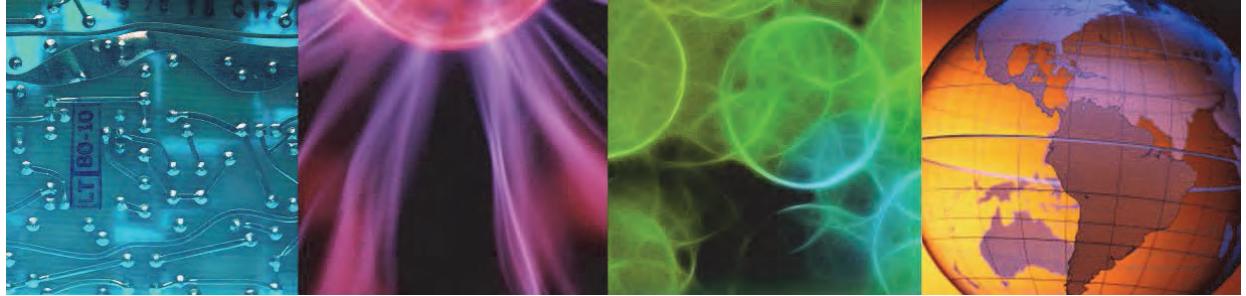
Installing a maintenance package, continued

Most maintenance packages contain three different installation scripts. One script is used to install the maintenance with a graphical interface. One script is used to install the maintenance with a textual interface. The last script is used to install the maintenance silently. The sample screen capture shows the graphical option.



Lesson 5 Modifying behavior

Lesson 5 Modifying behavior



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Change the basic behavior of the ObjectServer
- Change the basic behavior of probes and gateways
- Change the basic behavior of Web GUI

Overview

Most component behavior is controlled with parameters

- For example
 - Whether to create a log file or not
 - Where the log file is saved
 - Maximum size of the log file
 - Others

Parameters can be set on the command line

Parameters can be set in a property file

Overview

The term *behavior* as used in this unit refers to the basic operation of a component. All Netcool/OMNIbus components provide a basic set of behavior, for example, whether to create a log file, the name of the file, the size of the file, and the content of the file. Components also support advanced behaviors. Those behaviors are outside of the scope of this unit.

In most cases, basic behavior is controlled with a textual property file. The content of the file varies by component. The property file is modified with any type of text editor. For most components, a change to the property file requires a component restart to take effect.

Configuring ObjectServer behavior

ObjectServer property file

`$OMNIHOME/etc/ObjectServerName.props`

Manually change property settings

- Use any text editor
- Requires a restart of the ObjectServer to take effect

Change property settings with Administrator GUI

- Most changes take effect immediately
- Some parameter changes require a restart
- Changes that are made with the Administrator GUI are also written to the property file

Do not change any property unless you know exactly what it does

- Some property settings can potentially affect ObjectServer performance

© Copyright IBM Corporation 2016

39

Configuring ObjectServer behavior

There is a property file for the ObjectServer. It is found in the following location:

`$OMNIHOME/etc/<ObjectServer name>.props`

When you create an ObjectServer, the corresponding property file is also created. The property file contains a collection of parameters. Most of the parameters have default settings. You can modify the property file with a text editor, and change the setting for one or more parameters. You must restart the ObjectServer for the change to take effect.

The Netcool/OMNibus Administrator utility can also be used to modify some ObjectServer properties. You cannot modify all property values with the utility. Of the properties that can be modified, some take effect immediately, and others require an ObjectServer restart.

Configuring probe behavior

Some behavior is controlled through the property file

`$OMNIHOME/probes/<arch>/probename.props`

Other behavior is controlled through the rules file

`$OMNIHOME/probes/<arch>/probename.rules`

Manually change property settings

- Use any text editor
- Might require a restart of the probe to take effect
 - UNIX or Linux can use signal to cause the probe to reread property settings
 - Probe does not stop

Manually change rules file

- Use any text editor
- Might require a restart of the probe to take effect
 - UNIX or Linux can use signal to cause the probe to reread rules file
 - Probe does not stop

© Copyright IBM Corporation 2016

40

Configuring probe behavior

The behavior of a probe is configured in more than one textual file. The probe has a property file, just like other components. The probe also uses other textual files to implement more advanced behavior. The advanced behavior is outside the scope of this unit.

You modify a probe property file with a text editor. In some cases, you must restart the probe for the change to take effect. If the probe is running on a Linux or UNIX server, you can use a command-line option to cause the probe to reread the property file. When you use this option, the probe implements the change and does not stop.

The latest version of probes provides the option to configure the probe to support an HTTP interface. When the probe is configured to support HTTP, you can run a command-line utility to send a command to the probe through the HTTP interface. The command can change one or more probe properties. The HTTP interface is convenient when you run probes on remote servers. You can send a command to the remote probe, and change a property value without changing the property file.

Configuring gateway behavior

Some behavior is controlled through the property file

`$OMNIHOME/etc/gatewayname.props`

Other behavior is controlled through other files

`$OMNIHOME/etc/gatewayname.map`

`$OMNIHOME/etc/gatewayname.startup.cmd`

`$OMNIHOME/etc/gatewayname.rdrwtr.tblrep.def`

Manually change any file

- Use any text editor
- Requires a restart of the gateway to take effect

Configuring gateway behavior

The behavior of a gateway is configured in more than one textual file. The gateway has a property file, just like other components. The gateway also uses other textual files to implement more advanced behavior. The advanced behavior is outside the scope of this unit.

You modify a gateway property file with a text editor. You must restart the gateway for the change to take effect.

Configuring Web GUI behavior

Some behavior is controlled through the property file

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

Other behavior is controlled through the data source definition file

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

Manually change property file or data source definition file

- Use any text editor
- Requires a restart of Web GUI to take effect

Change data source definitions with Dashboard Application Services Hub

- Changes take effect immediately
- Changes are also written to the data source definition file

Configuring Web GUI behavior

The behavior of the Web GUI is configured in two text files. Most of the behavior is controlled with the `server.init` file. You modify the file with a text editor, and you must restart the Web GUI for the change to take effect.

Other behaviors are controlled with the *data source definition* file. The data source definition file identifies what ObjectServers are accessible by Web GUI. The file contains the access information for each ObjectServer. You can modify the file with a text editor, and you must restart Web GUI for the change to take effect.

Dashboard Application Services Hub can also be used to modify the data source definitions. You use the web-based graphical interface to add, change, or delete an ObjectServer definition. The change takes effect immediately, and does not require a Web GUI restart. In addition, the change is written to the data source definition text file on disk. Modifying the text file ensures that the change is preserved between Web GUI restarts.

Student exercises



Complete all exercises for this unit

Summary

You now should be able to perform the following tasks:

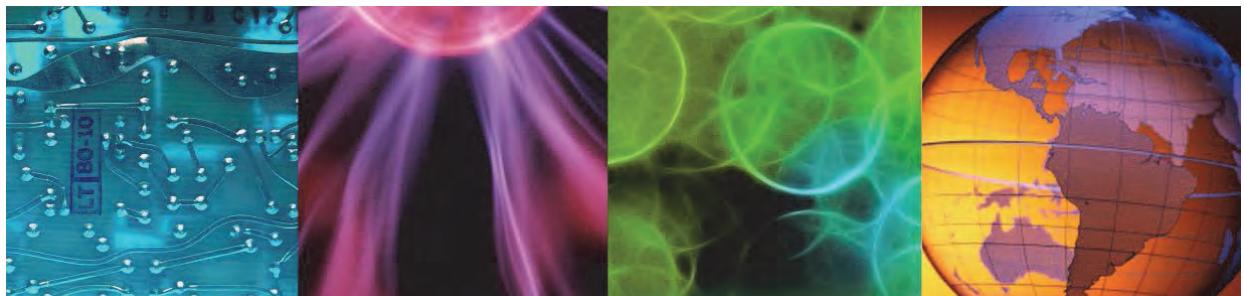
- Describe basic administration tasks
- Describe how to start and stop Netcool/OMNIbus components
- Describe how to back up and restore Netcool/OMNIbus components
- Describe how behavior is configured within Netcool/OMNIbus components
- Describe how to locate, download, and apply maintenance packages



2 ObjectServer administration



ObjectServer administration



© Copyright IBM Corporation 2016

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 of the ObjectServer, and how to use the administrator utility to modify the ObjectServer.

References: SC27-6265-00 *Netcool/OMNIbus Version 8 Release 1 Administration Guide*,
SC27-6505-00 *Web GUI Administration, and User's Guide*

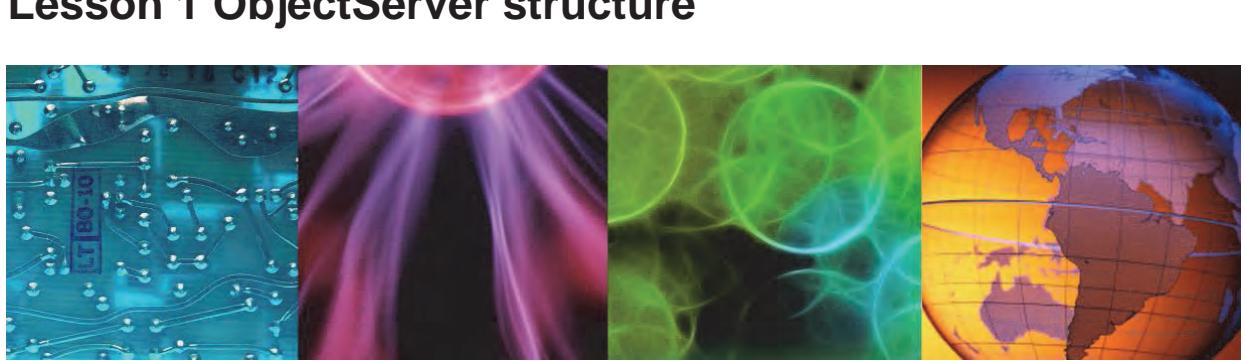
Objectives

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

- Describe the ObjectServer structure
- Describe important event record columns and their use
- Describe important ObjectServer properties
- Use the administration GUI to modify a property
- Use the administration GUI to modify an ObjectServer
- Create an ObjectServer



Lesson 1 ObjectServer structure



© Copyright IBM Corporation 2016

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 added by users
- 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
- tools: User tool and menu structure for native desktops
- transfer: Used by the ObjectServer gateways

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

Database and table restrictions

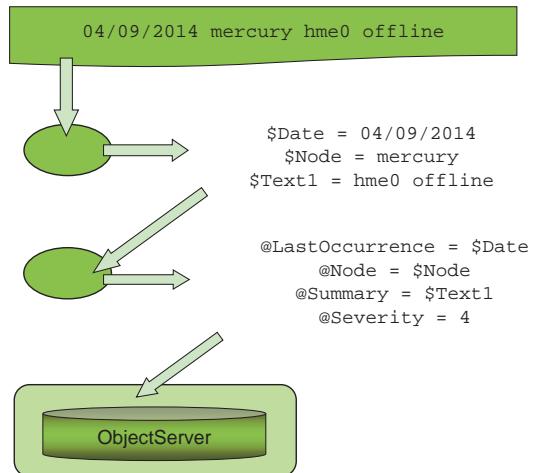
This slide lists some of the architectural limitations of the ObjectServer.

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



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 with 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)

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)

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 2016

9

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 2016

10

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 2016

11

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

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 that are 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

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 with the corresponding .log file

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 that they are only represented once in the ObjectServer

If an event comes in more than once:

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

© Copyright IBM Corporation 2016

15

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 more 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

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 with the Netcool/OMNIBus Administrator utility, nco_config

© Copyright IBM Corporation 2016

17

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

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.

Viewing properties with 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

Viewing properties with 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.

Viewing properties with nco_sql, continued

Some properties take immediate effect when changed

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

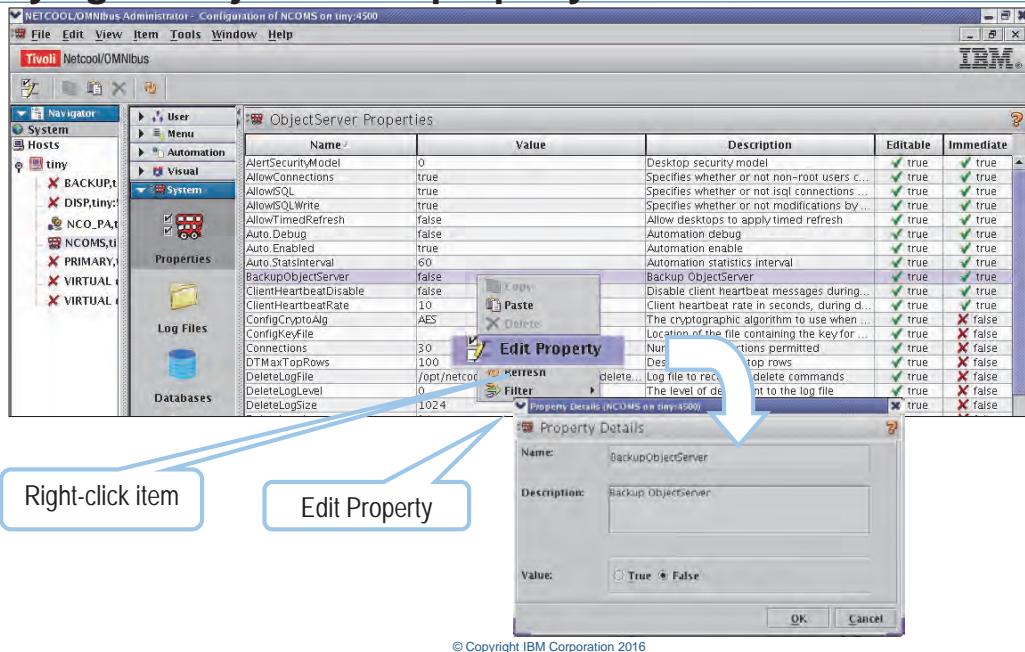
```
1> select PropName, Value from catalog.properties where
   IsImmediate=TRUE;
2> go
```

You can also find this information in the administrator GUI

Viewing properties with 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 2016

22

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 2016

23

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 2016

24

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 2016

25

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.

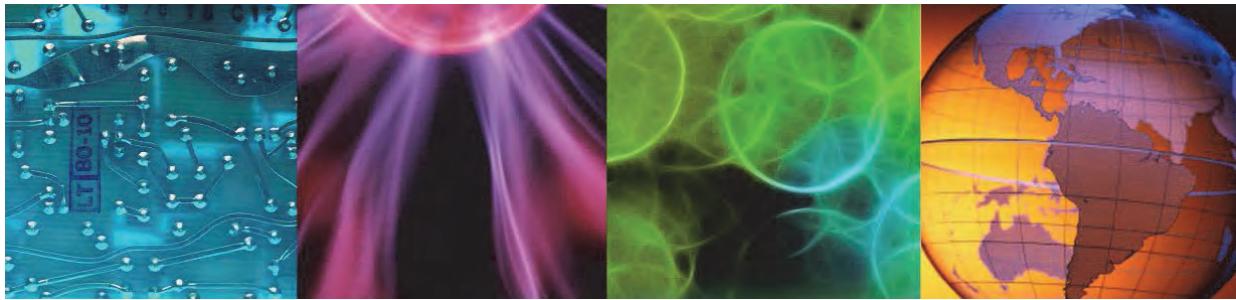
- **NHttpd.ListeningPort:** integer

Specifies the port on which the ObjectServer listens for HTTP requests. The default is 0.

Lesson 2 Modifying the ObjectServer structure



Lesson 2 Modifying the ObjectServer structure



© Copyright IBM Corporation 2016

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 use the Netcool/OMNIbus Administrator utility to modify the ObjectServer.

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

- Adding or changing columns in database tables (System)
- Adding or changing users (System)
- Adding or changing desktop menus and tools (System or Administrator)
- Adding or changing automations or triggers (System)

© Copyright IBM Corporation 2016

27

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

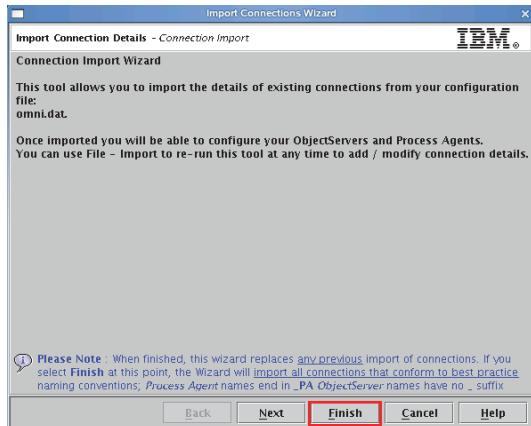
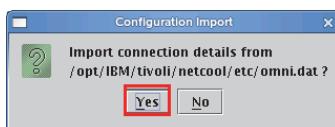
Most changes that are made by the Netcool/OMNIbus Administrator take effect immediately. Stopping and starting the ObjectServer is not needed to activate these changes.

Connection wizard

Tivoli Netcool/OMNIbus Administrator is launched from the command line

```
$OMNIHOME/bin/nco_config &
```

- Uses the interfaces file to identify components
- Wizard imports file and defines connections



© Copyright IBM Corporation 2016

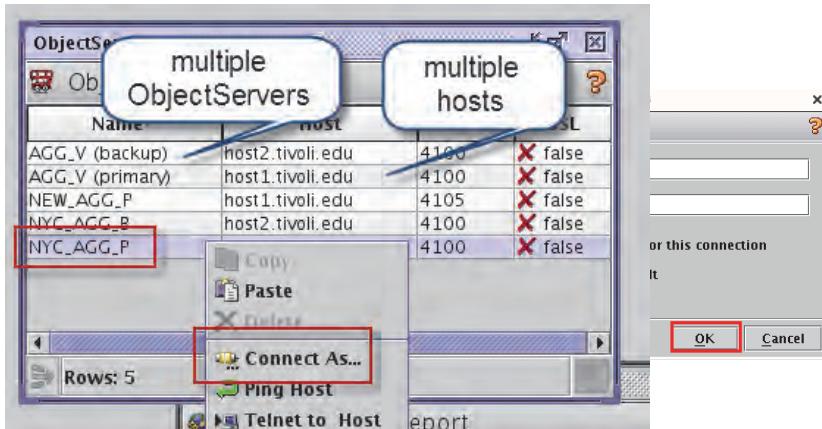
28

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 2016

29

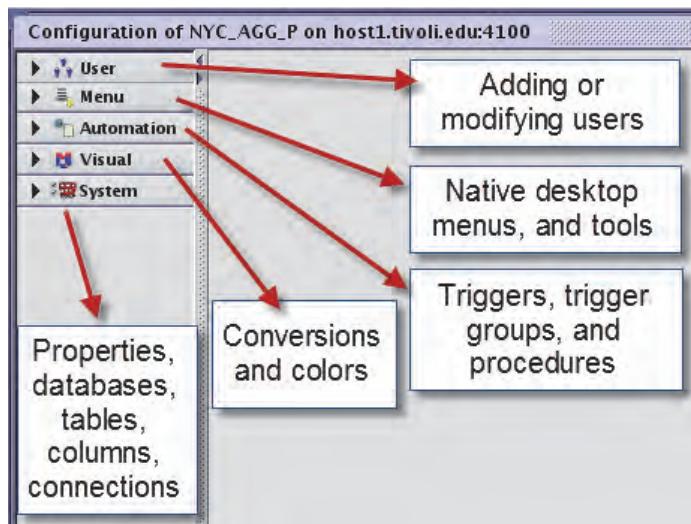
Tivoli Netcool/OMNibus Administrator

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. One box provides access to process activity daemons. The ObjectServer Report box contains a list of all ObjectServer definitions that are in the **omni.dat** file. In the example, there are multiple ObjectServers listed, and the ObjectServers reside on different host systems. You can use the administrator utility to manage all ObjectServers from a central location.

To connect to an ObjectServer, right-click the entry 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.

Navigating within the ObjectServer Administrator



© Copyright IBM Corporation 2016

30

Navigating within the ObjectServer Administrator

The administrator interface consists of a series of roll-up buttons. Each button corresponds to a functional aspect of the ObjectServer. To access the features within a category, you click the corresponding button to expand the list.

The User tab

Restriction filters

- An option that can be used to restrict the events the user can access
- Applied at the user or group level

Roles

- Define database and table access permissions
- Alter, delete, drop, insert, select, and update

Groups

- Used to associate common attributes to one or more users
- Roles, restriction filters
- Can use to restrict desktop tool access

Users

- Username: Textual user ID
- Full Name: Long name for user
- Restriction filter: If used
- Assigned groups
- Password: Internal or external

Name	Full Name	Type	ID	Extern...	Enabled
ncoadmin	Netcool Admin	Normal	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ncouser	Netcool User	Normal	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
nobody	Nobody	Unknown	65534	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
root	Root User	Super User	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
whill	William Hill	Normal	3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

© Copyright IBM Corporation 2016

31

The User tab

Any user that requires access to ObjectServer event data must have a user ID defined within the ObjectServer. Four functional aspects are related to ObjectServer users.

- Restriction filters are optional. A restriction filter is used to limit access to event records. In your organization, there might be users who have no need to view certain event data, for example, events that are related to security issues. You can create a restriction filter that prevents the user from seeing any security-related event records. A restriction filter can be applied to a user or a group.
- Roles determine access authority. Specifically, they control access to ObjectServer tables. Roles grant or deny access permissions to tables. A role determines whether a user is allowed to view, update, or delete data from a table. Other roles determine whether a user is allowed to modify the structure of a table. An ObjectServer administrator is associated with the roles that allow modifications to table structure. Normal users cannot modify table structure.
- Groups are used to associate one or more roles to users. You create roles, which provide access permissions, and associate the roles to a group. Next, you create a user, and assign the user to the group. The user inherits the access permissions for all roles that are assigned to the associated groups. An ObjectServer administrator is a user that belongs to a particular group: System. The System group contains the roles that grant the authority to make administrative changes to the ObjectServer.
- Users are defined within the ObjectServer as integer values, also referred to as a UID. When you create a user, you specify a textual user ID and a textual name. The user is created as an integer, and the textual values are added to a conversion table. The conversion table translates

the corresponding integer into the associated text. The password for a user can be stored within the ObjectServer, or in some external repository, such as LDAP.

The Menu tab

These objects all relate to the native desktop

Menus

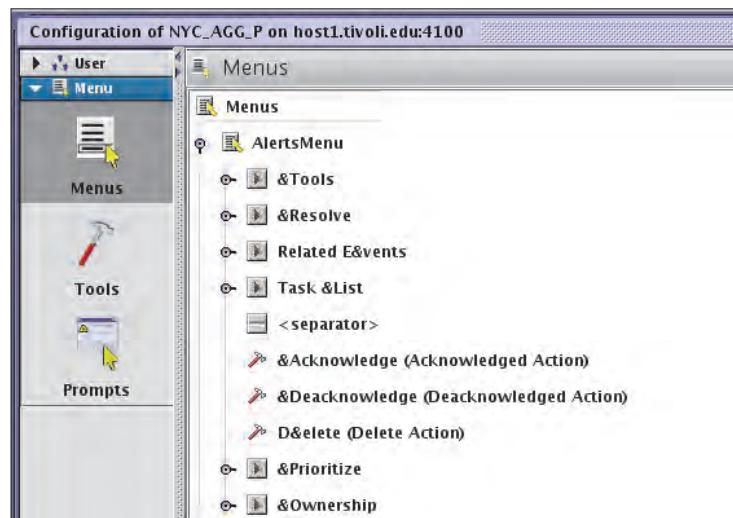
- Used to organize tools into collections
- Defines where on the desktop the tools are located
- A menu can contain another menu

Tools

- Command that is run when selected in a menu
- Can run SQL commands in the ObjectServer
- Can run an executable file on the user workstation

Prompts

- Used within a tool
- Allows the user to enter or select data that is used by the tool



© Copyright IBM Corporation 2016

32

The Menu tab

The **Menu** tab contains the features that are related to native desktop tools.

- Menus organize the tools. A desktop user accesses a menu, and the menu contains one or more tools.
- Tools are either SQL or executable. An SQL tool runs one or more SQL commands when selected. An executable tool runs a command, script, or bat file on the local workstation.
- You use prompts to allow user input to a tool. For example, a tool that is used to PING a device can prompt the user for the IP address of the target device.

The Automation tab

Trigger Groups

- Used to organize triggers into collections
- Can enable or disable the collection

Triggers

- Automated processes that run inside the ObjectServer
- Triggers are activated based on:
 - Timed frequency, for example every minute
 - Database change
 - Signal

Procedures

- Executable code called to perform common functions
- SQL procedures run inside the ObjectServer
- Executable procedures run outside of the ObjectServer
 - Can run on the local host or a remote host

User Defined Signals

- Can be used to cause a trigger to run based on some user-defined condition

Name	Enabled
AdvCorr	✓ true
audit_config	✗ false
automatic_backup_system	✓ true
compatibility_triggers	✓ true
connection_watch	✓ true
default_triggers	✓ true
fallback_triggers	✓ true
gateway_triggers	✓ true
iduc_triggers	✓ true
oslc	✓ true
primary_only	✓ true
profiler_triggers	✓ true
registry_triggers	✓ true
sae	✓ true
scala_triggers	✗ false
security_watch	✓ true
self_monitoring_group	✓ true
stats_triggers	✓ true
system_watch	✓ true
trigger_stat_reports	✓ true

© Copyright IBM Corporation 2016

33

The Automation tab

The **Automation** tab is used to manage ObjectServer triggers and procedures.

- Trigger groups organize triggers into logical groupings. A trigger group can be used to manage a collection of triggers. If you disable a trigger group, all triggers that are associated with that group no longer run.
- Triggers automate functions within the ObjectServer. The ObjectServer includes a number of triggers. These triggers are used to remove old events, correlate events, create log files for auditing purposes, and other functions.
- Procedures are collections of commands. You can invoke a procedure from a trigger, a desktop tool, or through a command-line utility. The ObjectServer includes a procedure to send an email. A trigger searches the ObjectServer for one or more vents that meet a defined criteria. If an event is found, the trigger invokes the send_email procedure, and passes selected fields from the corresponding event record. The result is an email message that contains data from the event record.

The Visual tab

Conversions

- Used to associate text to integer values
- The text is shown in the desktop instead of the integer

Colors

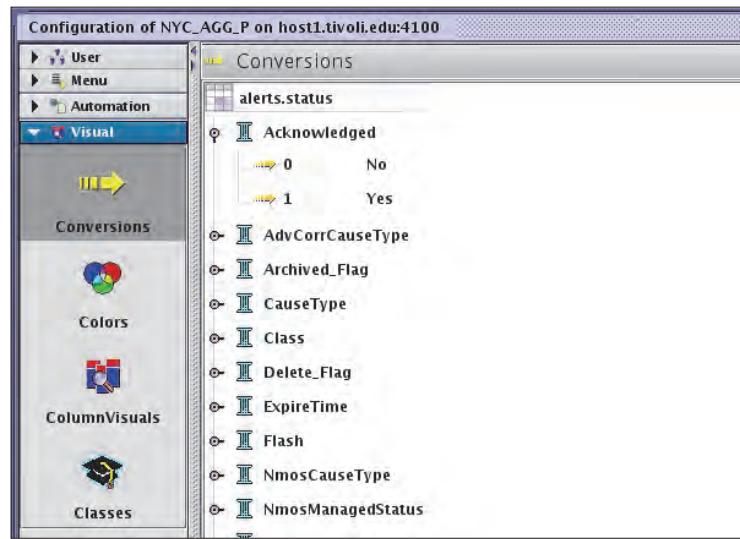
- Defines the color that is used to depict the various event severities

ColumnVisuals

- Defines the default display characteristics for event record columns
 - Column title, width, justification

Classes

- Classes group events by source
 - Example: Cisco events set to Class 410
- Tools can be enabled for different classes
 - Example: Only Cisco events



© Copyright IBM Corporation 2016

34

The Visual tab

The **Visual** tab contains the features that are used to define how event data appears in a user desktop.

- Conversions are used to relate text to integers. An ObjectServer user is defined as an integer. When a user displays any event data that relates to the user ID, a conversion takes place and the user sees the textual user name. The ObjectServer includes conversions for various event record columns, including: Severity, Acknowledged, OwnerUID, and others. Administrators create conversion entries for any custom integer event columns that they create.
- Colors define the specific color that a user sees in their desktop when they view certain event columns, like Severity.
- ColumnVisuals define the default event view. When a user displays an event column in a desktop, the column display has certain characteristics. For example, the column width, the title of the column, the justification of the data within the column and others.
- Classes define an integer value for various categories of events. The most common category is vendor. The ObjectServer includes class definitions for a number of equipment manufacturers. The events for a specific manufacturer contain a unique class value. The class can be used to organize events in a display or to limit the events. The class can also be used to associate a desktop tool to the events for a certain equipment manufacturer. The tool might be used to connect to the vendor-specific element management system to help diagnose a hardware issue.

The System tab

Properties

- Control basic behavior of the ObjectServer

Log Files

- Defines the physical characteristics of a log file
 - Name, location, size
- The log file is populated with a trigger

Database

- Used to create or modify databases, and tables within the ObjectServer

SQL

- SQL workbench
- Used to manually run one or more SQL commands within the ObjectServer

Connections

- Shows every component that is connected to the ObjectServer

Channels

- Define connections that are used by the Accelerated Event Notification component

Name / Value	Description	Editable	In
ActingPrimary	true	Acting Primary ObjectServer	✓ true
AlertSecurityModel	0	Desktop security model	✓ true
AllowConnections	true	Specifies whether or not ...	✓ true
AllowISQL	true	Specifies whether or not ...	✓ true
AllowSQLWrite	true	Specifies whether or not ...	✓ true
AllowTimedRefresh	false	Allow desktops to apply ...	✓ true
Auto.Debug	false	Automation debug	✓ true
Auto.Enabled	true	Automation enable	✓ true
AutoStatsInterval	60	Automation statistics inte...	✓ true
BackupObjectServer	false	Backup ObjectServer	✓ true
ClientHeartbeatDl...	false	Disable client heartbeat ...	✓ true
ClientHeartbeatRate	10	Client heartbeat rate in s...	✓ true
ConfigCryptoAlg	AES	The cryptographic algorit...	✓ true
ConfigKeyfile		Location of the file contai...	✓ true
Connections	256	Number of connections ...	✓ true
DeleteLogFile	/opt/IBM/tivoli/netcool/o...	Log file to record all dele...	✓ true
DeleteLogging	false	Turn on the delete logging	✓ true
DeleteLogLevel	0	The level of detail sent t...	✓ true
DeleteLogSize	1048576	The maximum size of th...	✓ true
DTMaxTopRows	100	Desktop maximum top r...	✓ true
Granularity	60	Iduc update granularity.	✓ true
GWDEDuplication	0	Gateway deduplication ...	✓ true
Help	false	Display help information.	✓ true
Hostname	host1.tivoli.edu	ObjectServer's hostname	✗ false
IduC.ListeningHost...		Hostname to listen for Iduc...	✓ true
IduC.ListeningPort	0	Iduc port to listen on.	✓ true
Ipc.QueueSize	1024	Size of middleware interna...	✓ true
Ipc.ServerCharacter...	utf8	Server's character set	✗ false
Ipc.ServerLanguage	us_english	Server's language	✗ false
Ipc.ServerLocale	en_US.UTF-8	Server's locale	✗ false

Rows: 108

© Copyright IBM Corporation 2016

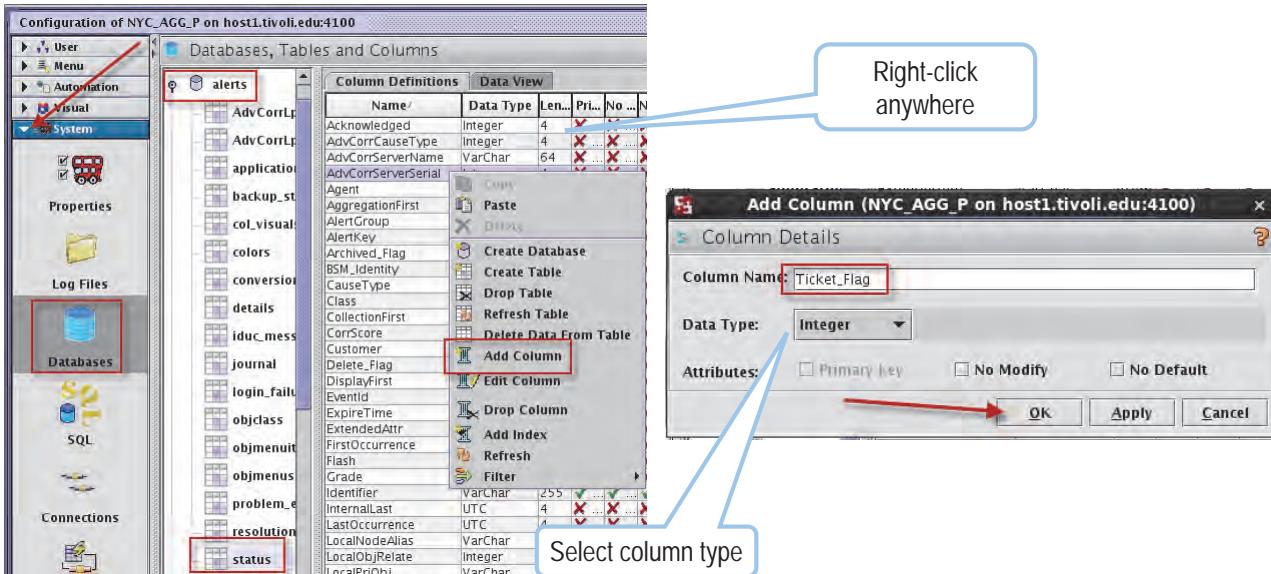
35

The System tab

The **System** tab contains the features that control some of the basic ObjectServer behavior. Other features display information that is related to ObjectServer operation.

- Properties control much of the basic behavior of the ObjectServer. You can change some properties. Other properties cannot be changes. Some property changes take effect immediately, and other changes require an ObjectServer restart. Do not change a property unless you understand what the change does.
- Log Files define external files. Triggers write to these files. A trigger cannot specify things like the name of the file and where it resides. An example is the profile report. The Log Files feature contains a definition for the file name and location. Several triggers write data into this log file.
- Databases provide access to every ObjectServer database and the tables within each database. You use this feature to modify the structure of the event record and add extra columns.
- SQL provides a command-level utility that is used to enter one or more SQL commands. The capability is similar to the command-line nco_sql utility. This feature is commonly referred to as a *workbench*. An administrator can enter SQL commands to search an ObjectServer table or to modify the contents of a table.
- Connections provides a list of every component that is connected to the ObjectServer. An administrator can view the connections, and selectively drop any connection.
- Channels are used by the Accelerated Event Notification feature.

Adding a column to the ObjectServer event record table



© Copyright IBM Corporation 2016

36

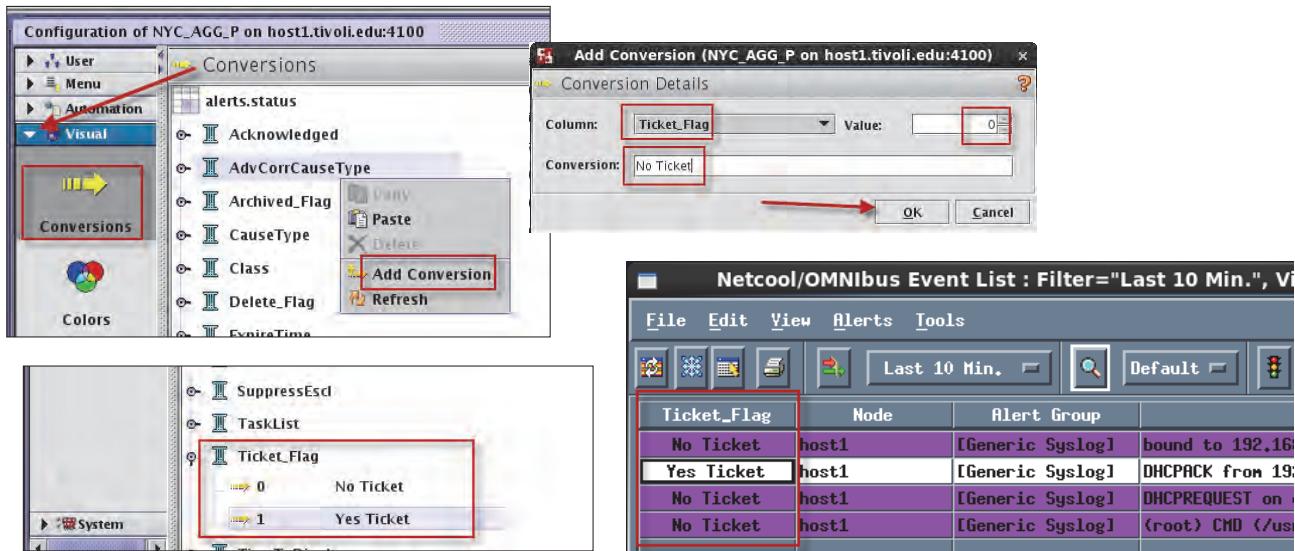
Adding a column to the ObjectServer event record table

Adding a column to the ObjectServer event record is an example of modifying the structure of the ObjectServer. You change the structure of the alerts.status table when you add a column. Adding a column to the table does not change the contents of the table. The new column appears in the table, but the column has no data.

Adding a column to the event record is one of the most common administrative functions. Many users might add columns that are used to contain customer-specific data. For example, business unit name or point of contact email address. Another use for a custom column is to control certain behavior. For example, whether to create a trouble ticket. Netcool/OMNIbus provides a gateway that integrates to IBM SmartCloud Control Desk, the IBM trouble ticket system. An administrator can add a column to the event record called Ticket_Flag. The administrator creates a desktop tool called Open Ticket. A user selects an event in the desktop and runs the tool. The tool runs an SQL that updates the Ticket_Flag column and sets the value to 1. The gateway searches the ObjectServer events for any event where Ticket_Flag=1. When the gateway locates the event, it passes event data to SmartCloud Control Desk. SmartCloud Control Desk creates an incident based on the data from the event record. The gateway passes the incident number back to the ObjectServer, and adds that value to a column called Ticket_ID.

To add a column to the event record, you expand the **System** tab and select **Databases**. You expand the **alerts** database and select the **status** table. The list of column names is displayed. Right-click any place in the list of columns and select **Add Column**. Enter the column name with no spaces or special characters. Select the column type and click **OK**.

Defining column conversion values



© Copyright IBM Corporation 2016

37

Defining column conversion values

It is a recommended best practice to create conversions for any custom integer event columns that you create. The conversion values offer a convenience to desktop users. The users see the text instead of an integer. The users might not be aware of what the integer value represents.

To add conversion values for the custom column, you expand the Visual tab and select Conversions. The list of column names with conversions defined appears. Right-click any place in the list of columns and select Add Conversion. Select the column name from the drop-down list. Enter the integer value and the corresponding text. Click OK. Right-click the new conversion, and select Add Conversion. Enter another integer value and the corresponding text. Click OK. Repeat these steps to enter text for all possible integer values.



Lesson 3 Creating ObjectServers



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Create an ObjectServer from the command line
- Create a best practice aggregation ObjectServer

Creating a default ObjectServer

Utility to create an ObjectServer

```
$OMNIHOME/bin/nco_dbinit -server <ObjectServer name>
```

Input

```
$OMNIHOME/etc/application.sql  
$OMNIHOME/etc/automation.sql  
$OMNIHOME/etc/desktop.sql  
$OMNIHOME/etc/security.sql  
$OMNIHOME/etc/system.sql
```

Output

```
$OMNIHOME/etc/<ObjectServer name>.props  
$OMNIHOME/db/<ObjectServer name>/  
$OMNIHOME/db/<ObjectServer name>/master_store.tab  
$OMNIHOME/db/<ObjectServer name>/table_store.tab
```

Do not modify any of the default SQL files

© Copyright IBM Corporation 2016

39

Creating a default ObjectServer

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 initial ObjectServer framework on disk. The input to the utility is a collection of SQL files. The SQL files contain commands that create the ObjectServer framework. Some of the files create the ObjectServer databases and tables. Some of the files insert records into these tables. The inserted records constitute the default values for various objects, including these examples:

- Menus and tools
- Triggers, trigger groups, and procedures
- users, groups, and roles
- others

The initial framework is in the form of two checkpoint files. The first time the ObjectServer starts, these checkpoint files are read and the framework is created in memory. After the initial start, new checkpoints are created every 60 seconds.

The utility also creates the initial ObjectServer property file. This file contains the default values for all ObjectServer properties.



Important: Do not modify any of the default SQL files. When the Netcool/OMNIBus software is upgraded, the upgrade might modify one or more of these files.

Creating a custom ObjectServer

Utility to create an ObjectServer

```
$OMNIHOME/bin/nco_dbinit -server <ObjectServer name> -customconfigfile  
directory/your_file.sql
```

Input

```
$OMNIHOME/etc/application.sql  
$OMNIHOME/etc/automation.sql  
$OMNIHOME/etc/desktop.sql  
$OMNIHOME/etc/security.sql  
$OMNIHOME/etc/system.sql  
directory/your_file.sql
```

ObjectServer created with default content plus customizations, for example:

- Additional columns in alerts.status
- Custom triggers
- Custom tools
- Others

© Copyright IBM Corporation 2016

40

Creating a custom ObjectServer

The nco_dbinit utility provides an option to include ObjectServer modifications when the ObjectServer is created. This feature essentially eliminates the need to modify any of the default SQL files. If you want to include modifications when the ObjectServer is created, you can put the necessary commands in a separate SQL file. You reference that file when you run the nco_dbinit utility. The ObjectServer that is created contains all of the default framework and any customizations in the additional file.

Most users do not need to use this capability. Even though most users customize their ObjectServers, the modifications are typically made after the ObjectServer is created. The customization feature is most beneficial to users that must create multiple ObjectServers. For example, a large multinational enterprise might create multiple ObjectServers to distribute their management infrastructure geographically. The administrator can create multiple ObjectServers with the same customizations, which eliminates the need to modify each ObjectServer after they are created.

Creating the best practice aggregation ObjectServer

Utility to create an ObjectServer

```
$OMNIHOME/bin/nco_dbinit -server <ObjectServer name> -customconfigfile  
$OMNIHOME/extensions/multitier/objectserver/aggregation.sql
```

Input

```
$OMNIHOME/etc/application.sql  
$OMNIHOME/etc/automation.sql  
$OMNIHOME/etc/desktop.sql  
$OMNIHOME/etc/security.sql  
$OMNIHOME/etc/system.sql  
$OMNIHOME/extensions/multitier/objectserver/aggregation.sql
```

ObjectServer created with default content plus best practice aggregation customizations

Creating the best practice aggregation ObjectServer

The recommended best practice for ObjectServer configurations is the multitier architecture. The multitier architecture is based on multiple layers of ObjectServers. The layers are categorized as collection, aggregation, and display. Netcool/OMNibus includes a collection of files that are used to create the multitier architecture. These files include the customizations that are recommended for each layer in the architecture. All three layers are recommended for those enterprises with enormous amounts of event data. However, for many enterprises, only a single layer is required. For a single-layer architecture, the recommendation is to deploy the aggregation layer.

You can use the nco_dninit utility to create an ObjectServer that includes the recommended best practice customizations for an aggregation ObjectServer. The modifications are contained within an SQL file: aggregation.sql. You merely run the nco_dbinit utility and include that file.

Student exercises



© Copyright IBM Corporation 2016

42

Student exercises

Summary

You now should be able to perform the following tasks:

- Describe the ObjectServer structure
- Describe important event record columns and their use
- Describe important ObjectServer properties
- Use the administration GUI to modify a property
- Use the administration GUI to modify an ObjectServer
- Create an ObjectServer



3 Probes



Probes



© Copyright IBM Corporation 2016

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

In this unit, you learn about some of the advanced features for probes.

References: SC27-6266-00 *Netcool/OMNibus Version 8 Release 1 Probe and Gateway Guide*,
SC23-6386-17 *IBM Tivoli Netcool/OMNibus Knowledge Library Reference Guide*

Objectives

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

- Use the MIB Manager utility to import a MIB file
- Use the MIB Manager utility to generate rules files
- Incorporate generated rules into the Netcool Knowledge Library
- Use the MIB Manager utility to generate test SNMP trap messages
- Configure probe peer-to-peer mode
- Implement the procedure to reload rules for all active probes



Lesson 1 MIB Manager

Lesson 1 MIB Manager



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Use MIB Manager to import a MIB file
- Use MIB Manager to generate rules files
- Use MIB Manager to generate SNMP traps

MIB Manager overview

Convert SNMP MIB into rules files for use with SNMP probe

Installs as part of Netcool/OMNIbus probe support option

Java GUI application based on the eclipse platform

Includes most common base and RFC SNMP MIB files for resolving imported MIB dependencies

MIB Manager overview

MIB Manager is a Java based graphical tool that is based on the Eclipse platform. MIB Manager is included with Netcool/OMNIbus. The tool is used to convert SNMP MIB files into Netcool/OMNIbus rules. The tool includes many of the common base MIB files. Vendor-specific MIB files reference the base files.

MIB Manager 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 to send to SNMP probe

Export to multiple format rules files including:

- Netcool Knowledge Library
- Lookup files

Select multiple MIBs or MIB objects for export

MIB Manager functions

The utility provides a number of features:

- Resolution of MIB dependencies: Many MIB files contain references to objects that are defined in other MIB files. The utility uses the library of objects to automatically resolve all references that are found during an import operation.
- Group MIB files into devices: A MIB Manager device is a logical entity. You create a device based on the MIB files that are related to a vendor, model, and operating system. You can use a device to organize all MIB objects for a specific vendor, like IBM. You can use the device to export all objects for the vendor.
- Configure traps with preset values: You use the MIB Manager utility to generate rules to interpret traps, and generate events. A primary characteristic of an event is the severity. The severity determines how serious the issue is, and typically you use the severity to prioritize the resolution efforts. The rules that the utility generates contain severity settings for every event. You can set the value that you want in the MIB Manager utility. Then, when the utility generates rules, the rules contain your settings.
- Generate test SNMP traps: You use the utility to generate rules. Then, you can use the utility to generate a test SNMP trap that you use to verify the rules.

Launching MIB Manager

Use the wrapper script:

```
$NCHOME/omnibus/bin/nco_mibmanager
```

Will not launch if an X-display is not available on UNIX

Most likely cause of failure to launch is because of gtk library dependencies

Logs to troubleshoot upon launch failure:

```
$NCHOME/omnibus/platform/<arch>/mibmanager/configuration/<timestamp>.log
```

```
$NCHOME/omnibus/platform/<arch>/mibmanager/workspace/.metadata/.log
```

Launching MIB Manager

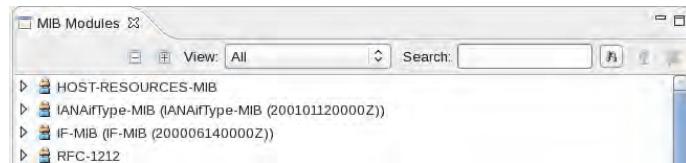
MIB Manager is a graphical utility. On a UNIX system, you must run the utility from a console. Or, you must use something like X Window System.

MIB Module view

Shows all of the MIB modules available to MIB Manager

Select MIB modules to export

Search by name



Filter which objects appear in this view

© Copyright IBM Corporation 2016

7

MIB Module view

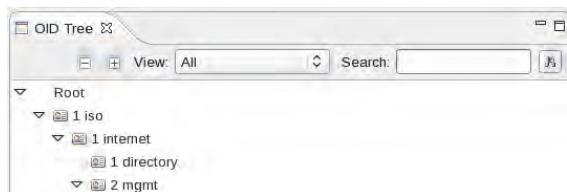
The user interface consists of several windows. One window contains the list of known MIB modules. You use this window to explore the contents of a MIB module and select objects for export.

Object Identifier (OID) view

Shows all of the MIB objects available to MIB Manager

Select objects to export

Search by name or OID



Filter which objects appear in this view

Object Identifier (OID) view

Another window is called the OID view. This window displays the known Object Identifiers in the hierarchical dotted decimal notation.

Details view

Show specific information about the selected MIB object

Field information can be entered for a trap

Field	Value (double-click to edit)
OID	1.1.2.1.11.0.0
Name	coldStart
ObjectType	TRAP-TYPE
Version	SNMP v1
ENTERPRISE	snmp
Specific Trap	0
DESCRIPTION	A coldStart trap signifies that the sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation may be altered.
@Severity	3 - Minor
@Type	

© Copyright IBM Corporation 2016

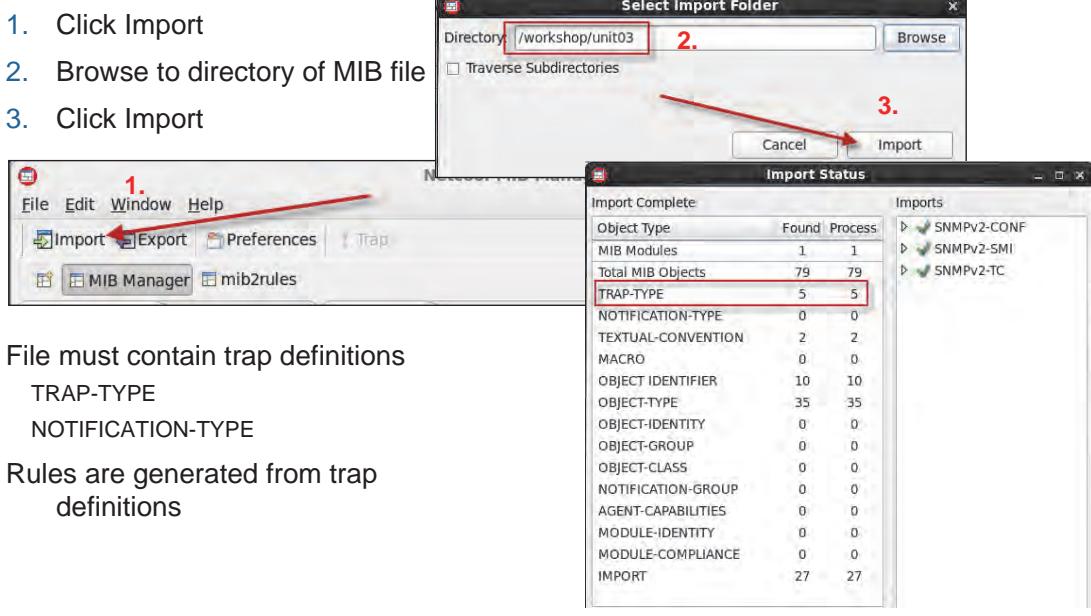
9

Details view

The Details view contains the characteristics for an individual MIB object. You use this view to enter custom values for ObjectServer columns. The custom values are used to create the corresponding rules.

Importing a MIB file

1. Click Import
2. Browse to directory of MIB file
3. Click Import



10

Importing a MIB file

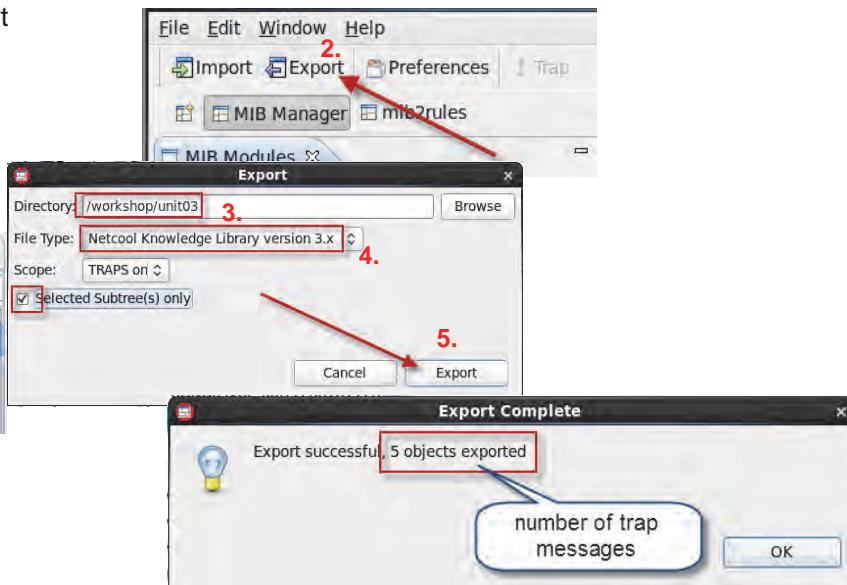
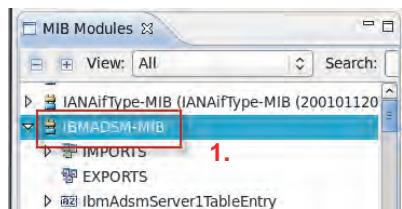
An SNMP MIB file is created in a specific format: Abstract Syntax Notation. Vendors generally supply MIB files for their devices that support access with SNMP. To import a MIB file, you place the file in some directory. You start MIB Manager, and click Import. You browse to the directory where you saved the MIB file. You supply only the directory path, not the individual file name. MIB Manager processes every file in the directory. After you select the directory, you click Import.

The import process normally runs quickly. If you import multiple files at the same time, the process might take longer. When the import completes, a status window opens. The window contains a summary of the standard MIB object names, and the count of each object that was found during the import.

MIB Manager generates rules based on specific MIB objects. The TRAP-TYPE, and NOTIFICATION-TYPE objects contain the trap definitions. If the utility does not find any trap definitions in the MIB file, you cannot generate rules from the MIB file.

Generating rules

1. Select MIB objects to export
2. Click Export
3. Select directory for output
4. Select output format
5. Click Export



© Copyright IBM Corporation 2016

11

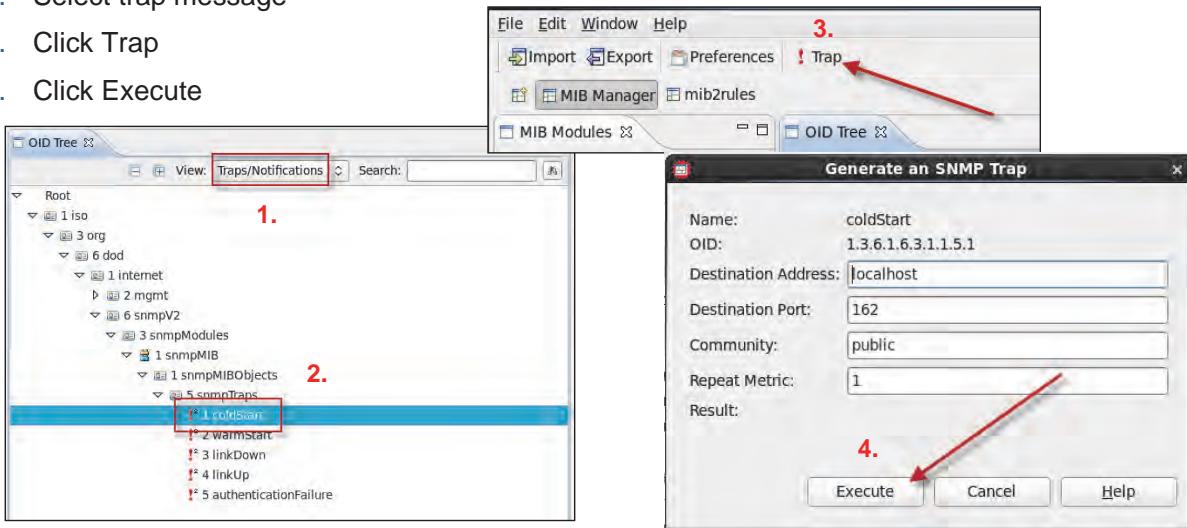
Generating rules

After the file is imported, you can use the utility to generate rules. You select one or more MIB modules. You can select a device, if one is available. Then, you click Export. You browse to the location where you want the output files saved. You select the type of output. The utility can export selected MIB objects in various formats, including: CSV, HTML, and several types of rules format. You click Export, and the utility creates the files in the directory that you specified.

It is suggested that you create rules in the Netcool Knowledge Library format. The Netcool Knowledge Library contains a collection of rules for various vendors, and technologies.

Generating SNMP traps

1. Change View to Traps/Notifications
2. Select trap message
3. Click Trap
4. Click Execute



Generating SNMP traps

You can also use MIB Manager to generate a test SNMP trap. The test trap is especially useful for validating the rules that are generated with the utility.

You select a trap object, and then click Trap. A new window opens that contains the attributes for the trap. You can set the host name or IP for the trap destination. You can set the trap port number, and community string. You can use the Repeat Metric to generate the same trap multiple times. Then, you click Execute, and the trap is generated.

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 2016

13

Netcool/OMNIbus Knowledge Library

Netcool/OMNIbus Knowledge Library is a collection of rules files that are written to a common standard, and provides event correlation, and causal analysis for the IBM Tivoli Netcool suite. The default rules file necessary for the execution of a probe performs only generic grouping of data. A rules file enhanced to cater for SNMP Management Information Base (MIB) events from a specific device provides sharpened event enrichment and causal analysis. The Netcool/OMNIbus Knowledge Library is a collection of such enhanced rules files that are tuned to specific managed objects that send SNMP-based events.

The Netcool/OMNIbus Knowledge Library increases the ability of the Tivoli Netcool/OMNIbus ObjectServer automations to correlate alarms, and identify root causes by employing the following techniques:

- Event Pre-Classification: This process identifies, and flags events within the probe rules files to indicate the causal relevance of events, where this relevance can be determined without the need for correlation.
- Intra-Device Correlation: This process enhances probe rules files with the addition of automations to the ObjectServer. These automations perform correlations beyond deduplication, and problem or resolution correlation. The automations identify intra-device root causes, and symptoms.

Installed files

Directory	Description
/rules	The base IBM Netcool/OMNIbus Knowledge Library directory which contains the base rules files.
/rules/include-common	This directory contains include files that provide probe-independent logic - for example, 3GPP and TMF814 specific log, lookup table to help convert between hex, decimal, and ASCII.
/rules/include-compat	This directory contains include files to activate support for various IBM Tivoli Netcool/OMNIbus Knowledge Library features.
/rules/include-snmptrap	This directory contains include files for processing events from the SNMP trap-based probes.
/rules/include-snmptrap/common-lookup	This directory contains the common lookup files that any vendor can use.
/rules/include-snmptrap/generic	This directory contains SNMP trap-based include files that improve the handling of enterprise-specific implementations of the generic SNMP traps.

© Copyright IBM Corporation 2016

14

Installed files

Netcool Knowledge Library is delivered as a compressed archive file. To install Netcool Knowledge Library, you expand the archive into a directory. Then, you define the NC_RULES_HOME environment variable to point to this target directory.

The Netcool Knowledge Library contains over 2000 files. Most of these files contain rules based on vendor-specific MIB files.



Hint: The Netcool Knowledge Library Reference Guide contains a list of the supported vendors. The vendor files are contained in vendor-specific directories.

Within each vendor directory, a *master rules file* contains a collection of include statements. Each include statement references an individual vendor rules file. In addition, there is one *master rules file* for the entire Netcool Knowledge Library. This master rules file also contains a collection of include statements. However, each of these include statements references one of the vendor-specific master rules file.

For example, IBM is a supported vendor. There is an IBM vendor directory. This directory contains a collection of files that contain the rules for all supported IBM MIB files. The master rules contains one include statement for each individual file. There is a Netcool Knowledge Library master rules file. This rules file contains an include statement that references the IBM master rules file.

In addition to the rules files, there are also a collection of *lookup files*. Each lookup file contains a collection of include statements. These include statements reference files that contain statements

for creating lookup tables. The lookup tables are used for various purposes, which vary based on the vendor. There is one master lookup file for each vendor. There is also one master lookup file for the Netcool Knowledge Library.

Installed files, continued

Directory	Description
/rules/include-snmptrap/vendor	This directory identifies the specific vendor and contains the following set of master and preclass files that improve the handling of enterprise-specific implementations of the traps specific to the vendor:
vendor.master.include.lookup	This file lists all the lookup files related to that vendor.
vendor.master.include.rules	This file lists all the rules files related to that vendor.
vendor-preclass.snmptrap.lookup	This file contains all the PreClassification entries related to that vendor.
vendor-preclass.include.snmptrap.rules	This file maps the PreClassification entries to ObjectServer.
vendor-MIB.include.snmptrap.rules	This file contains the include statement defining the path to the vendor specific rules file in the include-snmptraps directory.
vendor-MIB.include.snmptrap.lookup	This file contains the include statement defining the path to the vendor specific lookup file in the include-snmptraps directory.

Installed files, continued

This slide illustrates some of the vendor-specific files that are included with the Netcool Knowledge Library.

Installed files, continued

Directory	Description
<code>vendor-MIB.user.include.snmptrap.rules</code>	This file contains the include statement defining the path to the vendor specific rules file of the user in the include-snmptrap directory.
<code>vendor-MIB.sev.snmptrap.lookup</code>	This file lists the severity lookup files related to the vendor.
<code>vendor_MIB.adv.include.snmptrap.lookup</code>	This file contains the include statement defining the path to the vendor specific lookup file for the advanced traps in the include-snmptrap directory.

The files shown on these slides are for the SNMP probe
Additional files are included for use with the Syslog probe

© Copyright IBM Corporation 2016

16

Installed files, continued

The Netcool Knowledge Library contains rules for the SNMP probe, and a separate set of files for the Syslog probe. The files on these slides are specific to the SNMP probe. Other files are used for the Syslog probe.

Enabling probe to use vendor rules

Default configuration uses only generic rules, no vendor-specific rules

Uncomment the following lines from the snmptrap.rules file:

```
#include "$NC_RULES_HOME/include-snmptrap/vendor/vendor.master.include.lookup"  
#include "$NC_RULES_HOME/include-snmptrap/vendor/vendor.master.include.rules"  
#include "$NC_RULES_HOME/include-snmptrap/vendor/vendor.preclass.include.  
snmptrap.rules"
```

Where vendor is the subdirectory specific to the vendor

For example, following is the section of the snmptrap.rules file when the vendor is IBM:

```
include "$NC_RULES_HOME/include-snmptrap/ibm/ibm.master.include.lookup"  
include "$NC_RULES_HOME/include-snmptrap/ibm/ibm.master.include.rules"  
include "$NC_RULES_HOME/include-snmptrap/ibm/ibm.preclass.include.snmptrap.rules"
```

Enabling probe to use vendor rules

The master rules file and master lookup files for the Netcool Knowledge Library contain collections of include statements for other master rules files, and lookup files. However, most of the include statements are commented out. Each Netcool administrator must uncomment the rules files that relate to the equipment used in their respective infrastructures. To enable support for a vendor, the administrator removes the comment character from three include statements. This slide contains an example for enabling support for IBM.

Adding rules from MIB Manager for a new vendor

If the vendor does not exist in the Netcool/OMNIbus Knowledge Library:

1. Create vendor directory

```
$NC_RULES_HOME/include-snmptrap/vendor/
```

2. Copy all files from MIB Manager to vendor directory

3. Place includes for the vendor master files into snmptrap.rules

```
include "$NC_RULES_HOME/include-snmptrap/vendor/vendor.m2r.master.include.lookup"  
include "$NC_RULES_HOME/include-snmptrap/vendor/vendor.m2r.master.include.rules"  
include "$NC_RULES_HOME/include-snmptrap/vendor/vendor.preclass.include.  
snmptrap.rules"
```

For example, if the vendor is IBM:

```
include "$NC_RULES_HOME/include-snmptrap/ibm/ibm.m2r.master.include.lookup"  
include "$NC_RULES_HOME/include-snmptrap/ibm/ibm.m2r.master.include.rules"  
include "$NC_RULES_HOME/include-snmptrap/ibm/ibm.preclass.include.snmptrap.rules"
```

© Copyright IBM Corporation 2016

18

Adding rules from MIB Manager for a new vendor

You use the MIB Manager utility to generate rules for a MIB file that is not contained within the Netcool Knowledge Library. If you find an event with a Summary field like *Unknown Enterprise OID*, this message indicates that the SNMP probe received a trap, and the probe did not contain rules to interpret the trap message. The first thing to check is whether the Netcool Knowledge Library contains rules to interpret the trap. The trap might be supported, but the required include statements might be commented out. If the Netcool Knowledge Library does not contain the necessary rules, then you need to locate a copy of the vendor MIB file. You import the rules into MIB Manager, and then export the trap definitions into rules files. Then, you must incorporate the new files into Netcool Knowledge Library.

You must determine whether the new rules are for a vendor that exists in Netcool Knowledge Library. The process to incorporate rules for a new vendor is different than if the vendor exists.

If the vendor does not exist in the Netcool Knowledge Library, you must create a vendor-specific directory. The output from MIB Manager is contained within a vendor directory. You can copy the directory and contents to the Netcool Knowledge Library.

The files that MIB Manager creates contain a master rules file, and a master lookup file. You must edit the *Netcool Knowledge Library master files* and add include statements for the new vendor files.

Adding rules from MIB Manager for an existing vendor

1. Uncomment lines in snmptrap.rules to enable vendor support

2. Copy all files from MIB Manager to *vendor* directory:

```
$NC_RULES_HOME/include-snmptrap/vendor/
```

3. Place includes for the vendor.m2r master files into the existing *vendor* files:

```
$NC_RULES_HOME/include-snmptrap/vendor/vendor.master.include.lookup
```

```
$NC_RULES_HOME/include-snmptrap/vendor/vendor.master.include.rules
```

4. Merge the <vendor>/<vendor>-preclass.snmptrap.lookup file into the existing preclass lookup file:

```
$NC_RULES_HOME/include-snmptrap/vendor/vendor.preclass.include.snmptrap.rules
```

Adding rules from MIB Manager for an existing vendor

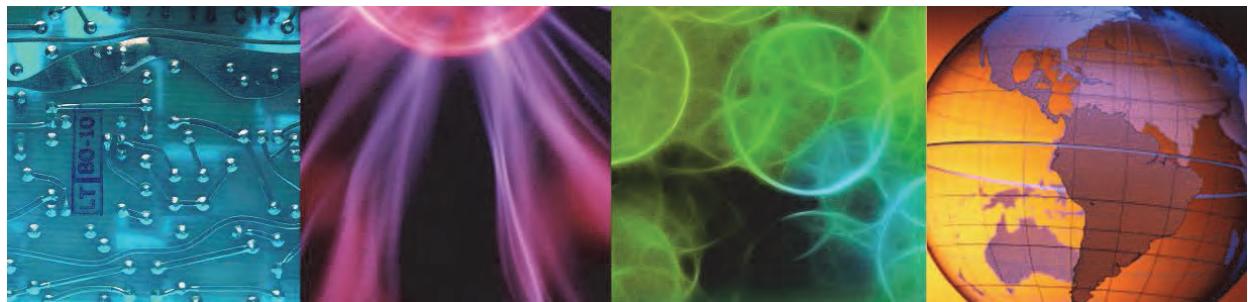
If the vendor exists in the Netcool Knowledge Library, the process is easier. The vendor directory exists, the Netcool Knowledge Library master rules file, and master lookup file contain include statements for the vendor-specific master files. Make sure that those include statements are not commented out.

You copy the files from MIB Manager into the existing vendor directory in the Netcool Knowledge Library. You must update the vendor master rules file, and master lookup files to add include statements for the new rules, and lookups.

One more file that must be modified. Each vendor has a lookup table that defines every supported trap message. The table is used by the preclassification process. MIB Manager creates a lookup table with entries for traps that are defined in the MIB file that it processed. The existing vendor directory contains a lookup table with the known trap messages. You must update the existing file, and add the new messages. You essentially copy the entries from the MIB Manager file, and paste them into the existing vendor file.



Lesson 2 Probe high availability



© Copyright IBM Corporation 2016

In this lesson, you learn how to configure the peer-to-peer probe option.

peer-to-peer overview

Two instances of a probe can run simultaneously in a peer-to-peer failover relationship

- One instance is designated as the master
- The other instance acts as a slave and is on hot standby

If the master instance fails, the slave instance is activated

Note: Peer-to-peer failover is not supported for all probes

Probes that list the **Mode**, **PeerHost**, and **PeerPort** properties when you run the command \$OMNIHOME/probes/nco_p_probename -dumpprops support peer-to-peer failover

peer-to-peer overview

Some Netcool/OMNibus probes support peer-to-peer mode. You configure two probes in peer-to-peer mode to provide high availability for the probe. In peer-to-peer operation, one probe is designated a master, and the other is the slave. Both probes receive data from their event source, and generate ObjectServer events. However, only the master probe sends the events to the ObjectServer. The slave probe caches the event records locally. The master probe sends a heartbeat message to the slave. When the slave receives the heartbeat message, it removes the events from the local cache. If the slave does not receive a heartbeat message within a user-specified interval, the slave forwards the events from the cache to the ObjectServer. It continues to forward subsequent events until it receives a heartbeat.

The peer-to-peer option is not available for all probes. You can run the probe with the dumpprops option, and examine the output. If the output contains, Mode, PeerHost, and PeerPort property settings, the probe supports peer-to-peer mode.

Configuring peer-to-peer mode

To set up a peer-to-peer failover relationship:

- For the master instance:
 - Set the **Mode** property to master
 - Set the **PeerHost** property to the network element name of the slave
- For the slave instance:
 - Set the **Mode** property to slave
 - Set the **PeerHost** property to the network element name of the master
- For both instances:
 - Set the **PeerPort** property to the port through which the master and slave communicate

Configuring peer-to-peer mode

In peer-to-peer mode, two copies of the probe run simultaneously. Ideally, you run the probes on different physical servers. You configure each probe to collect information from the same source. For example, you run two copies of the SNMP probe. You configure the equipment in your infrastructure to forward traps to both probes. You configure the peer-to-peer property settings for each probe, and start each probe.

peer-to-peer operation

The master instance sends a heartbeat poll to the slave instance at the time interval specified by the **BeatInterval** property

The slave instance caches all the alert data it receives, and deletes all alert data from the cache each time a heartbeat is received from the master instance

If the slave instance receives no heartbeat in the time period defined by the sum of **BeatInterval + BeatThreshold**, the slave instance:

- Assumes that the master is no longer active
- Forwards all alerts in the cache to the ObjectServer
- The slave instance continues to forward all alerts until it receives another heartbeat from the original master instance

The timeout period while waiting for heartbeats is 1 second

There can be a maximum delay of **BeatInterval + BeatThreshold + 1** seconds before the slave instance forwards its cached alerts

peer-to-peer operation

The **BeatInterval** property defines the frequency of heartbeat messages. The **BeatThreshold** property specifies the extra period that the slave probe waits for before switching to active mode. The slave probe caches events until it receives a heartbeat. When it receives a heartbeat, it clears the cache. If the slave does not receive a heartbeat message within the designated time, it forwards the events from the cache to the ObjectServer.

Example configuration

Example properties file values for the master are as follows:

PeerPort: 9999

PeerHost: "slavehost"

Mode: "master"

Example properties file values for the slave are as follows:

PeerPort: 9999

PeerHost: "masterhost"

Mode: "slave"

Example configuration

The required elements to configure peer-to-peer mode are the host name and a port number.



Lesson 3 Remote probe administration



© Copyright IBM Corporation 2016

In this lesson, you learn how to perform the following tasks:

- Learn about the options for remote probe administration
- Configure the probe rules reload feature

Features to facilitate remote administration

Distributed probe with local rules file

- Probe configured to retrieve rules at startup using HTTP or FTP

Local rules file caching

- Used when remote probe retrieves rules file using HTTP or FTP
- Probe caches a local copy of rules file

Bidirectional 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 2016

26

Features to facilitate remote administration

Netcool/OMNIbus probes are generally distributed across one or more remote systems. The distributed operation can present a challenge for the administration of the probes. There are a number of options available to facilitate remote probe administration.

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

Rules file caching

Probes are typically configured to use a local copy of a rules file. When you have multiple probes that are spread across remote servers, local rules files can be a challenge to maintain. Probes can be configured to retrieve a copy of rules from a remote server when the probe starts. With this configuration, you can maintain all rules files on a local server. The local copies are easier to maintain. However, the probe must have access to the file whenever the probe starts. If a network issue prevents access to the file, the probe cannot read the rules.

You can configure the probe to retain a local copy of the rules. With rules file caching configured, the probe starts up, retrieves the rules from a remote server, and saves a copy of the rules locally. If the probe restarts, it attempts to retrieve the rules remotely. If the probe cannot retrieve the rules for some reason, the probe uses the local copy of rules instead.

Bidirectional communication

Provides an HTTP or HTTPS 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

Bidirectional communication

If you change the probe rules file, you must run a command that tells the probe to reread the rules file. With the default probe configuration, you must run the command from the same server that runs the probe. When you configure the probe for remote access, you can run the command from a local server, and cause the probe on a remote server to read its rules file. The access to the remote probe is over HTTP or HTTPS.

Probe registry

Data View												
	RowID	RowSerial	Name	Hostname	PID	Status	HTTP_port	HTTPS_port	StartTime	ProbeType	ConnectionID	LastUpdate
20	20	heart	devtest43.hurley.ibm.com	6981	1	0	0	1349367009	heartbeat	1	1349774684	
24	24	simnet	devtest43.hurley.ibm.com	14222	1	0	0	1349367010	simnet	5	1349774684	
23	23	tivoli_ef	devtest43.hurley.ibm.com	15233	1	0	0	1349436851	tivoli_ef	4	1349774684	
21	21	sim1	devtest43.hurley.ibm.com	14252	1	0	4198	1349367016	simnet	2	1349774684	
22	22	sim2	devtest43.hurley.ibm.com	14295	1	4199	0	1349367013	simnet	3	1349774684	
25	25	heartbeat	devtest43.hurley.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 the ObjectServer

Requires no configuration

Function:

Global actions on all probes

For example: *Flood Control* sample feature

© Copyright IBM Corporation 2016

29

Probe registry

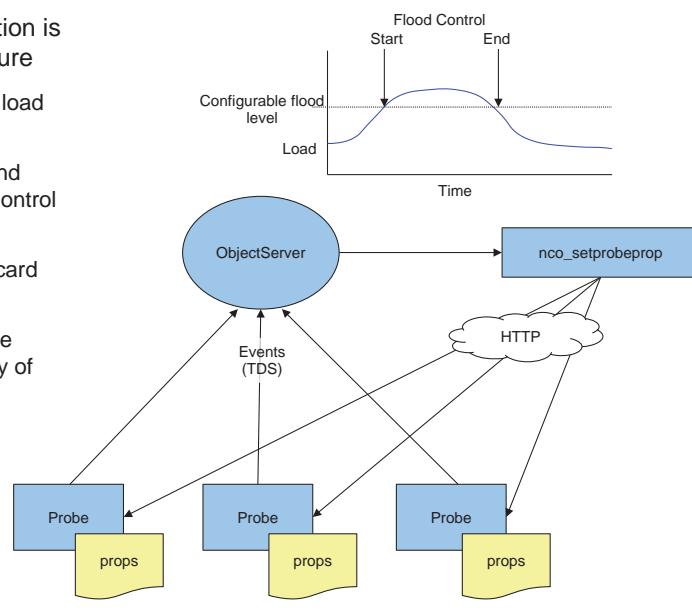
When a probe starts, it connects to an ObjectServer. When the probe connects to the ObjectServer, an entry is created in a table. The entry contains the name of the probe, where probe runs, and whether the probe is configured for remote access. If the probe is configured for remote access, the table contains the designated HTTP or HTTPS port number.

This table is a convenient mechanism to facilitate remote probe administration. A Netcool administrator can examine the table, and determine the host name, and port number for remote access to one or more probes. In addition, a program or trigger can parse the table, and send commands to all probes. Netcool/OMNIbus includes two optional configurations that use the probe registry table to control remote probes.

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_setprobeprop 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 a acceptable level, the trigger resets the FloodControl property of the probes



© Copyright IBM Corporation 2016

30

Bidirectional probe communication usage

One optional configuration is called Flood Control. The configuration includes several ObjectServer triggers. One trigger periodically evaluates some key performance indicators (KPIs) for the ObjectServer. If the trigger determines, based on the KPIs, that the ObjectServer is overloaded, it calls a second trigger. The second trigger reads the entries in the probe registry table. For every probe that is configured for remote access, the trigger sends a command that changes the value of a probe property. The change in property indicates a flood condition. When the probe is in a flood condition, the probe discards low priority events: does not send them to the ObjectServer. The ObjectServer now receives fewer events, and requires fewer resources. The ObjectServer requires fewer resources, and the performance improves. The first trigger, based on the KPIs, determines that the ObjectServer is no longer overloaded. The trigger calls a third trigger. The third trigger sends commands to the remote probes to reset the value of the property. With the property reset, the probe resumes sending all events.

Rereading rules: single probe

Utility to remotely reload a probe rules file without restarting the probe

```
$OMNIHOME/bin/nco_probereloadrules command_line_options
```

Prerequisites:

- The probe must have its HTTP or HTTPS interface enabled

Example:

```
$OMNIHOME/bin/nco_probereloadrules -host <servername> -port 2020
```

The probe processes the reread request only on receipt of a new event

If the probe is idle or is already processing an event, it will not reread the rules file until a new event is received

Rereading rules: single probe

Netcool/OMNibus includes a command that is used to cause a probe to reread its rules file. The command uses the HTTP or HTTPS remote access facility. If a remote probe is configured to read its rules from a local server, the Netcool administrator changes the local copy of the rules file. When the change is complete, the Netcool administrator can use the reload command to cause the remote probe to read the rules file.



Important: Whenever you change a rules file, make certain that you validate the syntax of the modified file with the nco_p_syntax probe. If there is a syntax error in the file, and you instruct the probe to reread the rules, the probe continues to run with the old version of rules. When a probe starts, it caches the rules file in memory. Whenever it rereads a rules file, if it detects an error in the file, it continues to run with the memory copy of rules. If you do not examine the probe log file, you do not know that there is a syntax error.

Rereading rules: all probes

Request all connected probes reload their rules files

Log status to file

```
$OMNIHOME/log/<ObjectServer name>_probemanagement.log
```

Pre-requisites:

- Process activity is running
- Remote management of probes is active
- Probes include \$OMNIHOME/extensions/roi/probewatch.include file
 - Required for reporting status

Rereading rules: all probes

Another optional configuration is called *reload all rules*. This configuration is an automated version of the reload rules command. The configuration uses several ObjectServer procedures. After this configuration is implemented, a Netcool administrator can run a single command, and cause probes to reread their rules files. The prerequisites for the use of this configuration are:

- Probes are configured for remote access
- Probes are configured to use the alternative ProbeWatch rules
- Process activity is running

The alternative ProbeWatch rules are used to capture feedback from the probe. When the probe is configured to use the alternative rules, the probe sends an event to the ObjectServer whenever it rereads its rules file. The event contains the result of the reread operation, either successful or failed.

Rereading rules: configuration

Import SQL file into ObjectServer

```
$OMNIHOME/extensions/roi/probemanagement.sql
```

Creates the following ObjectServer objects:

- Trigger group
- Log file
- Triggers
- Procedures

Rereading rules: configuration

The configuration is contained within an SQL file. To implement the configuration, you import the SQL file into an ObjectServer. The file creates several objects in the ObjectServer, including:

- Triggers: The triggers search the ObjectServer for probe ProbeWatch events. When the triggers find one of these events, the trigger removes the reread status from the event record, and writes the information into a log file.
- Log File: The log file contains the results from all probes whenever the administrator uses the command to reread all rules. The file contains one line for every probes, and a summary for all probes.
- Procedures: The command to reread all rules is a call to an SQL procedure. The SQL procedure calls an external procedure to send the reread rules command to each probe.

Rereading rules: usage

From a command shell execute

```
$OMNIHOME/bin/nco_sql -server <ObjectServer name>
1> execute reloadrules_allprobes
2> go
```

What happens:

1. Procedure iterates through all entries in the probes registry ObjectServer table
2. Calls 'nco_probereloadrules' for each entry with HTTP or HTTPS port
3. nco_probereloadrules issues a *reload rules* request to each probe
4. On success, or failure, each probe sends a ProbeWatch event to the ObjectServer
5. ObjectServer trigger logs ProbeWatch in probemanagement.log

Rereading rules: usage

The reloadrule_allprobes procedure reads the probe registry table. For every probe that is remote accessible, it calls an external procedure. It uses one external procedure for HTTP access, and another one for HTTPS access. The SQL procedure calls the external procedure once for every probe. The external procedures use the probereloadrules command to cause one remote probe to reload its rules. If there are 15 probes, the external procedure is called 15 times: once for each probe.

After the probe rereads its rules file, it generates a ProbeWatch event and sends the event to the ObjectServer. The event contains the status of the reread operation. An ObjectServer trigger finds the event, extracts the status, and writes the information to the log file.

Rereading rules: example

```
Fri May 16 17:25:22 2014 Reload all probe rules request-
Fri May 16 17:25:22 2014 Sent HTTP reload rules request to simnet on
devtest42.hursley.ibm.com:6798
Fri May 16 17:25:22 2014 nnm7 on devtest42.hursley.ibm.com is not running
Fri May 16 17:25:22 2014 nnm_ws on devtest42.hursley.ibm.com is not running
Fri May 16 17:25:22 2014 glf on devtest42.hursley.ibm.com is not running
Fri May 16 17:25:22 2014 Reload all rules summary: managed probes 1, unmanaged probes 0, probes
not running 3
Fri May 16 17:25:24 2014: simnet probe on devtest42.hursley.ibm.com:6798 : Rules file reread
upon HTTP request successful ...
```

Written to the probe management log file

- '\$OMNIHOME/log/<SERVER>_probemanagement.log'
- 4 probes have connected, only 1 running
- Running probe has rules management on
- The reload rules was successful for the running probe

© Copyright IBM Corporation 2016

35

Rereading rules: example

This slide contains a sample of the probe management log file. In this example, the probe registry table indicates that four probes are registered with the ObjectServer. Only one probe is running. After the reload is complete for that probe, the log shows that the reload is successful.

Student exercises



Complete the exercises for this section.

Summary

You now should be able to perform the following tasks:

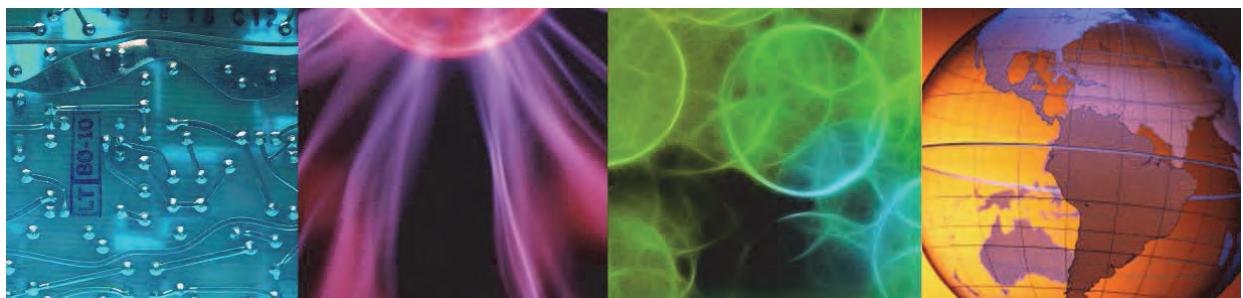
- Use the MIB Manager utility to import a MIB file
- Use the MIB Manager utility to generate rules files
- Incorporate generated rules into the Netcool Knowledge Library
- Use the MIB Manager utility to generate test SNMP trap messages
- Configure probe peer-to-peer mode
- Implement the procedure to reload rules for all active probes



4 Automations



Automations



© Copyright IBM Corporation 2016

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

In this unit you learn about ObjectServer SQL, and how SQL is used in automations.

References: SC27-6265-00 *Netcool/OMNibus Version 8 Release 1 Administration Guide*,
SC27-6505-00 *Web GUI Administration, and User's Guide*

Objectives

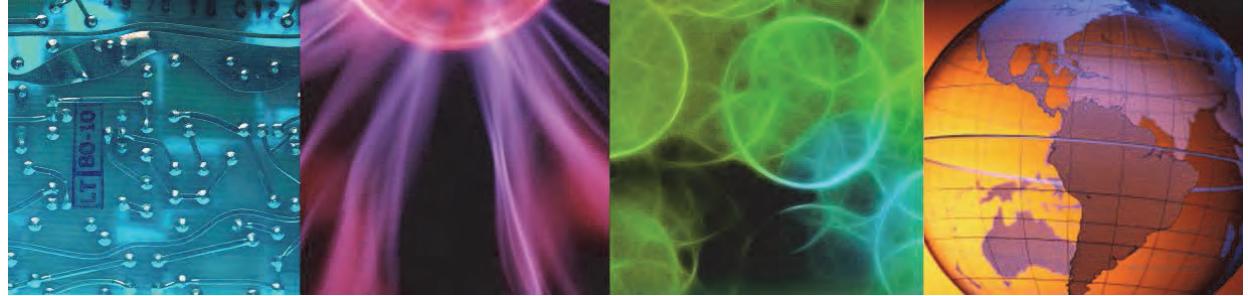
In this unit, you learn to perform the following tasks:

- Use SQL to examine ObjectServer tables
- Use SQL in automations
- Create a database trigger
- Create a temporal trigger
- Create a signal trigger
- Create procedures



Lesson 1 Basic SQL

Lesson 1 Basic SQL



© Copyright IBM Corporation 2016

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 use basic SQL commands.

ObjectServer SQL defined

Subset of ANSI SQL

- Includes some proprietary extensions

Used throughout OMNIbus:

- Automation
- Tools
- Filters

Three general functional areas:

- Data definition: ObjectServer structure and behavior
- Data manipulation: Automations, tools
- System administration: Command line

The focus of this unit is data manipulation

ObjectServer SQL defined

The ObjectServer SQL is a subset of ANSI SQL and is used widely throughout OMNIbus. For example, it is used in the SQL file that creates the ObjectServer and in all automations to interrogate the ObjectServer.

ObjectServer SQL commands can be roughly divided into three functional areas:

- *Data definition commands* define and create databases and tables in the SQL file.
- *Data manipulation commands* are used by automations, tools, and filters to retrieve, modify, and delete data.
- *System administration commands* are used on the command line to manage the system. For example, they are used when shutting down an ObjectServer or changing properties.

This section concentrates on the data manipulation commands, useful for automations and tools.

Starting the SQL interactive interface

To start the SQL interactive interface, run the **nco_sql** command as follows:

- UNIX: `$NCHOME/omnibus/bin/nco_sql -server <ObjectServer> -user <ObjectServer user>`
- Windows: `%NCHOME%\omnibus\bin\isql -S <ObjectServer> -U <ObjectServer user>`

To encrypt a plain text password, enter the following command:

`$NCHOME/omnibus/bin/nco_sql_crypt plaintext_password`

Running SQL commands at the 1> prompt:

- To cancel a command, enter *reset* or press Ctrl+C
- To run the Editor, enter *vi*
- To read in a file, enter *:r file name*
- To run an operating system command, enter two exclamation marks (!!) followed by the command

Starting the SQL interactive interface

To start the SQL interactive interface, run the **nco_sql** command on UNIX and **isql** command on Windows, as follows.

- UNIX:

`$NCHOME/omnibus/bin/nco_sql -server servername -user username`

- Windows:

`%NCHOME%\omnibus\bin\isql -S servername -U username`

In these commands, *servername* is the name of the ObjectServer, and *username* is a valid user name. If you do not specify an ObjectServer name, the default name NCOMS is used. If you do not specify a user name, the default is the user who runs the command. You must enter a valid password for the user, either when prompted or by specifying the **-password** command-line option (-P on Windows).

Encrypting passwords in UNIX nco_sql scripts

You can use the **nco_sql_crypt** utility to encrypt login passwords so that they do not appear in plain text in UNIX scripts that run **nco_sql**. You can do so only when not running in FIPS 140–2 mode. When running in FIPS 140–2 mode, leave the passwords in plain text in the scripts, or use the **nco_aes_crypt** utility with the **-d** option to decrypt any sensitive data before use.

To encrypt and use a plain text password in non-FIPS 140–2 mode, follow these steps:

1. Enter the following command:

```
$NCHOME/omnibus/bin/nco_sql_crypt plaintext_password
```

In this command, *plaintext_password* represents the unencrypted form of the password.

The nco_sql_crypt utility displays an encrypted version of the password.

2. Copy the encrypted password into the script.

The ObjectServer decrypts the password when the connection is made. After connection, you enter ObjectServer SQL commands.

Running SQL commands in the SQL interactive interface

After you connect to the SQL interactive interface with a user name and password, the interface displays a numbered prompt, as shown in the following example:

```
1>
```

Enter ObjectServer SQL commands at the prompt. When you enter text, the following options apply:

- Commands can be split over multiple lines.
- Commands are not processed until you enter the keyword **go** in lowercase letters at the beginning of a new line, and press Enter.

 **Note:** The nco_sql utility does not allow white space to precede the **go** keyword. For example, if you run nco_sql from a script, and use white space to indent the **go** keyword, the SQL statements fail.

- Multiple commands, which are separated by a semicolon, can be run with a single **go** command.
- You can enter a command of up to 4094 characters.
- To cancel a command, enter **Reset** at the beginning of a new line, or press **Ctrl+C** anywhere on a line. Any commands not run are discarded.
- To run the default editor (as defined by the **EDITOR** environment variable) in the nco_sql utility, enter **vi** at the beginning of a new line.
- To read in a file, enter **:r filename** at the beginning of a new line. Do not include the **go** command in the file. Instead, enter the **go** command at the beginning of a new line.
- To run an operating system command, enter two exclamation marks (!!) followed by the command (for example, **!!ls**) at the beginning of a new line.

ObjectServer SQL: selecting events

```
SELECT * FROM alerts.status WHERE Severity = 5;
```

SELECT *	Retrieve all columns
FROM alerts.status	From the alerts database and the status table
WHERE Severity = 5	That meets this condition

© Copyright IBM Corporation 2016

6

ObjectServer SQL: selecting events

Use the SELECT command to retrieve one or more rows, or partial rows of data from an existing table, and to perform grouping functions on the data.

```
select * from database.table where FieldName=expression;
```

select * Select all columns.

from database.table from the database.table

where FieldName=expression that meets the following condition.

Example:

```
select * from alerts.status where Severity=5;
```

Logical operators

You can combine conditions with the operators AND and OR

```
SELECT Summary, Class FROM alerts.status  
WHERE Severity = 5 AND Node = 'batton';
```

```
SELECT * FROM alerts.status  
WHERE Node = 'batton' OR Node = 'catwell';
```

Logical operators

ObjectServer SQL supports the operators AND and OR to build compound conditions.

Comparison operators

> greater than

```
SELECT * FROM alerts.status WHERE Severity > 3;
```

>= greater than or equal to

```
SELECT * FROM alerts.status WHERE Grade >= 3;
```

< less than

```
SELECT * FROM alerts.status WHERE Severity < 5;
```

<= less than equal to

```
SELECT * FROM alerts.status WHERE Grade <= 5;
```

< > Not equal to

```
SELECT * FROM alerts.status WHERE Severity <> 4;
```

Comparison operators

ObjectServer SQL supports the standard comparison operators that are used with numeric values.

Comparison operators: LIKE

Use LIKE for string comparisons

```
SELECT * from alerts.status WHERE Node LIKE '^bat.*';
```

Use LIKE only with character fields

LIKE often used with regular expressions

The NOT keyword inverts the result of the comparison:

```
SELECT * from alerts.status WHERE Node NOT LIKE '.*man$';
```

Comparison operators: LIKE

The LIKE operator is used with character values.

The operators LIKE and NOT LIKE are usually used with regular expressions.

In this example, events are selected from the alerts database and the status table whose Node field value starts with **bat**, and has anything that follows.

Regular expressions

The LIKE operator supports regular expression metacharacters

.	Match any single character
*	Match none or more of the previous characters or character pattern
+	Match one or more of the previous characters or character pattern
[]	Match any of the values in the range [0-9] or [a-z]
^	Match beginning of string
\$	Match end of string
.*	True wildcard

© Copyright IBM Corporation 2016

10

Regular expressions

The LIKE operator supports regular expression matching. Here are the most commonly used metacharacters:

- Match any single character: *link.north* matches *link2north* but not *link21north*.
- Match none or more of the previous characters: *link** matches *lin*, *link*, or *linkkk*.
- Match one or more of the previous characters: *link+* matches *link* or *linkkk*, but not *lin* or *linxk*.
- Match any single character in the specified range: *link[0-5]* matches *link2* but not *link9*.
- Ensure that the pattern matches at the beginning of the string: *9link.** matches *linknorth* but not *northlink*.
- Ensure that the pattern matches at the end of the string: *.*link\$* matches *northlink* but not *linknorth*.

Within a regular expression, you can use the backslash (\) character before any special character to ensure that the literal value is matched.

More comparison operators

IN compares a value to a list of values

```
SELECT * FROM alerts.status WHERE Node IN ('batton', 'catwell', 'robel');
```

The **NOT** keyword inverts the result of the comparison

You can also use **IN** with subqueries:

```
SELECT * FROM alerts.status WHERE Serial IN (SELECT Serial FROM alerts.journal);
```

- This example retrieves all events that have a journal entry
- Use **IN ()** lists instead of the **OR** operator, particularly if you have more than 10 items to search

More comparison operators

You can use the **IN** list comparison operator to compare 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.

IN() searches scale logarithmically, whereas using an **OR** scales in linear time. For example, if you have 100 items to search, the worst case with an **IN()** is $(\text{base } 2) \log(100)$. The worst case if you have 100 items with a compound expression with an **OR** is 100. The item that you are trying to match is at the end of the entire compound expression.

Use **IN()** lists if you have more than 10 items you are searching. **IN()** lists require processor usage to create.

Modifying data with UPDATE

Use the UPDATE command to change table data

- Single field:

```
UPDATE alerts.status SET Severity = 5 WHERE Severity = 4 AND Acknowledged = 0;
```

- Multiple field:

```
UPDATE alerts.status SET Severity = 5, Service = 'Web Hosting' WHERE Severity = 4 AND Customer like 'STAR ISP';
```

The WHERE clause controls which rows are updated

© Copyright IBM Corporation 2016

12

Modifying data with UPDATE

Use the UPDATE command to update one or more columns in an existing row of data in a table.

```
UPDATE database.table SET <ColumnName=Value, [ColumnName=Value, ... ] WHERE <condition>;
```

UPDATE

Update the records.

SET

Assign values to column names. Assignments are typed, meaning that you cannot assign an integer value to a column typed as Char.

WHERE

Apply a condition to the update command. If <condition> is true, the update is performed. A condition can consist of a subselect as follows:

```
update alerts.status set Summary = Summary + ' additional info' where Grade in (select Grade from alerts.some_other_table where UpdateSummary = true);
```

Subselects are evaluated first.

Often an update is used with a **via()** expression. Here is an example:

```
update alerts.status via 'my_primary_key' set Severity=4, Grade=99, Summary=Summary + 'concatenated with original Summary';
```

The **via()** expression is powerful because a full table scan is not required to perform the update. Therefore, using **via()** produces SQL that is considerably more scalable. You *must* know the primary key to use **via()**.

Creating data with INSERT

Use the INSERT INTO command to enter new rows into a table

```
INSERT INTO alerts.status
(Node, Severity, Summary, Identifier)
VALUES
('batton', 5, 'Disk Usage 95%', 'battonDU')
```

Must include the Identifier column

- Identifier is the event record key

© Copyright IBM Corporation 2016

13

Creating data with INSERT

Use the INSERT command to create a new row of data in an existing table. If you do not insert values for every column in the row, you must specify a comma-separated list of columns. The list of columns is enclosed in parentheses, followed by the VALUES keyword, followed by a comma-separated list of values within parentheses, for example:

```
INSERT INTO database.table <Column_List> VALUES <Value_List> [updating
(Column)];
```

Position is important in the *Column_List*, and in the *Value_List*. Position 1 in the Column_List is assigned the value in position 1 of the Values_List. More generally, position *n* in the Column_List is assigned the value of position *n* in the Value_List.

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. The UPDATING keyword is typically used only by probes, for example:

```
insert into alerts.status (Identifier, Severity, Summary) values
('my_primary_key', 4, 'my_summary_information');
```

Removing data with DELETE

Use the DELETE command to delete one or more rows of data from a table

```
DELETE FROM alerts.status WHERE Severity=0;
```

- In this example, all events with a Severity of *Clear* are deleted
- You cannot recover deleted events

Removing data with DELETE

This statement deletes the rows from the table that meet the specified condition. The same *<condition>* rules apply as for a SELECT or UPDATE. Here is an example of the DELETE statement:

```
DELETE FROM database.table WHERE <condition>;
```

Note that you can also use **via()**:

```
delete from alerts.status via 'my_primary_key' ;
```

ORDER BY clause

The ORDER BY clause places the selected events in a defined order

```
SELECT * FROM alerts.status
WHERE AlertGroup='link'
ORDER BY Node ASC;
```

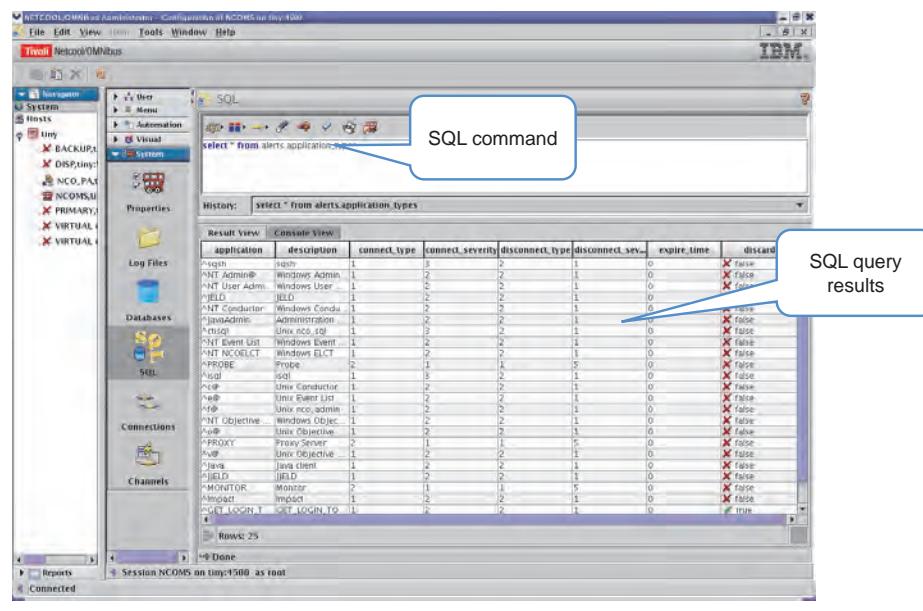
© Copyright IBM Corporation 2016

15

ORDER BY clause

The ORDERBY clause causes the selected events to be displayed in the defined order. The syntax is the name of the field and then the ordering option, either *desc* (descending) or *asc* (ascending).

SQL workbench



16

SQL workbench

You can use the Netcool/OMNIbus Administrator utility GUI to run SQL commands in the ObjectServer. The utility has SQL builder buttons to help construct SQL statements. The output is displayed in a formatted table result set. There is also a **History** menu list of previously used commands.



Important: You must have the correct role to enable access to the ISQL interface.



Lesson 2 ObjectServer automations



© Copyright IBM Corporation 2016

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

In this lesson, you learn how automations are used in the ObjectServer.

Trigger applications

Triggers are used for the following tasks:

- Automate management of events
- Perform external commands automatically on receipt of certain events
- Incorporate escalation procedures
- Correlate events

Trigger applications

Triggers are a way to respond to events that happen within the ObjectServer. You can perform these tasks, among others:

- Perform correlations
- Perform automatic suppression
- Flag alarms for ticket systems
- Implement various N-to-N relationships among alarms
- Run external scripts
- Respond to changes made from other Tivoli/Netcool products that are connected to the ObjectServer

Trigger types

There are three types of triggers:

- Database
A database condition exists in ObjectServer
- Temporal
This trigger runs on a timed basis
- Signal
A system or user-defined signal was raised

© Copyright IBM Corporation 2016

19

Trigger types

There are three types of triggers.

- Database: A database trigger activates based on a change to a database table. The trigger defines the name of the database table, and what type change activates the trigger.
- Temporal: A temporal trigger is activated based on a frequency. For example, a temporal trigger can activate once every minute.
- Signal: A signal trigger activates based on the occurrence of a special event, which is called a signal. The ObjectServer includes a collection of system signals. An example of a system signal is the connect signal. The connect signal is raised when a component connects to the ObjectServer. You can define a user signal. When you create a user signal, you define the conditions that are related to the signal. The ObjectServer raises system signals automatically. A user signal is typically raised manually with a desktop tool.

Trigger general settings

- Trigger name
- Group
 - Trigger group
- When
 - Valid SQL condition
- Action, which can be one or more of these types:
 - SQL action
 - SQL procedure
 - External procedure, which requires process control
- Comment
 - Description of trigger
- Declare
 - Local variables

© Copyright IBM Corporation 2016

20

Trigger general settings

When you create a trigger, you must configure some settings. Some of the setting values are common across all trigger types. Some triggers do not use some settings.

Trigger names are character strings. A trigger name cannot contain spaces or special characters except for the underscore. The trigger name must be unique.

Triggers are organized into trigger groups. A trigger group has a name, and the name has the same constraints as a trigger name. A trigger group can be used to control the activation of multiple triggers. If you disable a trigger group, you prevent the activation of all triggers that belong to that group.

The WHEN setting is used to configure an optional condition that must be met before a trigger can activate. The condition that is defined in the WHEN setting is in addition to the type of trigger. For example, you create a temporal trigger with a frequency of every hour. The trigger activates every hour. If you create a WHEN condition, the trigger runs every hour only if the WHEN condition is met.

The ACTION setting contains the commands that run when the trigger is activated. The commands are typically one or more SQL commands.

More trigger general settings

- State
 - Debug
Logging to message log (ObjectServer log file)
 - Enabled
Enable or disable trigger
Both the trigger and trigger group must be enabled for the trigger to fire
- Priority
 - Set trigger fire order (1 – 20), 1 being highest
Trigger types are checked constantly for a TRUE condition
The PRIORITY determines the fire order when one or more triggers of a particular type are TRUE

© Copyright IBM Corporation 2016

21

More trigger general settings

The STATE setting is used to control other aspects of the trigger. For example, you can enable or disable a trigger. If you disable a trigger, the trigger does not activate even if the trigger group is enabled. The STATE setting is also used to control the creation of a log file, the generation of debug messages.

If you have several temporal triggers that you configure with the same frequency, the PRIORITY setting determines the order in which they activate.

Trigger groups

All triggers are part of a trigger group

- Use trigger groups to manage multiple triggers

For example, you can disable or enable multiple triggers if all are in the same group

- You create them on the **Trigger Group** tab

From the GUI, you can manage and group related triggers

- You can drop (delete) trigger groups only if they are empty

- You can move triggers between groups

Trigger groups

Trigger groups are used to organize, and control one or more triggers. If you have five triggers that are assigned to the same trigger group, you can disable the group, and the five triggers no longer activate. The triggers must be enabled or they do not activate even if the trigger group is enabled.

WHEN clause

You can determine when the trigger fires:

- Day of week
- Time of day
- Severity at a certain level:
`new.Severity>3`
- Deduplication period is below a certain amount:
`(new.LastOccurrence – new.FirstOccurrence) < 60`

WHEN clause

The WHEN setting is used to define a condition that must be met before a trigger activates. A user might create a trigger to automatically delete certain events from the ObjectServer. The user creates a temporal trigger that runs once every hour. After the trigger is enabled, the trigger activates every hour, on every day of the week. The user might not want the trigger to remove events on Saturday or Sunday. The user can add a WHEN condition to the trigger to test for the day of the week. In the WHEN condition, the user specifies that the trigger does not activate on Saturday or Sunday. After the WHEN condition is added, the trigger activates once every hour on every day of the week except Saturday or Sunday.

Example of a WHEN clause

```
create or replace trigger expire
group default_triggers
priority 1
comment 'Expiration'
every 60 seconds
evaluate
  select Identifier, ExpireTime from alerts.status where ExpireTime > 0 and Severity > 0 bind
as expires
when
  -- it is not saturday or sunday
  (dayofweek( getdate() ) <> 7) and (dayofweek( getdate() ) <> 1)
begin
  for each row expire in expires
  begin
    update alerts.status via expire.Identifier set Severity = 0 where LastOccurrence <
    (getdate() - expire.ExpireTime);
  end;
end;
```

© Copyright IBM Corporation 2016

24

Example of a WHEN clause

This slide contains an example of the use of a WHEN condition. The example is a temporal trigger that activates every 60 seconds. The WHEN condition uses a special function to determine the day of the week. The function returns an integer based on the day of the week. A value of 7 indicates Saturday, and a value of 1 indicates Sunday.

When the temporal trigger activates, the when condition is evaluated. If the day of the week is not 7, and not 1, the command runs. If the value is 7, or 1, the command does not run.

Trigger actions

Trigger actions can be procedural SQL code blocks

They can run a predefined procedure:

- Run an SQL procedure:

EXECUTE PROCEDURE

- External executable procedures:

External executable procedures require Process Control running on the host that runs the procedure

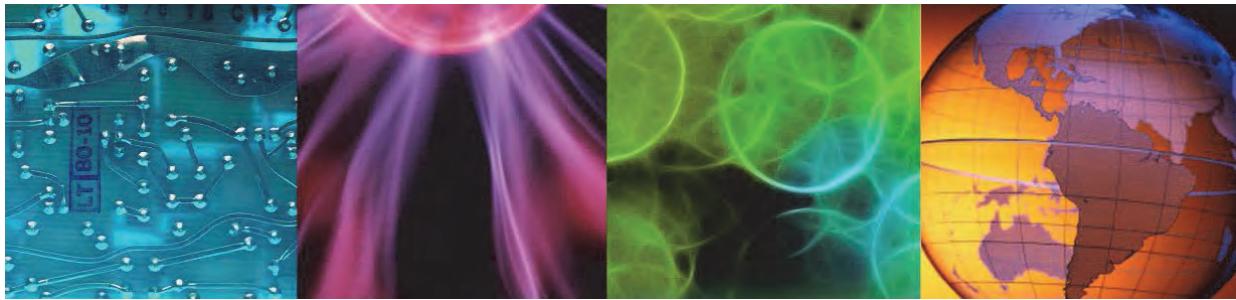
Trigger actions

The ACTION contains the commands that run when the trigger activates. The ACTION can contain a single SQL command, a block of commands, or a command that runs a procedure.

Lesson 3 SQL code blocks



Lesson 3 SQL code blocks



© Copyright IBM Corporation 2016

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 and use of SQL code blocks.

SQL code blocks

SQL code blocks are the body of any trigger or stored procedure

All code blocks share common characteristics

The **Action** tab of a trigger (or the body of a procedure) is enclosed within the keywords BEGIN and END

A semicolon terminates each SQL statement, expression, loop, or conditional

A semicolon also terminates the code block

SQL code blocks

SQL code blocks are *segments* of SQL that are enclosed in the keywords BEGIN, and END; (semicolon included). An SQL code block is the fundamental expression that is used within trigger actions, and within procedures.

SQL commands that are commonly used

A body statement can contain one or more SQL commands:

```
ALTER <ObjectServer Object>
UPDATE
INSERT
DELETE
WRITE INTO
RAISE SIGNAL
{ EXECUTE | CALL } PROCEDURE
CANCEL
BREAK
IF / THEN / ELSEIF
CASE / WHEN
FOR / FOR EACH ROW
SET <TypedObject> = <TypedValue>
```

SQL commands that are commonly used

This slide contains a list of some of the SQL commands that you can use within an SQL code block.

SQL code blocks: %user variables

With code blocks, you can access certain contextual data

Use %user variables to determine who is running the block

%user.variable (the current client information)

```
%user.userid  
%user.user_name  
%user.app_name  
%user.host_name  
%user.connection_id  
%user.is_auto  
%user.is_gateway  
%user.is_eventlist  
%user.description
```

© Copyright IBM Corporation 2016

29

SQL code blocks: %user variables

Information is available for use within an SQL code block. The ObjectServer provides a set of system variables. The percent (%) character precedes all system variables. This slide shows an example of the useful %user variable.

The Netcool/OMNIbus Administrator utility includes a *helper* feature. When you click the yellow % button, you see a complete of list of available variables.

SQL code blocks: SET statement

Use a SET assignment statement to write the value of an expression to a variable or parameter

Often used within a database trigger or within a FOR EACH ROW loop

Syntax:

```
SET { parameter_name | variable_name } = expression
```

An expression might be one of the following items:

- A quoted string ('Node XB1')
- A number (9)
- A column name (Severity)
- An ObjectServer property (Server Name)
get_prop_value ('Name')
- An environment variable (OMNIHOME)
getenv ('OMNIHOME')
- A variable that holds a temporary value in a procedure or a trigger

SQL code blocks: SET statement

SET is a keyword that makes assignment. The value of the right side of the equals (=) operator is to the variable on the left side of the = operator.

As with everything in the ObjectServer SQL language, column type is enforced. You cannot assign a string value to an integer column type, for instance.

Example of a SET statement

```
BEGIN
    IF <some condition> THEN
        SET Summary = Summary + ':' + %user.user_name;
    END IF;
END
```

© Copyright IBM Corporation 2016

31

Example of a SET statement

SQL code blocks: IF THEN ELSE

The IF THEN ELSE statement performs one or more actions based on the specified conditions

Syntax:

```
IF condition THEN
    action_command_list
    [ELSEIF condition THEN optional
        action_command_list
    ...
    ]
[ELSE optional
    action_command_list]
END IF;
```

SQL code blocks: IF THEN ELSE

ObjectServer SQL is used to implement logical controls within any code block. The most common way is by using an IF() statement.

For instance, you might want to implement the logic that if Grade is 99, do one thing; else if Grade is 98, do something else. You can use IF() blocks for this task.

SQL Code Blocks: CASE/WHEN Statement

The CASE WHEN statement performs one or more actions based on a condition. If the condition is not met, you can optionally perform a different action.

Syntax

```
CASE
WHEN condition1 THEN action_command_list1
WHEN condition2 THEN action_command_list2    optional
...
ELSE action_command_list      optional
END CASE;
```

© Copyright IBM Corporation 2016

33

SQL Code Blocks: CASE/WHEN Statement

The CASE / WHEN conditional is more similar to an IF / THEN / ELSEIF conditional than it is to a traditional CASE / SWITCH statement that you might find in other languages. The case block starts with the keyword CASE (no arguments), and ends with the keyword END CASE; (semicolon included). A block of SQL test follows each WHEN(<**condition**>). You do not need to use the begin and end keywords in this block. You do not have to explicitly break from each WHEN condition, as you would normally do in a SWITCH/CASE block.

You use an ELSE statement as the final (or default) condition for the entire case block. Usually, CASE/WHEN is preferred instead of long IF/THEN/ELSEIF blocks because it is much easier to read.

SQL code blocks: FOR EACH ROW

The FOR EACH ROW loop performs actions on a set of rows that match a certain condition

Syntax:

```
FOR EACH ROW variable_name in database_name.table_name  
[ WHERE condition ]  
BEGIN  
    action_command_list;  
END ;
```

SQL code blocks: FOR EACH ROW

You can use the FOR EACH ROW loop to perform actions on a set of rows that match a certain condition. The following example increases the severity of all alerts in the alerts.status table that have a severity of 3 to a severity of 4.

```
FOR EACH ROW alert_row in alerts.status WHERE alert_row.Severity=3  
BEGIN  
    SET alert_row.Severity = 4;  
END;
```



Note: Generally speaking, use of the EVALUATE clause is relatively inefficient and its use should be avoided whenever possible. In most cases, you can replace an EVALUATE clause with a FOR EACH ROW clause which cursors over the data and does not incur the processor usage of creating a temporary table.

SQL code blocks: Example of FOR EACH ROW

Example:

```
FOR EACH ROW alert_row in alerts.status WHERE alert_row.Severity=3
BEGIN
    SET alert_row.Severity = 4;
END;
```

© Copyright IBM Corporation 2016

35

SQL code blocks: Example of FOR EACH ROW

When this statement runs, the ObjectServer reads each row of the alerts.status table, and tests to see whether the value in the Severity column is 3. For each row that matches this condition, the statements within the BEGIN and END are run until all the rows are processed.

SQL code blocks: FOR

The FOR loop performs actions a set number of times, based on a counter variable

Syntax:

```
FOR counter = 1 to <integer> DO  
BEGIN  
    action_command_list;  
END;
```

- To loop 10 times, use integer = 10
- The BREAK statement in the action command list exits the loop
- The CANCEL statement in the action command list stops the procedure

SQL code blocks: FOR

ObjectServer SQL also allows for counter-controlled looping. You typically do not configure looping inside a trigger. You typically configure this type of looping within a procedure.

SQL code blocks: Example of FOR

This example also includes a Declare block, which always precedes a code block

```
DECLARE <variable_name> <variable_type>;
```

```
Declare
  x integer;
BEGIN
  FOR x=1 TO 1000
    begin
      insert into alerts.status (Node, Identifier, Severity)
      VALUES ('TEST', to_char(x), 5);
    end;
END
```

© Copyright IBM Corporation 2016

37

SQL code blocks: Example of FOR

This example runs the INSERT command 1000 times.

Trigger response sequence

An automation, or trigger, fires when certain conditions occur in the ObjectServer:

- Time
- Database action
- Signal

The WHEN clause determines whether the action should run:

- Condition to meet before running the action
- Can determine time of day or day of week first

The EVALUATE statement builds a read-only temporary table (not for database automations) to be used in the action

- Use of the EVALUATE clause is relatively inefficient and its use should be avoided whenever possible
- In most cases, you can replace an EVALUATE clause with a FOR EACH ROW clause

Action runs the SQL code block

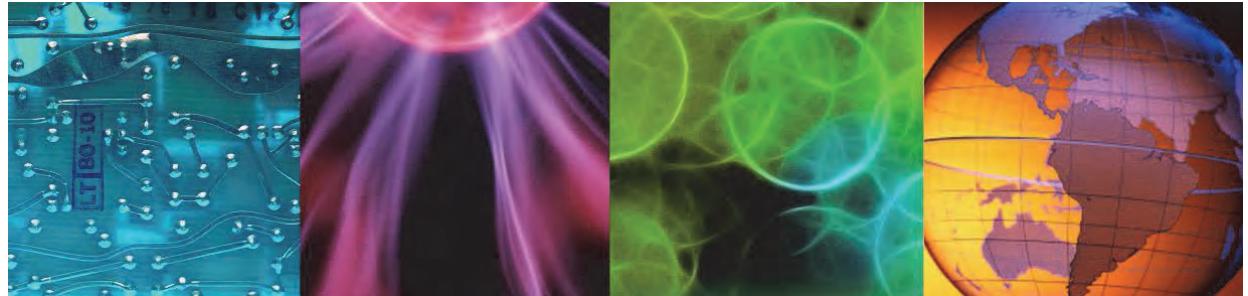
Trigger response sequence

In summary, triggers activate based on three types of conditions: time, database change, and signal. When the trigger activates, an optional WHEN clause is evaluated. If the WHEN condition is met, the trigger continues. The optional EVALUATE setting creates a temporary table that contains records that meet some condition. The ACTION setting contains the commands that run.



Lesson 4 Database triggers

Lesson 4 Database triggers

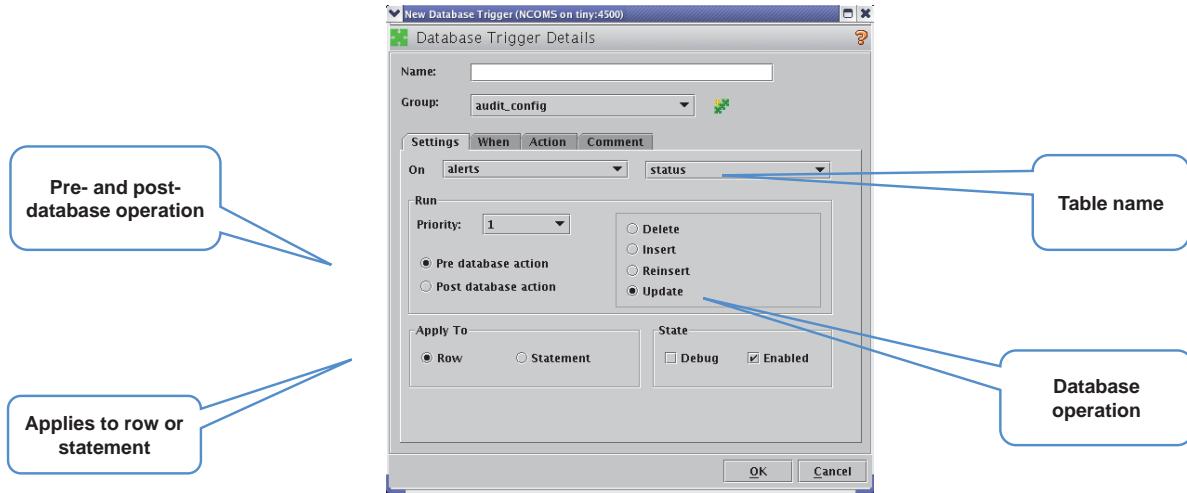


© Copyright IBM Corporation 2016

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 a database trigger.

Database trigger



© Copyright IBM Corporation 2015

40

Database trigger

This slide contains a screen capture of the Netcool/OMNIbus Administrator utility. The screen capture shows the feature that is used to create a database trigger.

Database trigger: Settings and action

Settings

- Table name
- Pre- and post-action on insert, update, delete, or reinsert
- Apply to row or statement

Action

- Fired whenever the condition is TRUE:
 - **Row:** Fired against each row that matches the condition
 - **Statement:** Fires only once, regardless of number of matching rows

© Copyright IBM Corporation 2016

41

Database trigger: Settings and action

Complete the **Settings** tab as follows:

On: Select the ObjectServer database and the database table that cause the trigger to activate.

Priority: Select a priority that determines the order in which the ObjectServer fires triggers when this database modification causes more than one trigger to activate. You can select numbers 1 - 20, with 1 being the highest priority.

Pre database action/Post database action: Click **Pre database action** to indicate that the trigger action should run before the database modification occurs. Click **Post database action** to indicate that the trigger action should run after the database modification occurs.

Delete/Insert/Reinsert/Update: Use these options to specify the type of database modification that activates the trigger.

Row/Statement: Click **Row** to set the trigger to activate once for each row that matches the trigger condition. Click **Statement** to set the trigger to activate once regardless of the number of matched rows in the table.

Debug: Select this option to send debugging information to the ObjectServer message log each time the trigger activates.

Enabled: Select this check box to enable the trigger for use. A disabled trigger does not activate when the associated database modification occurs.

Special row variables: Overview

Used in a row-level trigger

- **Old** variable refers to the value of a column before an incident is raised
`old.Severity`
- **New** variable refers to the value of a column that is affected by the incident
`new.Severity`

Special row variables: Overview

To change individual column values for an insert or update, you need a way to compare the existing column value (the one stored in your table) to the incoming column value. The ObjectServer provides you two row-level variables through which you can make column-level data assignments (or comparisons). The two variables are as follows:

- **old** refers to the current row in the table.
- **new** refers to the incoming row (assumes that you are working with an insert or update).

Special row variables for operation and timing

Operation	TimingMode	Is the New Variable Available?	Is the New Variable Modifiable?	Is the Old Variable Available?	Is the Old Variable Modifiable?
INSERT	BEFORE	Y	Y	N	N
INSERT	AFTER	Y	N	N	N
UPDATE	BEFORE	Y	Y	Y	N
UPDATE	AFTER	Y	N	N	N
DELETE	BEFORE	N	N	Y	N
DELETE	AFTER	N	N	Y	N
REINSERT	BEFORE	Y	N	Y	Y
REINSERT	AFTER	Y	N	N	N

Special row variables for operation and timing

This table contains a summary of the availability of the *new*, and *old* variables based on the type of database operation. For example, with a database INSERT operation, the *new* variables exists. However, the *old* variables do not exist because there is no record already in the table.

Pre-levels and post-levels

Sequence for pre-level and post-level database triggers

- A PRE statement fires after the ObjectServer write lock is obtained but before the row is formally changed
- A POST statement fires after the ObjectServer write lock is obtained and after the row is formally changed
- Database triggers fire only once for each specific database event

Deduplication automation action

For each row on reinsert:

```
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
            set old.Severity = new.Severity;
        end if;
End;
```

© Copyright IBM Corporation 2016

45

Deduplication automation action

This slide shows the ACTION definition for the deduplication trigger. The deduplication trigger activates based on a database REINSERT. The REINSERT is proprietary to the ObjectServer. The REINSERT results when an INSERT attempts to create a record in the ObjectServer, and a record with the same key exists. In other database management systems, this situation causes an error. The error is typically something like UNIQUE KEY CONSTRAINT VIOLATION.

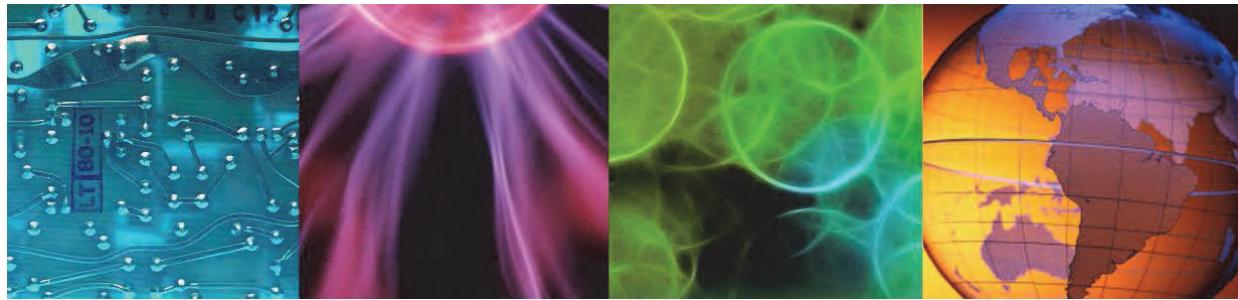
When the REINSERT is detected, the deduplication trigger activates before the database changes. When the trigger activates, a record exists in the ObjectServer, so the *OLD* references relate to the existing values. The *NEW* references relate to the values that are contained in the INSERT statement.



Lesson 5 Temporal triggers



Lesson 5 Temporal triggers

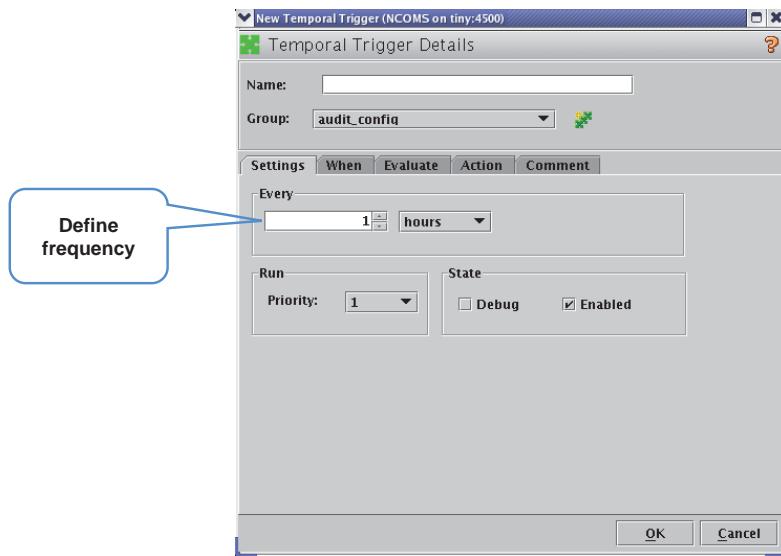


© Copyright IBM Corporation 2016

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 a temporal trigger.

Temporal trigger



© Copyright IBM Corporation 2016

47

Temporal trigger

A temporal trigger activates based on a time frequency. Do not confuse time in this case with time of day. The frequency defines how frequently the trigger activates, not when it activates. For example, a temporal trigger with a frequency of 1 hour activates every hour. The trigger might not activate on even hour boundaries, for example, 8:00, 9:00, 10:00. The activation is based on when the ObjectServer starts. After the ObjectServer starts, the trigger activates every hour.

Temporal trigger overview

WHEN

- A condition that is not dependent on the underlying table data
(for example, time of day, day of month)

Evaluate

- Accepts a temporary table that you can create
- Temporary table passes to the subsequent action
- Must be within the same trigger instance

Action

- The action is fired whenever the defined frequency and condition (when) are TRUE
- An example of a temporal trigger is mail_on_critical

Temporal trigger overview

The optional WHEN is used to define a condition that must be met before the ACTION occurs.

The EVALUATE setting is used to optionally create a temporary table before the ACTION occurs.

The ACTION setting contains the command or commands that run when the trigger activates. The commands run if the optional WHEN condition is satisfied.

Temporal trigger example: mail_on_critical

Action

```
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
```

Note: the trigger uses FOR EACH ROW instead of a separate EVALUATE

© Copyright IBM Corporation 2016

49

Temporal trigger example: mail_on_critical

This slide shows the ACTION definition for the mail_on_critical trigger. The mail_on_critical trigger is included with Netcool/OMNIbus. This temporal trigger uses a WHERE clause to locate any event record that meets the following conditions:

- Severity = 5, which denotes a critical event.
- Acknowledged = 0, which means that a user did not acknowledge the event.
- LastOccurrence <= getdate() - (60*30), which indicates that the event is more than 30 minutes old.
- Grade < 2, which is used to control subsequent actions.

If the WHERE clause locates an event record that meets the conditions, the trigger calls a procedure:

```
execute send_mail
```

The trigger passes various parameters to the send_mail procedure. The send_mail procedure causes a command to run on a server, which is defined as *localhost* in this example. The command generates an email message and sends the email to an email address, which is defined as *root@localhost* in this example.

The send_mail procedure also updates the corresponding event record, and sets Grade = 2. The value of 2 causes the mail_on_critical trigger to ignore this event the next time the trigger activates.

Event correlation: GenericClear

- Correlate problem events with resolution events
- Ensure that the event is from the same source:
 - Probe
Manager field
 - Node
Node field
 - Interface
AlertKey
 - Category
AlertGroup
 - Problem or resolution
Type field
 - Severity
Severity field

© Copyright IBM Corporation 2016

50

Event correlation: GenericClear

In many environments, events occur that indicate a *problem*, like a network link down. Other events occur that indicate a *resolution* to the problem, like a network link up. Netcool/OMNibus includes a trigger that is called generic_clear. The trigger correlates *problem*, and *resolution* event records that are related to the same situation, like the same network link. When the trigger correlates the problem and resolution event records, the trigger updates the Severity column for both records. The trigger sets the severity to 0, which indicates a clear condition.

The trigger uses several event record columns to correlate the *problem* and *resolution* event records. The trigger uses the following columns:

- Node: The problem and resolution must be from the same device.
- AlertGroup: This column identifies the type of situation. For example, a link issue or a port issue.
- AlertKey: This column provides more detail that is related to the type of situation. For example, for a link issue, AlertKey identifies the specific link.
- Manager: This column identifies the probe that created the event records.

Based on the values in these columns, the trigger ensures that the problem and resolution are related to the same device, and the same issue.

GenericClear automation

Temporal trigger

Clears all rows in alerts.status table where **Type = 1** and there is a subsequent row which indicates that the problem is resolved (**Type = 2**)

Uses a virtual table space, alerts.problem_events, which is cleared each time that the automation runs

GenericClear automation

The generic_clear trigger identifies a problem event based on a Type of 2, and a resolution as a Type of 1. The trigger is called *generic* clear because it is not specific to any particular type of issue.

GenericClear

```
begin  
-- Populate a table with Type 1 events corresponding to any uncleared Type 2 events  
  
for each row problem in alerts.status where problem.Type = 1 and problem.Severity > 0 and (problem.Node + problem.AlertKey  
+ problem.AlertGroup + problem.Manager) in ( select Node + AlertKey + AlertGroup + Manager from alerts.status where  
Severity > 0 and Type = 2 )  
begin  
insert into alerts.problem_events values ( problem.Identifier, problem.LastOccurrence, problem.AlertKey,  
problem.AlertGroup, problem.Node, problem.Manager, false );  
end;  
  
-- For each resolution event, mark the corresponding problem_events entry as resolved and clear the resolution for each row  
resolution in alerts.status where resolution.Type = 2 and resolution.Severity > 0  
begin  
set resolution.Severity = 0;  
update alerts.problem_events set Resolved = true where LastOccurrence < resolution.LastOccurrence and Manager =  
resolution.Manager and Node = resolution.Node and AlertKey = resolution.AlertKey and  
AlertGroup = resolution.AlertGroup ;  
end;  
  
-- Clear the resolved events for each row problem in alerts.problem_events where problem.Resolved = true  
begin  
update alerts.status via problem.Identifier set Severity = 0; end;  
-- Remove all entries from the problems table delete from alerts.problem_events;  
end
```

© Copyright IBM Corporation 2016

52

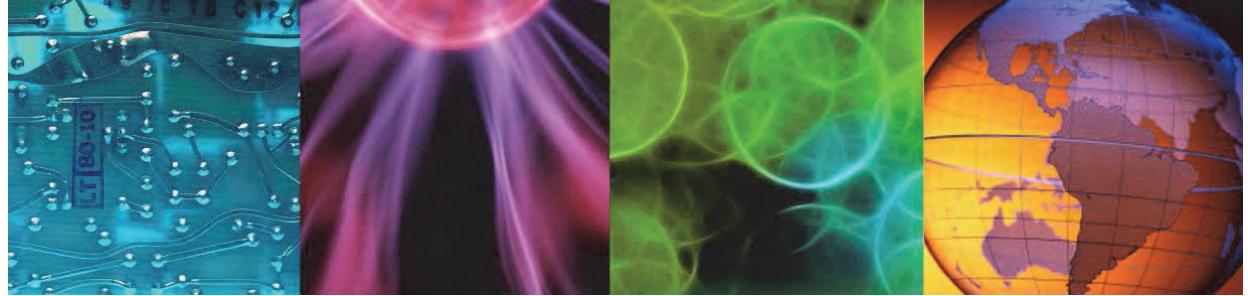
GenericClear

This slide shows the contents of the ACTION section for the generic clear trigger.



Lesson 6 Signal triggers

Lesson 6 Signal triggers



© Copyright IBM Corporation 2016

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 a signal trigger.

Signal trigger

There are two types of triggers:

- System
 - Raised spontaneously by the ObjectServer on changes to the system
- User
 - Created by a user with the GUI or command line
 - Can be raised within an action body of a trigger
 - Allows multiple actions against one trigger

Signal trigger

The ObjectServer includes several system triggers. System triggers are raised automatically based on some condition. For example, the connect system signal is raised when a component connects to the ObjectServer. A disconnect signal is raised when a component disconnects from the ObjectServer. If a signal trigger is configured based on the connect signal, the trigger activates when a component connects to the ObjectServer.

You can create user signals. For user signals, the signal must be raised in some fashion. The signal can be raised with a desktop tool, or from within another trigger. For example, a database trigger can be configured to activate based on a DELETE to the alerts.status table. In the ACTION section of the database trigger, you can configure a statement to RAISE a user signal.

System signals

Raised automatically by the ObjectServer on changes to the system

The screenshot shows the 'Configuration of NYC_AGG_P on host1.tivoli.edu:4100' window. The left sidebar has a tree view with 'System' selected, showing 'Properties', 'Log Files', 'Databases', and 'SQL'. The main area is titled 'Databases, Tables and Columns' and displays the 'catalog' schema. Under 'catalog', 'primitive_signals' is selected. On the right, a 'Data View' table is shown with the following data:

SignalName	IsSystem	Comment
alter_object	true	ALTER OBJECT
alter_property	true	ALTER PROPE
backup_failed	true	ALTER SYSTEM
backup_succeeded	true	ALTER SYSTEM
connect	true	Client connect
create_object	true	CREATE OBJE
disconnect	true	Client disconn
drop_object	true	DROP OBJECT
gw_counterpart_down	true	Counterpart h
gw_counterpart_up	true	Counterpart h
gw_resync_finish	true	Gateway Obj
gw_resync_start	true	Gateway Obj
iduc_connect	true	IDUC client co
iduc_data_fetch	true	IDUC data fetc
iduc_disconnect	true	IDUC client dis
iduc_missed	true	IDUC missed e
license_lost	true	ObjectServer l
login_failed	true	Client login fail
permission_denied	true	Permission de
profiler_report	true	ObjectServer
profiler_toggle	true	ObjectServer
resync_lock	true	ObjectServer

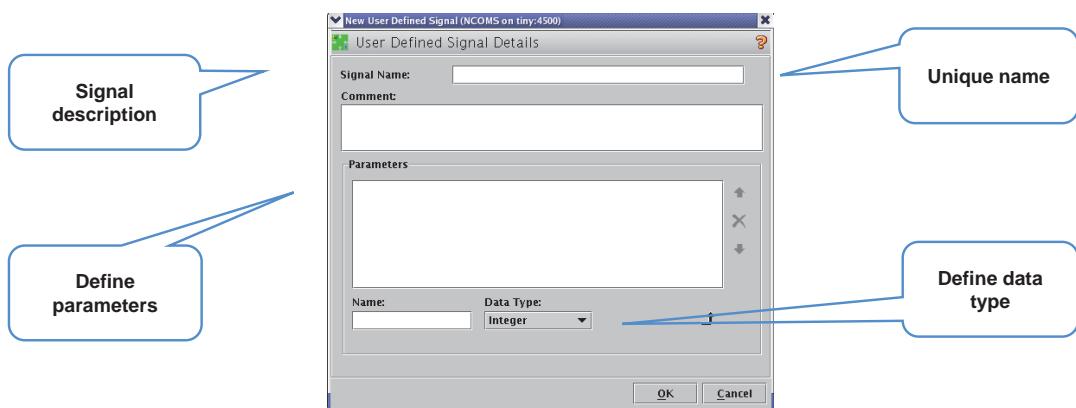
© Copyright IBM Corporation 2016

55

System signals

The list of system signal names is found in the catalog.primitive_signals table. Each system trigger contains one or more parameters. When the signal is raised, the parameters are populated automatically.

User-defined signals



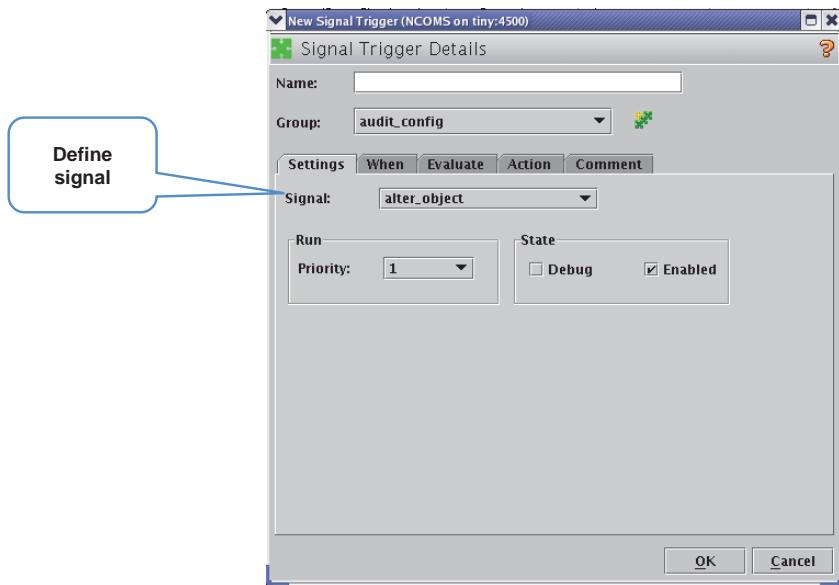
© Copyright IBM Corporation 2016

56

User-defined signals

A user-defined signal requires a name, and one or more parameter definitions. The parameter definition contains a name, and type. The parameters are related to the signal, and are available to any trigger that references the signal. With a user-defined signal, the process that raises the signal must provide the values for any parameters.

Signal trigger



© Copyright IBM Corporation 2016

57

Signal trigger

A signal trigger activates based on a signal. The signal can be a system, or user-defined signal. The ACTION section is the same format as the temporal or database triggers.

Raising and dropping signals

Explicitly created, raised, and dropped

- Raise to activate signal:

```
RAISE SIGNAL <signal_name> [(Arg1, ... ArgN)] ;
```

- Drop to delete signal:

```
DROP SIGNAL <signal_name>
```

Signals can be dropped only if no other dependent triggers reference the signal

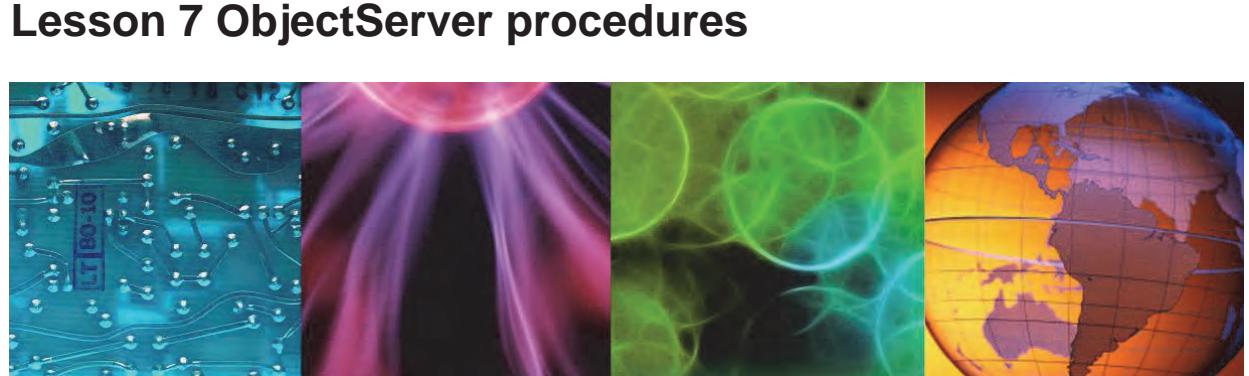
The name of the signal must be unique and comply with naming conventions

Raising and dropping signals

A user-defined signal must be raised with the RAISE SIGNAL command. You can create a tool that raises a signal. You can create a trigger that raises a signal. You can raise a signal manually with the nco_sql utility.



Lesson 7 ObjectServer procedures



© Copyright IBM Corporation 2016

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 SQL and external procedures.

ObjectServer procedures

An executable SQL script or external object that you call to perform common operations

Types of procedures:

- SQL procedures, which manipulate data in ObjectServer databases
- External procedures, which run an executable file on local or remote host. Process Automation is required to run a procedure on any host

Procedures are stored in the ObjectServer

Procedures are run from the following sources:

- nco_sql
- Trigger
- Desktop tool

Use the EXECUTE <PROCEDURE> command to run the procedure

ObjectServer procedures

A procedure is simply one or more commands that are stored in the ObjectServer. Each procedure has a unique name. The procedure is run when the name is referenced in a trigger or a tool.

There are two types of procedures. An SQL procedure is a collection of SQL commands. The SQL commands run inside the ObjectServer. An external procedure is a command or script that is run outside of the ObjectServer. The command or script runs on a host. The host can be local to the ObjectServer, or it can be a remote host.

SQL procedure syntax

Syntax

```
CREATE [ OR REPLACE ] PROCEDURE procedure_name ( [ procedure_parameter,...  
        ] )  
[ DECLARE variable_declaration;...[ ; ] ]  
BEGIN  
procedure_body_statement;...[ ; ]  
END
```

Example

```
CREATE PROCEDURE calculate_average_severity  
( IN severity_arr ARRAY OF INTEGER)  
BEGIN  
procedure_body_statement;...[ ; ]  
END
```

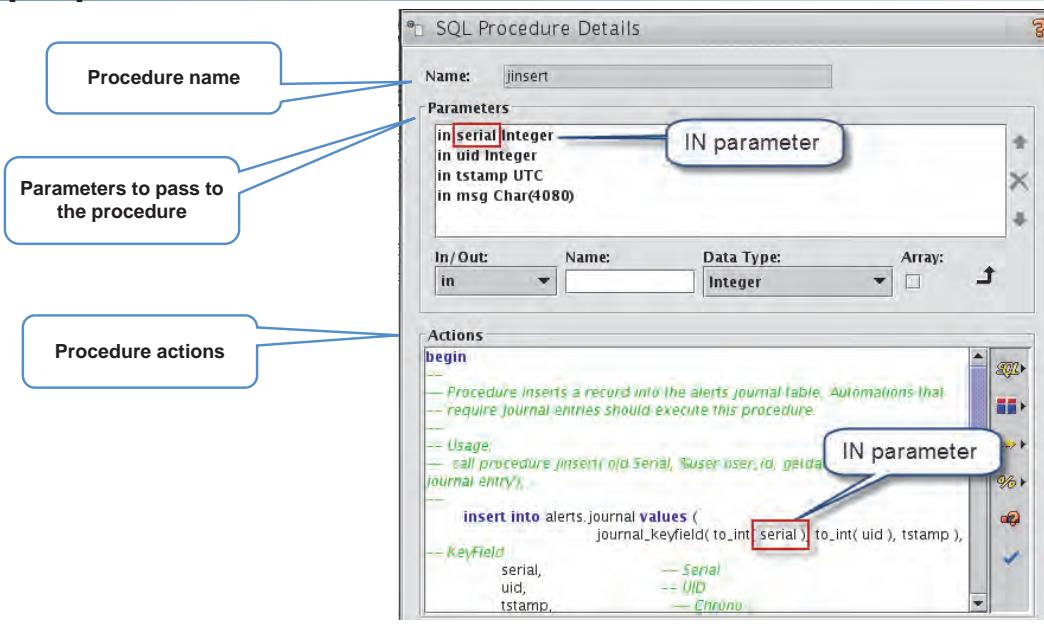
© Copyright IBM Corporation 2016

61

SQL procedure syntax

This slide illustrates the commands to create an SQL procedure. The procedure can also be created with the Netcool/OMNIbus Administrator utility.

Example procedure



© Copyright IBM Corporation 2016

62

Example procedure

An SQL procedure consists of parameters, and actions. This slide shows the definition for the jinsert procedure. The jinsert procedure creates a journal record. The parameters values identify the corresponding event record, and the message to place in the journal entry. You can use this procedure to create a journal entry when a trigger modifies an event record.

Parameter declaration

Procedures can have parameters

Each procedure parameter has a mode:

- IN
- OUT
- IN OUT

Depending on the mode that is chosen for parameters, you can use them in different ways

Parameter declaration

Procedure parameters are defined as IN, OUT, or IN OUT. The type of definition determines how they are used within the procedure.

Parameter declaration: IN mode

An IN parameter is a read-only variable

You can use the IN parameter in expressions to help calculate a value, but you cannot assign a value to the parameter

IN parameters are useful for passing variable values into a procedure that you do not want to change in the procedure

IN mode is the default if you do not specify the parameter mode

Parameter declaration: IN mode

A parameter that is defined as IN can be used only as input to a procedure. For example, an IN parameter that is called Node can be used to pass the value for a device name into the procedure.

Parameter declaration: OUT mode

An OUT parameter is a write-only variable

You can use an OUT parameter to assign a value to the parameter, but you cannot read from it in the body of the procedure

Therefore, you cannot use this type of parameter in an expression

OUT parameters are useful for passing values that are computed in a procedure out of the procedure

Parameter declaration: OUT mode

A parameter that is defined as OUT can be used only as output from a procedure. For example, an OUT parameter that is called RETURNCODE can be used to pass a value as the result of the procedure processing.

Parameter declaration: IN OUT mode and type

IN OUT Mode

- An IN OUT parameter is a read-and-write variable, with none of the constraints of an IN or OUT parameter
- This mode is useful for variables that you want to change in the procedure and pass out of the procedure

Parameter Type

- When creating parameters, you declare a data type for each parameter
- The data type can be any valid ObjectServer data type except VARCHAR or INCR
VARCHAR, and INCR are valid types for database tables, not variables

Parameter declaration: IN OUT mode and type

A parameter that is defined as IN OUT can be used as input or output with a procedure.

Variable declaration

You can define (declare) local variables to use in a procedure

- These variables are persistent in subsequent executions of a procedure
- Must initialize if this is not required

A local variable is a placeholder for values that are used during the execution of the procedure

Local variables are available only in the procedure

When declaring variables, you must supply a name and data type for each variable

The data type can be any valid ObjectServer data type except VARCHAR or INCR

© Copyright IBM Corporation 2016

67

Variable declaration

A local variable is used within a procedure. You use the DECLARE statement to define the local variable by name, and type.

External procedures

An external procedure runs a command outside of the ObjectServer

- The command runs on a server

The procedure defines:

- The host name of the server
- The user to run the command
- The command to run

The command can be any valid command based on the target server

- Native operating system command
- Shell file
- Script

Process Activity is required to run external procedures

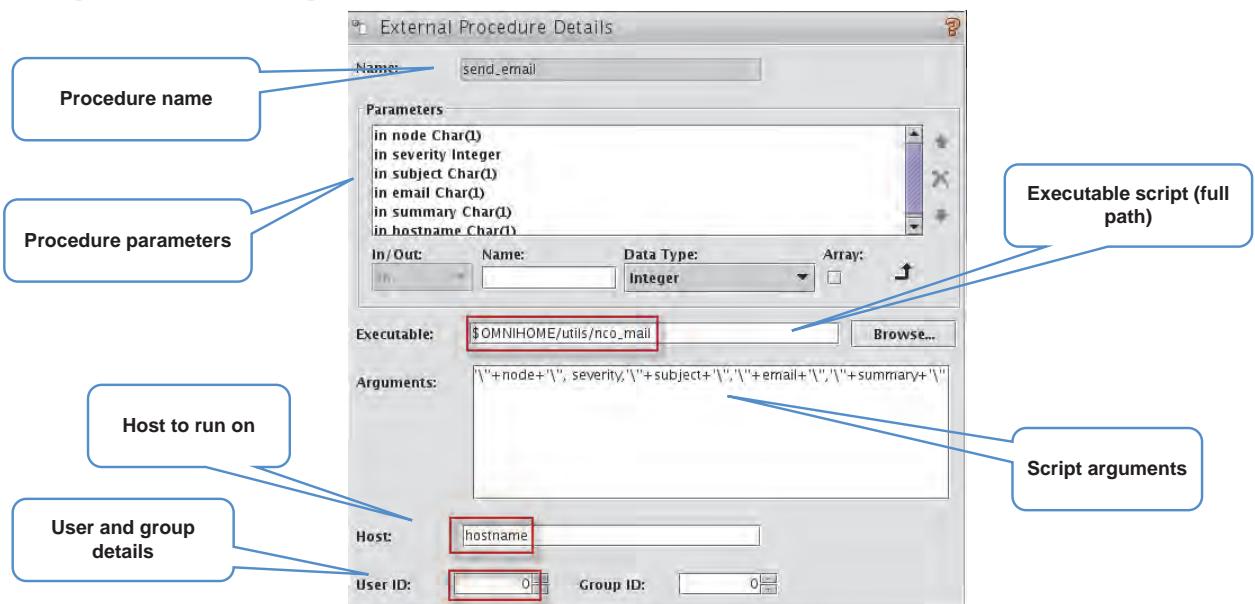
- ObjectServer passes request to Process Activity
- Process Activity runs the corresponding command

External procedures

An external procedure runs a command outside of the ObjectServer. The procedure definition identifies the host, the user, and the command. Because the command runs on host, the possible commands vary by host operating system. You can run a native operating system command, a script, or a bat file.

You must run process activity to use external procedures. The external procedure is called within the ObjectServer. The ObjectServer notifies process activity that a command must be run on a host. The ObjectServer passes the host, user, and command name to process activity. The process activity daemon runs the command.

Example External procedure



© Copyright IBM Corporation 2016

69

Example External procedures

This slide contains the definition for the send_email external procedure. This procedure is run from the mail_on_critical trigger. The trigger selects an event based on some condition. When the trigger locates an event, the trigger calls the send_email procedure, and passes several parameters. The procedure uses the parameters to construct a list of arguments. The ObjectServer notifies the process activity daemon that a command must run on a designated host. Process activity runs the command with the list of arguments.

Running a procedure

You can run a procedure from Tools, from ISQL, or from an automation trigger

Syntax

```
{ EXECUTE | CALL } [ PROCEDURE ] procedure_name  
[ ( expression,... ) | ( [ expression, expression,... ] ,... ) ];
```

Example

```
EXECUTE PROCEDURE myproc();
```

Or with parameters

```
EXECUTE PROCEDURE myproc("text",@Node,%user);
```

Running a procedure

Procedures are run with the EXECUTE or CALL PROCEDURE command. The command must contain the procedure name, and any required parameter values. The command is used for SQL, and external procedures.

Student exercises



Perform the student exercises for this section.

Summary

You now should be able to perform the following tasks:

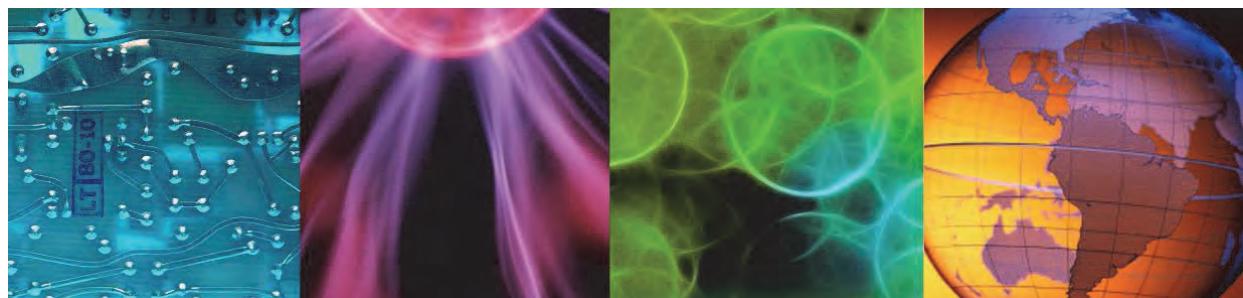
- Use SQL to examine ObjectServer tables
- Use SQL in automations
- Create a database trigger
- Create a temporal trigger
- Create a signal trigger
- Create procedures



5 Web GUI administration



Web GUI administration



© Copyright IBM Corporation 2016

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

in this unit, you learn basic Web GUI administration. You learn how to create event filters and views. You learn how to create desktop tools and add them to menus. You learn how to create a map and place the map on a page. You learn how to create a gauge and add the gauge to a page. You also learn how to use the Web GUI administrative API to perform basic administration from the command line.

References: SC27-6505-00 *Netcool/OMNIbus Version 8 Release 1 Web GUI Administration, and User's Guide*

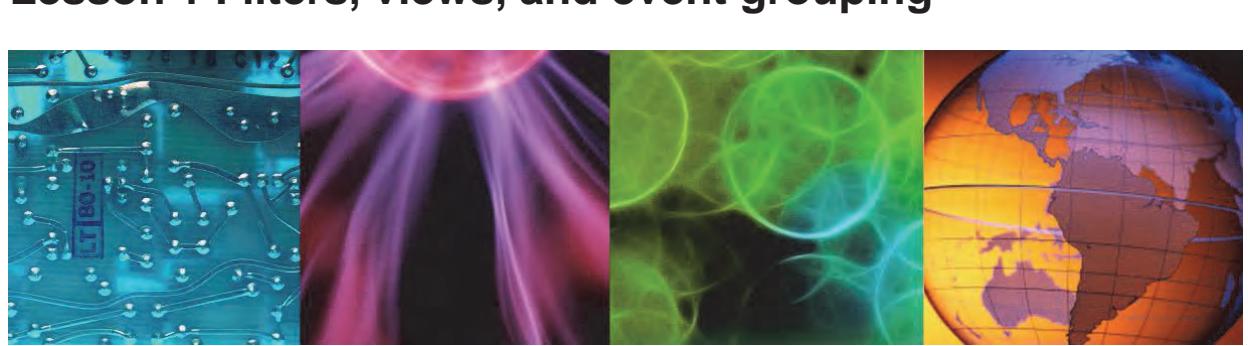
Objectives

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

- Create event filters and views
- Configure event grouping
- Create desktop tools
- Configure and create map pages
- Add a gauge and define thresholds
- Configure and use the Web GUI administrative API client



Lesson 1 Filters, views, and event grouping



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Create an event view
- Create an event filter
- Configure event grouping

Displaying event data

Two aspects to displaying event data

- View
- Filter

View defines how the event data is displayed

- Which columns
- Column headings
- Column order
- Event grouping
- Other

Filter defines which event records are displayed

- Only critical events
- Only events for Customer XYZ
- Other

© Copyright IBM Corporation 2016

4

Displaying event data

When a user displays event records in a desktop, two features control the display. The first feature is the event view. The ObjectServer event record contains more than 100 data columns. Most users are interested in only a few of those data columns. The event view is used to define which columns appear in the user display.

The second feature is the event filter. The ObjectServer typically contains numerous event records. Again, many users care only about some of the event records. The event filter defines the criteria for the events that the user needs to see. For example, a database administrator cares about events that are related to databases. That user does not care about network-related events.

Creating a view: Display columns

The screenshot shows the 'Administration' feature in the IBM Dashboard Application Services Hub. A blue arrow points from the 'Views' icon in the sidebar to the 'Views' window. The 'Views' window shows a list of existing views and a 'New View' button. In the 'Edit View: New View' window, a red box highlights the 'global' checkbox under 'Public'. Below it, a list of users ('jones', 'ncost', 'nouc') has checkboxes next to them. The 'Display Columns' tab is selected, showing a list of available fields (e.g., AENName, Acknowledged, AdvCorrCauseType) and a list of event list view columns (e.g., Serial [locked], LastOccurrence, Node, Summary, Tally). Red boxes highlight the 'add the column' button between the two lists and the move column up/down buttons on the right.

Create new view

1. Select type (global, system)
2. Enter name for view
3. Select column on left
4. Click right arrow icon
5. Move column up or down in the list

Creating a view: Display columns

There are two types of views: global and system. A global view is available to all users. A system view is available to only administrators. A user without administrative authority can copy a global view to their own profile. The user can modify the copy of the view and save the changes to their profile. A view in a user profile is available only to the user who owns the profile.

Access to the Administration feature within Dashboard Application Services Hub requires the *ncw_admin* role. To create a view, the user opens the Administration feature and selects **Views**. The Views page contains a list of the existing views. Collections organize the views. The collections are:

- My Views
- Global Views
- System Views

The default display is set to **My Views**. You click the icon to create a new view. In the next page, you select the type of view: global or system. You can also select one or more users. If you select a user, the view is saved in that user's profile. After selecting the type of view, the View Builder opens.

The view name must be unique, and cannot contain spaces or special characters, except the underscore. If the Web GUI is configured with access to multiple ObjectServers, you can select which ObjectServers to use for the view. If you select multiple ObjectServers, the list of column names is limited to the column names that are common to both ObjectServers.

Four tabs are used to define event column characteristics. The first tab is labeled **Display Columns**. When this tab is selected, the list of available columns appears. To add a column to the

display, you click an available column and click the right arrow icon to move the column to the list of selected columns. The column is added at the bottom of the list. You use the up and down arrows to move the column within the list. The first column in the selected list appears on the left of the user display.

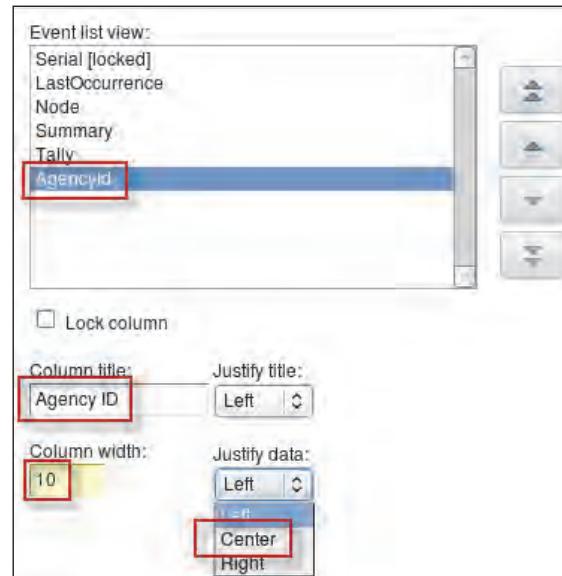


Hint: Configure event views with the most interesting column names at the top of the selected list. These columns appear on the left of the user display, and minimize the need to scroll back and forth in the event display. You can also *lock* one or more column names to eliminate scrolling.

Creating a view: Display columns, continued

Configure display characteristics

1. Select column in list
2. Set column title
3. Set title justification
4. Set column width
5. Set data justification



© Copyright IBM Corporation 2016

6

Creating a view: Display columns, continued

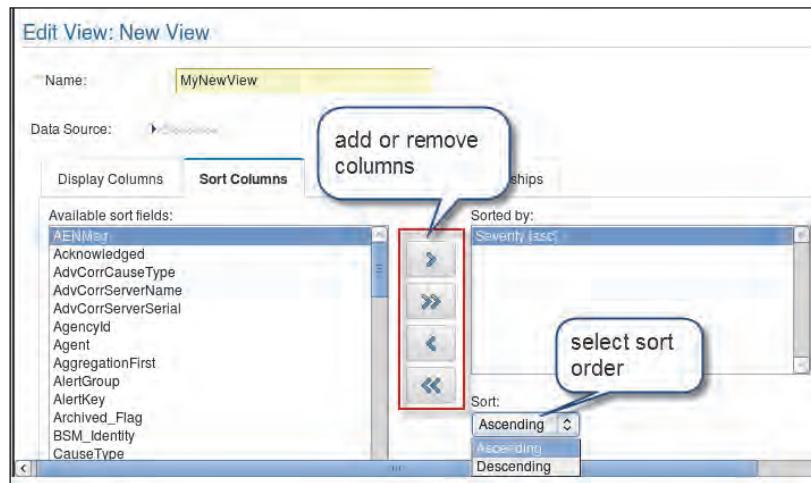
Click a column in the selected list, and scroll to the bottom of the page. On the bottom of the page, you configure the column display characteristics. You configure the column title, title justification, column width, and the data justification.

The default display characteristics for column names are defined in the *column visuals table*. When you define the display characteristics for a column in the column visuals table, the characteristics are applied to all event views where that column is used. When you maintain accurate data in the column visuals table, you eliminate the need to manually change the display characteristics in individual views.

Creating a view: Sort columns

Define sort order

1. Add column to list
2. Select column in list
3. Select sort order
4. Add more columns if necessary



© Copyright IBM Corporation 2016

7

Creating a view: Sort columns

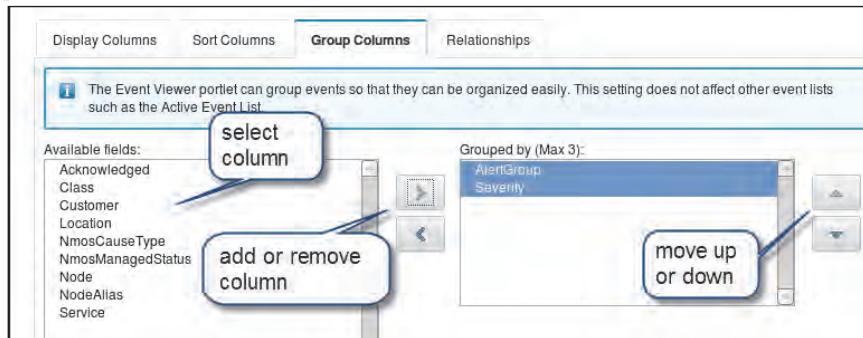
The next tab is labeled **Sort Columns**. When this tab is selected, the list of selected column names is displayed. You click a column name, and then click the right arrow icon to add the column to the sort order. The column is added to the bottom of the list. You use the up and down arrows to move the column within the list. Click a column name, and then select the sort order of either ascending or descending.

The sort information in the view controls the order in which the event records are shown in the user desktop. For example, when the sort order contains Severity Descending, the event records appear with the highest severity values first.

Creating a view: Group columns

Configure grouping

1. Select column
2. Add to list
3. Move column up or down in the list



List of available columns

```
/opt/IBM/netcool/omnibus_webgui/etc/server.init  
columnngrouping.allowedcolumns=Acknowledged,AlertGroup,Class,  
Customer,Location,Node,NodeAlias,NmosCauseType,NmosManagedSt  
atus,Severity,Service
```

Add columns to the list

Restart Web GUI

© Copyright IBM Corporation 2016

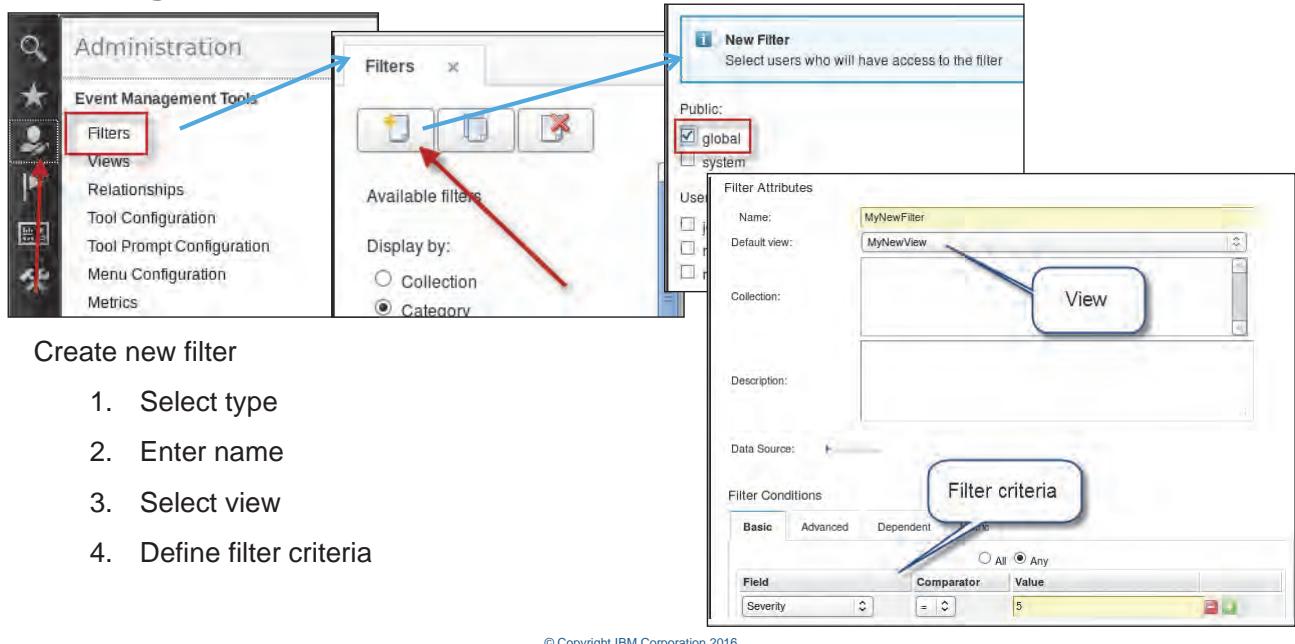
8

Creating a view: Group columns

The next tab is labeled **Group Columns**. This tab is used to define an optional event grouping. The event grouping is available in only the Event Viewer. The list of available column names is configured in the Web GUI server.init file. There are two parameters in this file that control the grouping configuration. One parameter defines the maximum number of grouping columns. The default is 3. The second parameter contains the list of column names available for use with grouping. You can modify the server.init file to change these parameters. You must restart Dashboard Application Services Hub for the changes to take effect.

The last tab is labeled **Relationships**. You use this tab to define an advanced level of event grouping based on a known relationship between event records. For example, some events can be considered to be the root causes of problems, and some events can be considered to be symptoms of those problems. You can define a hierarchical event relationship in which root cause events are treated as parent events, and are displayed at the top level of the hierarchy. Symptom events are treated as child events, and are displayed below root cause events in the hierarchy.

Creating a filter



Creating a filter

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

An event filter defines an SQL *where* clause. When a user opens a display, the corresponding SQL *where* clause is used to search the contents of the ObjectServer alerts.status table. Any event record that is found by the search appears in the user display.

A user requires the *ncw_admin* role to access the Administration feature within Dashboard Application Services Hub. To create a filter, you select **Filters**, and the list of available filters opens. The filters are organized by Collection and Category. A collection is used to organize filters. There are two default collections: Global and System. You can use the Filter Builder only to add filters to collections or remove filters from collections. To create or delete filter collections, or to modify collections use the WAAPI client.

Filter categories also organize filters, and control access to the filters. The filter categories are also Global, and System. Only an administrator can access a system filter. Global filters are available to all users.

To create a filter, you click the *new filter icon*. You select the type of filter, either system or global. You can also select a user or group. A filter that you create for a user is available to only the user. A filter that you create for a group is available to all users that belong to the group. Next, the **Filter Builder** opens.

Enter a name for the filter. The name cannot contain spaces or special characters except for the underscore. You select a view to associate with this filter. You also select one or more ObjectServers from the data source list.

Creating a filter: Filter condition

Define conditions

1. Select column
2. Select comparison type
3. Select value
4. Add second column
5. Define condition
6. Select how conditions are combined

AND or OR

The screenshot shows the 'Filter Conditions' dialog box with the 'Basic' tab selected. It displays two rows of conditions. The first row has 'Severity' as the field, '=' as the comparator, and '5' as the value. The second row has 'Acknowledged' as the field, '=' as the comparator, and '0' as the value. A red box highlights the first row. Above the rows, there are tabs for 'Basic', 'Advanced', 'Dependent', and 'Metric'. Below the rows, there are buttons for 'Comparator' (with options like '=', '>', etc.) and 'Value'. At the top right, there are buttons for 'All' and 'Any'. A green plus sign icon is located at the bottom right of the condition rows, with a tooltip 'add another condition' pointing to it.

© Copyright IBM Corporation 2016

10

Creating a filter: Filter condition

The Filter Builder contains four tabs. The first tab is labeled **Basic** and is used to define a simple condition. The filter condition consists of a column name, a comparison operator, and a value. The pull-down list under *Field* contains all column names from the selected ObjectServer. If more than one ObjectServer is selected, the list of columns contains the names that are common to all selected ObjectServers. The conditional operators that are found in the *Comparator* pull-down list vary based on the type of column that you select.

To configure a complex condition, you click the *green plus sign* icon to add another line. Then, you select the column, comparator, and value for the second condition. In addition, you select the type of logical operator for the two conditions. To configure an *ADD* operator, you select **All**. To configure an *OR* operator, you select **Any**.

The second tab is labeled **Advanced**. If you are comfortable with the SQL language, you can select the Advanced tab, and enter the SQL commands for your filter.

The third tab is labeled **Dependent**. You use this tab to define a filter that depends on one or more other filters. For example, you can have a filter that locates security events that are related to a firewall. You can have another filter that locates security events that are related to an intrusion detection system. You can create a third filter that depends on both of the security filters. You can think of the dependent filter as a container. The contents of the container, in this example, are based on whether there are any firewall events, or whether there are any intrusion detection events.

The last tab is labeled **Metric**. Every filter that you create results in a monitor box in the Event Dashboard. You use Metric tab to configure the display characteristics of the corresponding monitor box in the Event Dashboard.

Using a filter and view

The screenshot illustrates the process of creating and applying a new filter and view. On the left, the 'Incident' navigation pane shows the 'Event Dashboard' selected. A blue arrow points from this selection to the 'Event Dashboard' monitor box on the Event Dashboard page. The Event Dashboard page displays five filter monitors: 'AllEvents', 'Default', 'Escalated', 'MyNewFilter', and 'Unacknowledged'. A callout bubble labeled 'new filter' points to the 'MyNewFilter' monitor. Another blue arrow points from the 'MyNewFilter' monitor to the 'MyNewView' tab in the Active Event List (AEL) window. The AEL window shows a single event: 'Machine has gone offline' at '9/4/14 8:17:04 PM' from 'Sydney'. A callout bubble labeled 'new view' points to the 'MyNewView' tab. The AEL window title bar reads 'MyNewFilter@OMNIBUS - Active Event List (host2.tivoli.edu:16311)'.

New filters are immediately available for use

Automatically appear in the Event Dashboard

The view is associated with the filter

© Copyright IBM Corporation 2016

11

Using a filter and view

After the filter is saved, it is immediately available for use. The Event Dashboard page contains one monitor box for every filter. After the filter is saved, when the Event Dashboard page opens, a monitor box appears for the filter. You click the monitor box, and the Active Event List (AEL) opens. The events that are displayed in the AEL must meet the criteria in the filter conditions. The corresponding view controls the display configuration.

The user can manually change the contents of the AEL or Event Viewer. There is a pull-down list for filters and another pull-down list for views. The user can select a different filter or view from these pull-down lists.

Event grouping

The screenshot shows the Event Viewer interface. In the left sidebar, 'Event Viewer' is selected. The top bar shows a filter named 'MyNewFilter'. The main area displays a table of events grouped by 'AlertGroup' and 'Severity'. The table includes columns for Group, Count, Sev, Serial, LastOccurrence, Node, and Summary. Hierarchical grouping is indicated by plus signs (+) next to some group names.

Group	Count	Sev	Serial	LastOccurrence	Node	Summary
All	8	2	22,946	12/2/14 7:57:06 PM	host2.tivoli.edu	siminet pro
Probe Status	2	2	9,451	10/23/14 6:26:49 PM	host1.tivoli.edu	syslog pro
probestat	3	2	10,633	10/23/14 6:35:19 PM	host1.tivoli.edu	mittrapd pro
ProbeStatus	1	2	74,415	12/8/14 1:44:38 PM	host1.tivoli.edu	ALERT: sy
TopClasses	1	2	12,823	12/8/14 1:44:38 PM	host1	ALERT: las
TopNodes	1	2	24,419	10/23/14 7:33:26 PM	host1.tivoli.edu	Probe Goin
Critical	1	2	13,772	10/23/14 7:34:19 PM	host1.tivoli.edu	ALERT: las

© Copyright IBM Corporation 2016

12

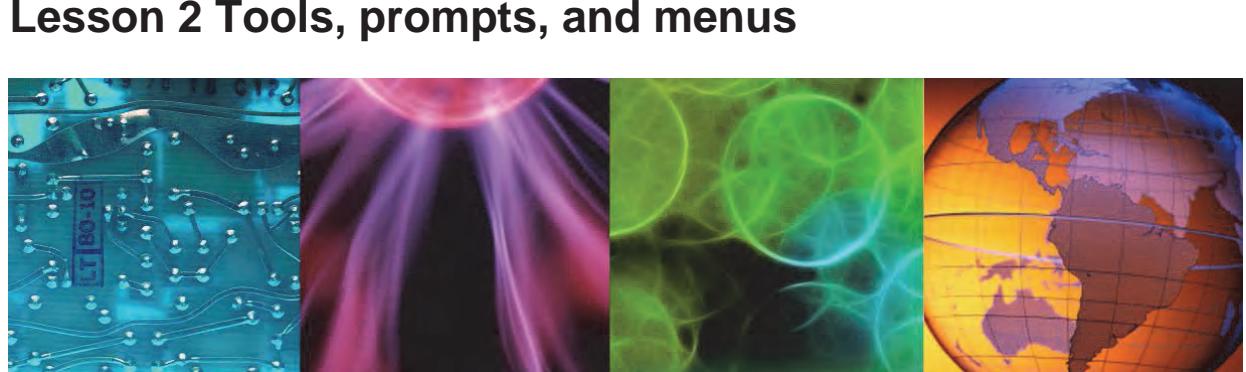
Event grouping

Event grouping is configured in the view. Event grouping is available only in the Event Viewer application. The event grouping appears on the left side of the display, and is in addition to the column name definitions that appear in the view. The user can expand or collapse individual groups.

In the sample screen capture that is shown here, two columns are used for event grouping. The first column is AlertGroup, and the second column is Severity. The event records are organized hierarchically based on the values for these two columns.



Lesson 2 Tools, prompts, and menus



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Create tool prompts
- Create tools
- Add tools to menus

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 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

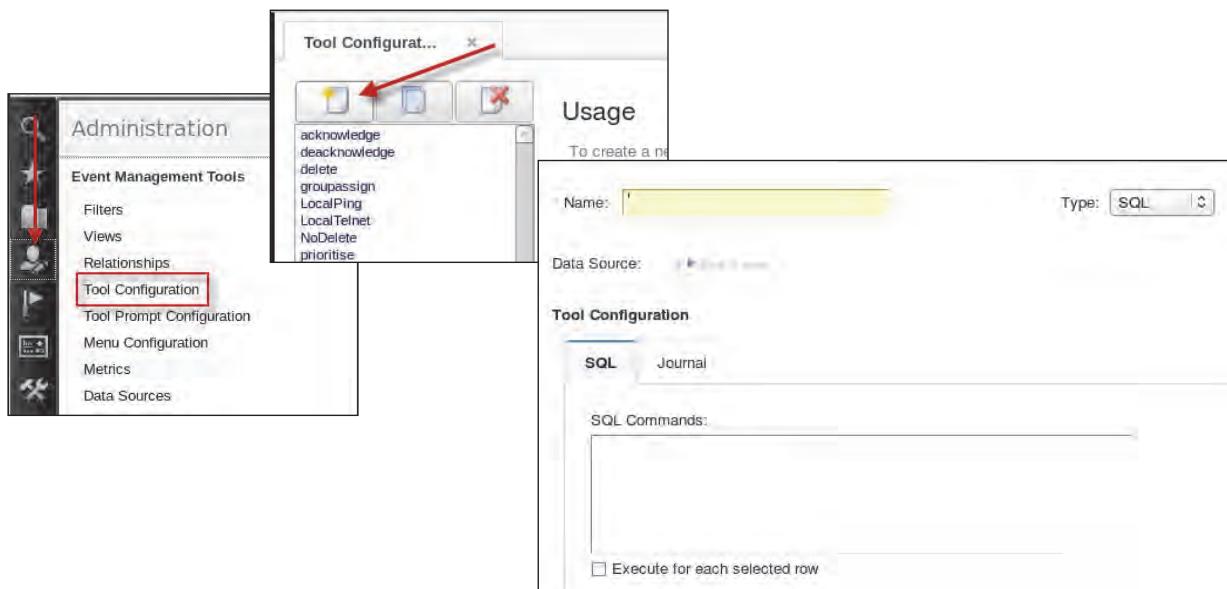
Access Web GUI tools with the Active Event List (AEL) or the Event Viewer

Web GUI tools

A tool is an action that is run from a user desktop. The action is typically associated with an event record. For example, select an event in the display, and run a tool that removes the event from the ObjectServer. Web GUI tools are defined within the Web GUI server. Web GUI tools are convenient when you use more than one ObjectServer in your environment. You need only to define the tool in the Web GUI server, and it is available for use with all ObjectServers that are defined to that instance of Web GUI.

Web GUI tools are accessed with the Active Event List or Event Viewer applications.

Creating a Web GUI tool



© Copyright IBM Corporation 2016

15

Creating a Web GUI tool

A user requires the *ncw_admin* role to access the Administration feature within Dashboard Application Services Hub. To create a tool, you open the Administration feature and select **Tool Configuration**. The list of available tools opens. You click the icon to create a new tool. You enter a tool name, which cannot contain spaces or special characters except the underscore. Then, you select the type of tool with the pull-down list.

There are four types of tools.

- SQL: The tool runs one or more SQL commands when selected. The commands run in the ObjectServer.
- Command: The tool runs a command on the user workstation when selected. The tool can use any command that is supported by the workstation operating system.
- CGI/URL: The tool runs a CGI script when selected. The script runs on the Web GUI server.
- Script: The tool runs a Java script when selected. The script runs on the user workstation.

Command tool example

Command runs on user workstation

Define command syntax by operating system

Can use any command supported by the operating system

Name: LocalPing Type: Command

Data Source: Click to choose

Tool Configuration

Command

Platform:

- Windows
- Solaris
- Linux
- HP-UX
- AIX

Executable command:

start cmd /k %WINDIR%\SYSTEM32\PING.EXE {@Node}
xterm -e /bin/sh -c '/usr/sbin/ping {@Node}; read a'
xterm -e /bin/sh -c '/bin/ping {@Node}; read a'
xterm -e /bin/sh -c '/usr/sbin/ping {@Node}; read a'
xterm -e /bin/sh -c '/usr/sbin/ping {@Node}; read a'

The value of the Node column in the selected event record is passed to the tool

© Copyright IBM Corporation 2016

16

Command tool example

The screen capture shows the configuration of the **LocalPing** tool. This tool is included with Netcool/OMNibus. Because command tools run on user workstations, the tool definition is based on operating system. In the example, the same tool runs a different command whether the tool is run from a Windows workstation or Solaris workstation.

Observe the use of @Node in each command string. The reference to @Node indicates that the value of the Node column is extracted from the selected event record. The value is substituted in the command string before the tool runs.



Important: By default command tools do not appear on the tools menu in Event Viewer. You can install a custom Tivoli Netcool/OMNibus Web GUI Tool Launch plug-in to run command tools from the Event Viewer. Refer to the Web GUI Administration Guide for complete details on how to install the plug-in.

SQL tool example

SQL command runs in the ObjectServer

The screenshot shows the configuration of the 'acknowledge' tool. In the 'Tool Configuration' section, the 'SQL' tab is selected. The 'SQL Commands:' field contains the following SQL update statement:

```
update alerts.status set Acknowledged=1 where Serial in ($selected_rows.Serial);
```

A callout bubble points to the 'Selected' row in the 'Active Event List' table, explaining that the tool runs against every event that is selected in the event list.

Sev	Ack	Delete_Flag	Archive_Flag	Node	Alert
●	Yes	No	Yes	Acknowledge	Ctrl+A
●	Yes	No	Yes	De-acknowledge	Ctrl+D
●	Yes	No	Yes	Prioritize	> TopN
!	Yes	No	Yes	Suppress/Escalate	> Gene
!	Yes	No	Yes	Take ownership	Stats

© Copyright IBM Corporation 2016

17

SQL tool example

The screen capture shows the configuration of the **acknowledge** tool. This tool is included with Netcool/OMNIbus. The tool runs a single SQL command that modifies the value of the Acknowledged column in the selected event record. The SQL command includes the following text:

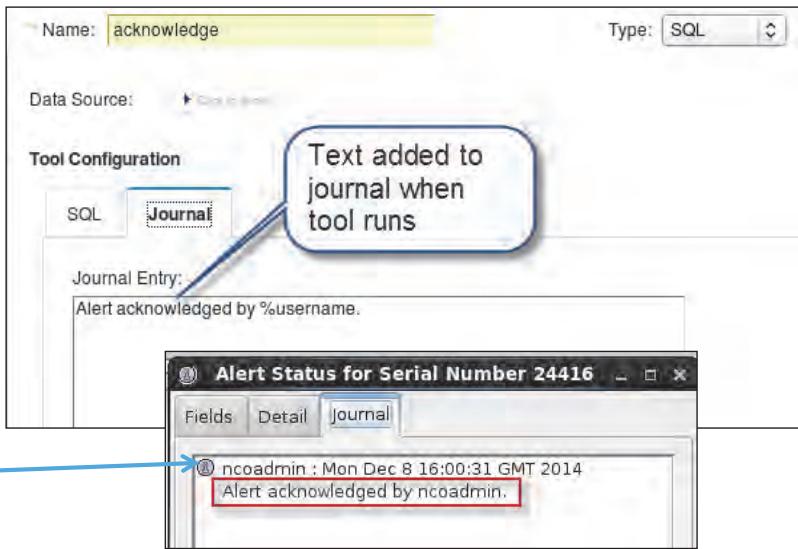
```
where Serial in ($selected_rows.Serial)
```

This text causes the SQL update command to run against every selected event record. The user can select one or more event records, and run the tool once. The tool modifies every selected record.

SQL tool journal entry

Specify text to add to journal when tool runs

User ID, date, and time added automatically when tool runs



© Copyright IBM Corporation 2016

18

SQL tool journal entry

The SQL tool can create a journal record when it is used. The journal record is added to the alerts.journal table when the tool runs. The corresponding journal record contains the user ID of the user who used the tool. The record also contains the date, and time. This information is automatically added to the journal. The tool definition contains extra text that is added to the journal record in addition to the user ID, date, and time.

CGI tool example

CGI command runs on
Web GUI server

The screenshot shows the configuration of the **RemotePing** tool. The tool name is **RemotePing**, and its type is **CGI/URL**. The URL is set to `$(SERVER)cgi-bin/nco_ping.cgi`. A callout bubble points to this URL field with the text "CGI command must be registered". Other settings include Method (GET), Open In (New window), and two checked checkboxes at the bottom: "Execute for each selected row" and "Window for each selected row".

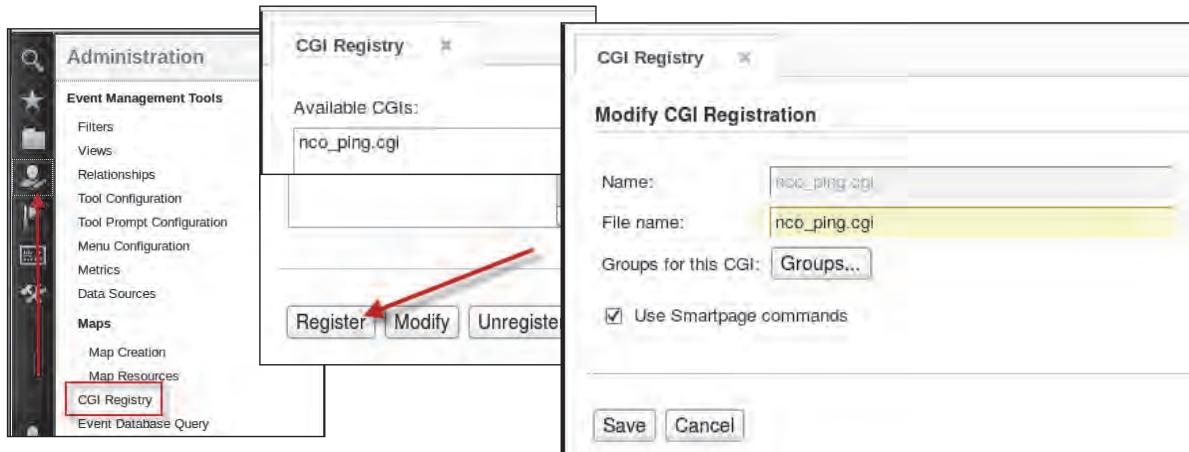
© Copyright IBM Corporation 2016

19

CGI tool example

The screen capture shows the configuration of the **RemotePing** tool. This tool is included with Netcool/OMNIbus. The tool runs a CGI script on the Web GUI server when selected. The tool name, **RemotePing**, indicates that the tool runs *remote* from the user. Similarly, the **LocalPing** tool that is shown previously, indicates that the tool runs *local* to the user. The CGI tool is convenient for situations where the user workstation cannot run a command. The most common example is a situation where the user workstation cannot access a remote device because it lacks network access to the device. Perhaps the device is located behind a firewall. If the Web GUI server has network access to the device, you can use a CGI tool.

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 CGI Servlet, extracted from Apache Tomcat

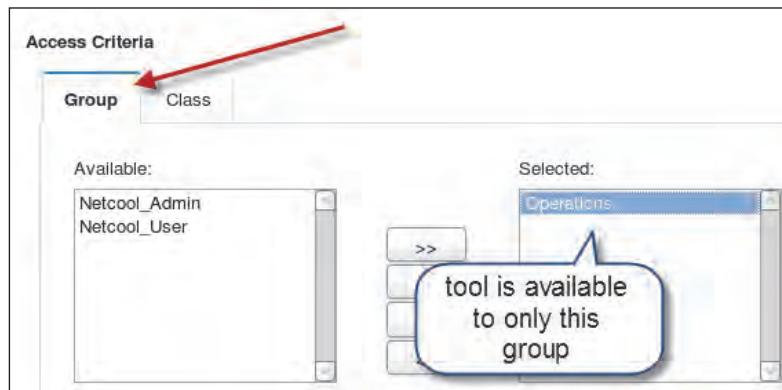
© Copyright IBM Corporation 2016

20

CGI command registration

Before you can use a CGI tool, you must *register* the CGI script with the Web GUI server. First, you save the CGI script in a directory on the Web GUI server. Then, open the Administration feature and select **CGI Registry**. Enter the name of the script and click **Save**.

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 2016

21

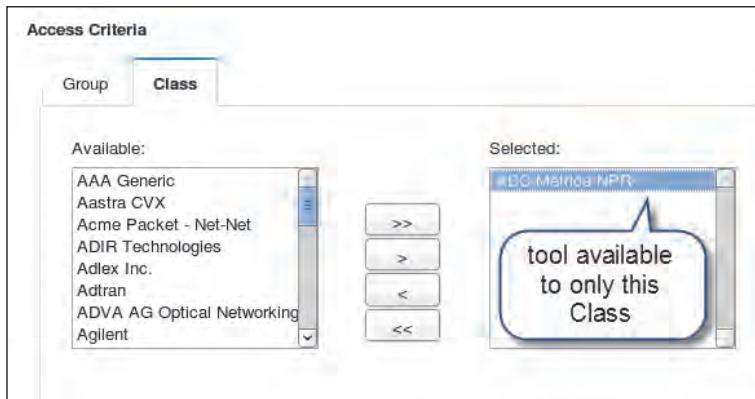
Limiting tool access by group

One of the primary uses of desktop tools is to create linkages to various applications that you use to diagnose infrastructure issues. You can limit access to tools based on group settings. This type of control allows you to create a tool to access a diagnostic application, and restrict access to the tool to only members of a specific group.

By default, all groups have access to a tool. You limit access by selecting one or more groups.

Group access is available for all four types of tools.

Limiting tool access by class

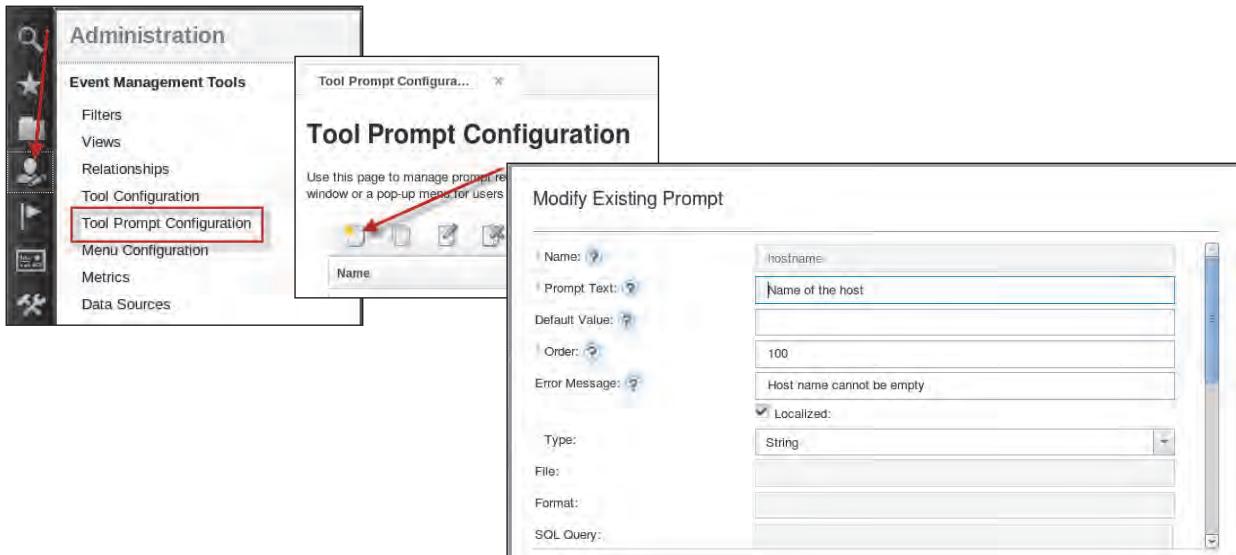


- All event classes have access by default
- Adding a class to the selected column defines the event class that is applicable to the tool
- The tool is accessible in the menu for any event that contains the selected class

Limiting tool access by class

Another technique for limiting access to tools is by event class. Class is a column name in the event record. The Class column contains an integer value. Class typically identifies the vendor or technology that is related to the event. The Class value provides a convenient mechanism for restricting a tool to a category of event. For example, an event that is related to an IBM power server. The user clicks an event in the desktop, and right-clicks to open the list of tools. The tools that appear are the tools that are applicable to the selected event. For example, if the event is related to an IBM power server, then the corresponding tool appears in the list.

Creating a tool prompt



© Copyright IBM Corporation 2016

23

Creating a tool prompt

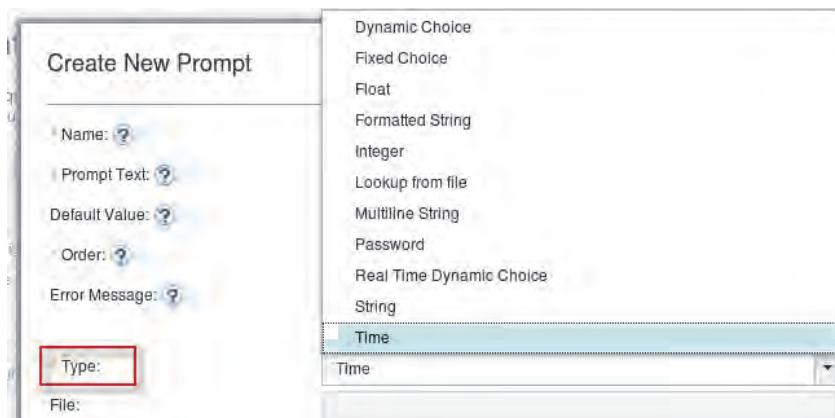
A prompt is used when the user must provide some type of input for the tool. In many cases, the tool can extract context from the event record, and does need a prompt. In other cases, a prompt can be required. To create a prompt, open the Administration feature and select **Tool Prompt Configuration**. The list of available prompts opens. Click the icon to create a new prompt.

Enter a value for the prompt name, which cannot contain spaces or special characters except for the underscore. Enter a message for the Prompt Text. This message appears on the user desktop when the tool is run. You can enter a default value that is used in the tool if the user does not enter a value.

If a tool contains more than one prompt, the value for Order determines the order in which the prompts appear to the user.

Prompt types

- Dynamic Choice
- Fixed Choice
- Formatted String
- Integer
- Lookup from file
- Multiline String
- Password
- Real Time Dynamic Choice
- String
- Time



© Copyright IBM Corporation 2016

24

Prompt types

Select the type of prompt from the following choices:

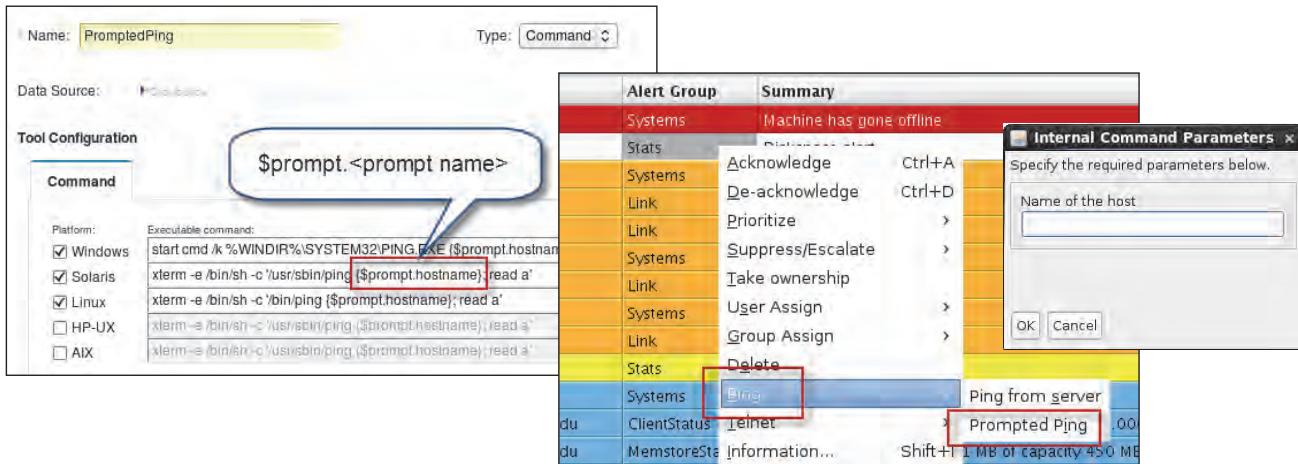
- String: Accepts one or more characters.
- Integer: Accepts an integer value with a minimum value of -2,147,483,648 and a maximum value of 2,147,483,647.
- Float: Accepts a floating point number, which can contain a decimal point.
- Time: Accepts a time and date. You can enter times and dates as free text or by clicking the selector buttons next to the text fields and selecting values.
- Lookup: Creates a menu or list that populated by the values in a specified file. The file attribute contains an absolute path to a file on the Web GUI server, where each line of text is displayed as an item in a list.
- Password: Accepts one or more characters as a password.
- Dynamic Choice: Creates a menu or a list that populated by the results of a query on the ObjectServer.



Note: Most of the included tools use dynamic choice prompts. You can view one of the existing prompts to see an example of how the prompt is defined.

- Multiline String: Creates a multiline prompt window that accepts one or more characters.
- Formatted String: Accepts one or more characters, provided the characters are in the predefined format. The **format** attribute contains a regular expression that must be matched for the value to be accepted.
- Real-Time Dynamic Choice: Creates a scrollable list that is populated by the results of a query on the ObjectServer in real time. Use this prompt type to display data from ObjectServers that contain data that changes frequently. Because this prompt type is run in real time, use it sparingly so that the server is not overloaded.

Using a prompt in a tool



© Copyright IBM Corporation 2016

25

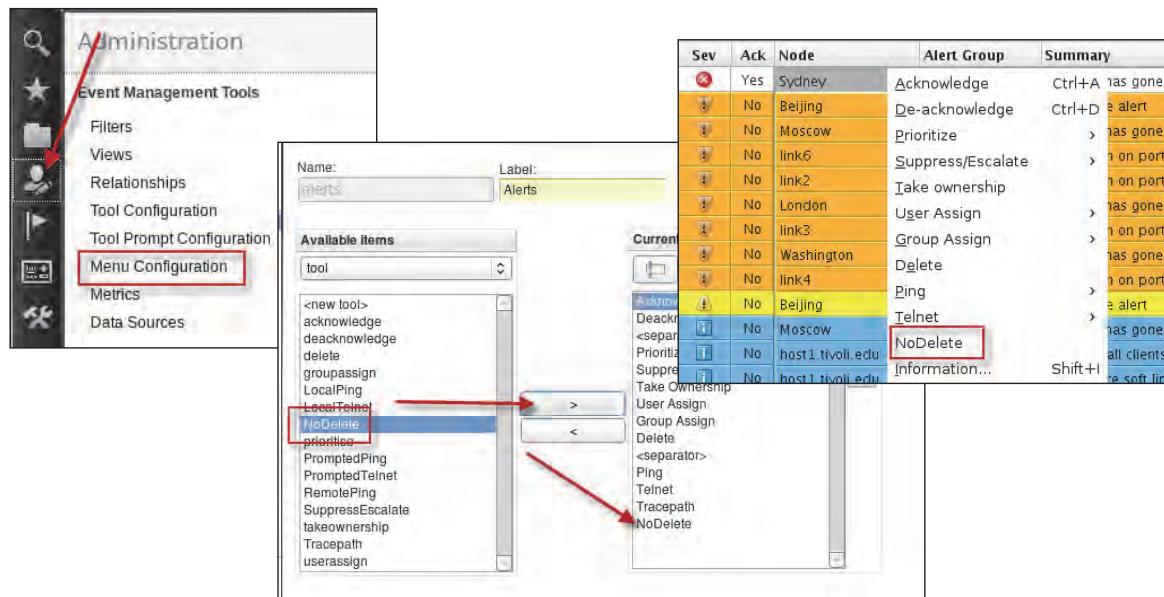
Using a prompt in a tool

After the prompt is defined, you reference the prompt in a tool with the following string:

```
$prompt.<prompt name>
```

When the tool runs, a window opens on the user desktop, and the user provides the necessary information to satisfy the prompt. After the user completes the prompt, the tool runs.

Adding a tool to a menu



© Copyright IBM Corporation 2016

26

Adding a tool to a menu

After you create a tool, you must add the tool to a menu. A menu contains one or more of the following items:

- Tool
- Menu
- Separator

The Web GUI Active Event List and Event Viewer contain a single menu. The menu is called **alerts**. The alerts menu contains menus, tools, and separators. When you add a tool to Web GUI, you add that tool to the alerts menu, or a custom menu that you create.

To add an item to a menu, you open the Administration feature and select **Menu Configuration**. The list of available menus opens. From here, you can create a menu, or modify an existing menu. To add a tool to the default menu, select **alerts**, and click **Modify**. The alerts menu configuration opens.

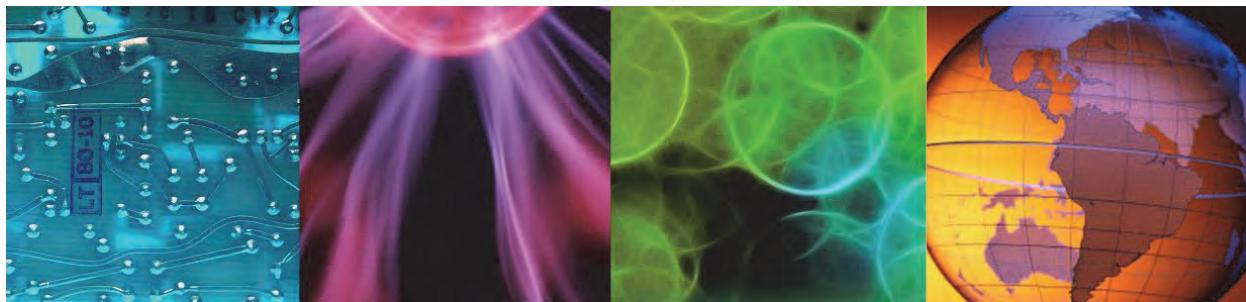
The list of available tools appears on the left. The entries in the alerts menu appear on the right. The pull-down list under Available Items is used to select either menus, tools, or the separator. Click a tool on the left and click the *right arrow* icon to add it to the menu. The tool appears on the bottom of the menu. You use the up, and down arrows to move the tool within the menu. You can change the text that appears on the user desktop for the tool.



Lesson 3 Maps



Lesson 3 Maps



© Copyright IBM Corporation 2016

In this lesson, you learn how to perform the following tasks:

- Create a simple map
- Add the map to a page

Visualizing high-level event information

- In environments with high numbers of alerts, event visualization components provide an overview of the available data.
- The following event visualization components provide a high-level view of the alerts:
 - Use charts to represent event information against scales that indicate the values of the information
 - Use event dashboards to represent multiple SQL queries against the alerts tables of the ObjectServer
 - Use maps to obtain an interactive representation of a network
- Maps are web applets that you can use to create a view of your infrastructure
 - Use maps for a network topology or geographical overview
 - You can overlay a background image with interactive widgets called map objects
 - The maps can be configured to display alert data, and to open event lists for a detailed view of the network

© Copyright IBM Corporation 2016

28

Visualizing high-level event information

Maps provide a convenient mechanism for portraying the status of your infrastructure in a graphical manner. For example, you can use a map of the United States, with color-coded boxes to represent the status of geographically distributed resources.

Maps: General information

- Map resources are:
 - Background graphics for the map
 - Icons that you want to use as map objects
 - Default graphics and icons are provided
 - You can upload more background images and icons as map resources
 - Map resources can be .gif files, .jpeg files, or .png files
- Map objects are widgets that you can put on a map and can be:
 - Buttons, lines, or icons
 - Customized for appearance and behavior
- The types of map objects are:
 - Active objects that can:
 - Display alert severity information.
 - Be associated with a filter to show the highest severity status alert that is captured by the filter
 - Have hover for help for the object, which displays information from the filter.
 - Inactive objects,
 - Can be associated only with the URL of another web page, for example, a page that contains another map.
 - Use a text object to write text directly onto a map.

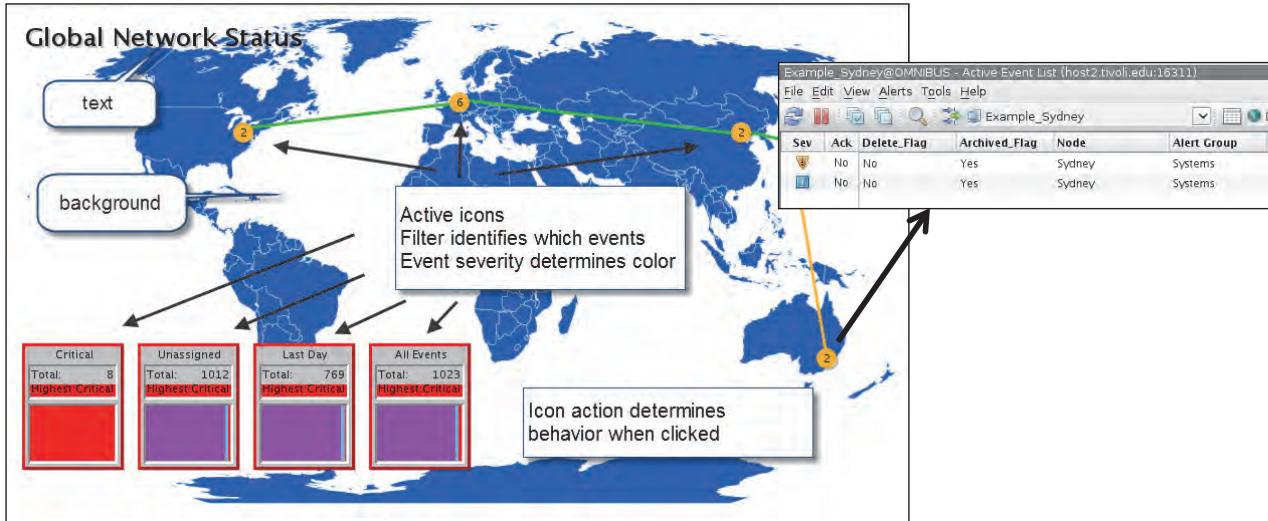
Maps: General information

A map consists of a number of functional components. One component is the map resource. A map resource is an image file. The file is used to display a background image for the map, like a map of the United States. The resource also provides a graphic for a display object, like a device or a building.

The primary building block of the map is the icon. There are two primary icon types. The first type is the *active* icon. An active icon is used to implement some form of action that occurs when a user clicks the icon. An *inactive* icon, is static, and does not provide any action.

In general, a map consists of a background image, with one or more active, and inactive icons that are overlaid on the background.

Sample map



© Copyright IBM Corporation 2016

30

Sample map

The sample map that is shown here illustrates most of the types of map objects. The map contains a background image and a title. The title is a form of inactive icon. The circular objects, and rectangular objects are active icons. The color of each active icon is determined based on the severity of events that map to the corresponding icon based on a filter.

Creating a map

Administration

- Event Management Tools
 - Filters
 - Views
 - Relationships
 - Tool Configuration
 - Tool Prompt Configuration
 - Menu Configuration
 - Metrics
 - Data Sources
- Maps
 - Map Creation**
 - Map Resources
 - CGI Registry
 - Event Database Query

Create a map

1. Enter name
2. Define background image
3. Select icon type
4. Click map to add one or more icons
5. Drag an icon to move and resize

Java Map Editor - MyMap*

Map Name: MyMap

Access Control: *

Map Size: Width 636

Background: background
white
europe.png

Items: visual_1, Moscow, Rome

Global Network Status

Untitled

click to add icon to map

click to select icon type

© Copyright IBM Corporation 2016

31

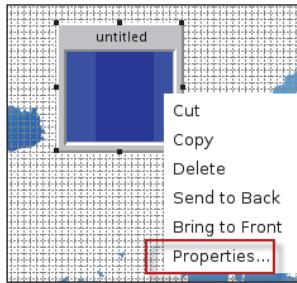
Creating a map

To create a map, you open the Administrative feature, and select **Map Creation**. The list of available maps opens. You click the icon to create a new map. You enter a name for the map, which cannot contain spaces or special characters except the underscore. The map editor opens.

You typically define a background color and image for the map first. Then, you adjust the values for the height and width of the map. Then, you add icons.

On the bottom of the map editor page, there are a series of icons. Each one relates to a type of map icon. The gray icons are inactive. The colored icons are active. You click an icon to select the type, and then click the map to create the corresponding icon on the page. Click as many times as necessary to add the required icons. Drag the icon to change the position, and resize the icon as necessary. Next, you configure the properties for each icon.

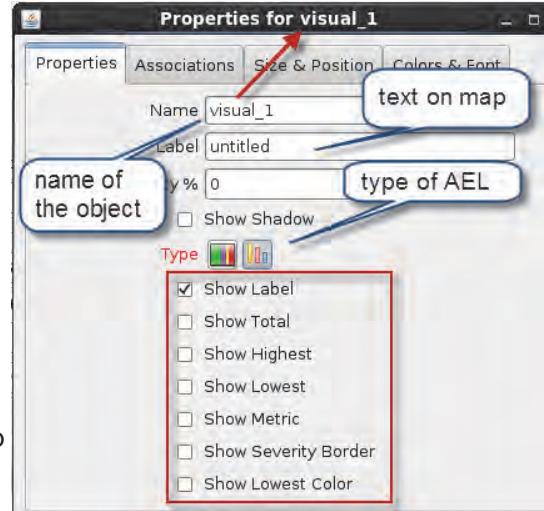
Icon properties



Properties vary by type of icon

Right-click icon and select Properties

- Name is the name of the icon object within the map
- Type defines type of AEL: Lava lamp, histogram
- More display characteristics



© Copyright IBM Corporation 2016

32

Icon properties

Right-click the icon, and select **Properties**. Icon properties vary based on the type of icon. The slide shows an example of the Active Event list icon. The Active Event List icon is an active icon.

The map editor generates the icon *Name* value. The *Label* can be used to enter text that appears on the top of the icon.

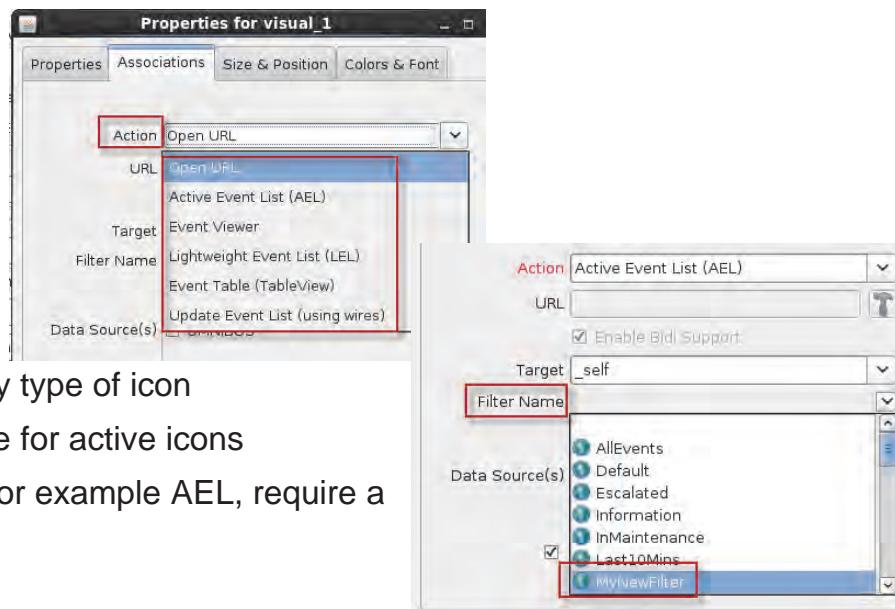


Hint: You can use a text icon instead of the Label to provide descriptive text for the icon. The text icon can be placed anywhere on the map, and you can configure the size, shape, and color of the text.

You can optionally configure the transparency for the icon. The default is 0, which indicates that the icon is opaque. A value of 100 causes the icon to be invisible.

The Active Event List icon appears as a monitor box on the map. The remaining parameters control the appearance of the monitor box. Select the type of monitor box: lava lamp or histogram.

Icon associations



Associations vary by type of icon

Action only available for active icons

Event List actions, for example AEL, require a filter

Icon associations

The **Associations** tab is used to define the major characteristics of the icon. Select the type of action from the pull-down list. The possible choices are as follows:

- Open URL: The configured URL opens when the icon is clicked.
- Active Event List (AEL): The AEL opens when the icon is clicked.
- Event Viewer: The Event Viewer opens when the icon is clicked.
- Lightweight Event List (LEL): The LEL opens when the icon is clicked.
- Event Table (TableView): The TableView opens when the icon is clicked.
- Update Event List (using wires): For this option, a system or custom wire must be configured to an Event Viewer or Active Event List.

Select a filter to associate with the event list action.

Icon associations, continued

The screenshot shows a configuration panel for an action named "Active Event List (AEL)". The "Target" dropdown menu is open, displaying four options: "_self", "_blank", "_parent", and "top". The option "_self" is selected and highlighted with a red border.

Target defines where the AEL open, for example, _blank indicates a new window

The screenshot shows the same configuration panel. The "Data Source(s)" dropdown menu is open, displaying two entries: "OMNIBUS" (with a checked checkbox) and "London" (with an unchecked checkbox). The entry "OMNIBUS" is highlighted with a red border.

Data Sources(s) used to select one or more configured ObjectServers

- Must be defined to Web GUI
- One data source entry can be a high availability pair of ObjectServers

© Copyright IBM Corporation 2016

34

Icon associations, continued

Select a target for the output of the action from one of the following choices:

- blank: The event list opens in a new browser window.
- parent: The event list opens in the parent frame set that contains the source link.
- top: The event list opens in the frame that contains the source link.
- self: The event list opens in the full current browser window, and *replaces* the map.

Select the ObjectServer from the list of data sources. You can select more than one ObjectServer if the Web GUI is configured with access to more than one ObjectServer.

Testing the map

The screenshot shows the 'Available maps:' list with 'MyMap' selected. Below it, the 'Editor Initialized' status is shown with Java selected. At the bottom are buttons for New, Modify, Delete, and Preview. A red arrow points from the 'Preview' button to the second screenshot.

Map : MyMap - Mozilla Firefox
 https://host2.tivoli.edu:16311/bm/console/webtop/Map/MyMap?enable tooltips=true&textd
Global Network Status

The preview window displays a map of Europe with a small inset titled 'untitled'. A red arrow points from the 'untitled' inset to the third screenshot.

MyNewFilter@OMNIBUS - Active Event List (host2.tivoli.edu:16311)
 https://host2.tivoli.edu:16311/bm/console/webtop/AELV/MyNewFilter@OMNIBUS - Active Event List (host2.tivoli.edu:16311)

Serial	LastOccurrence
10633	10/23/14 6:35:19 PM
9451	10/23/14 6:26:49 PM
13772	10/23/14 7:34:19 PM
22946	12/2/14 7:57:06 PM
12823	10/23/14 7:33:26 PM

Select map and click Preview

Click map icon

Active Event List open in new window

- Icon properties define the filter: MyNewFilter
- The filter defines the view

© Copyright IBM Corporation 2016

35

Testing the map

After the map updates are complete, you save the map definition. The map name appears in the list of available maps. Click the name of the map, and then click **Preview**. A browser window opens, and the map appears. The active icons reflect the status based on the events that are found in the ObjectServer that match the corresponding filter criteria. If you click an active icon, the corresponding event list opens. The event list contains the events that meet the filter criteria.

You can open the map in the map editor and make any necessary adjustments.

Adding a map to a page

The screenshot shows three panels of the console interface:

- Console Settings (Left Panel):** Shows the navigation tree with "Pages" selected.
- Pages (Middle Panel):** A list of pages with buttons for "New Page..." and "New Folder...".
- Page Settings (Right Panel):** A form to create a new page named "MyNewPage" with "Freeform" selected as the page layout.

A red arrow points from the "New Page..." button in the Pages panel to the "Edit properties" button on the Map portlet in the Page Settings panel.

Create a page, which requires *iscadmins* role

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

© Copyright IBM Corporation 2016

36

Adding a map to a page

After the map is tested, and it behaves as expected, you must add the map to a Dashboard Application Services Hub page. A user accesses the map through this page.

Open the Console Settings feature and select **Pages**. A user requires the *iscadmins* role to access the Console Settings feature. The list of available folders opens. Pages are stored in folders. Click **New Page**. Enter a name for the page, and select the type of page. You can change the default location to save the page if necessary. The page editor opens.

Locate the **Map** widget and drag the widget to the page layout area. Move and resize the widget. Click the arrow and select **Edit** to edit the properties for the widget. The property editor opens.

Adding a map to a page, continued

Select map name in widget properties, and save

Save the page

Access the page

The screenshot shows the 'Map - Edit Preferences' dialog with 'MyMap' selected in the 'Map name' dropdown. To the right is a preview window titled 'MyNewPage' showing a map titled 'Global Network Status Europe'. Below these is a browser window titled 'MyNewFilter@OMNIBUS - Active Event List' displaying a table of event logs.

Serial	LastOccurrence	Node	Summary
10633	10/23/14 6:35:19 PM	host1.tivoli.edu	rrtrapd probe on host1.tivo
9451	10/23/14 6:26:49 PM	host1.tivoli.edu	syslog probe on host1.tivo
13772	10/23/14 7:34:19 PM	host1.tivoli.edu	Probe Going Down ... (Pro
22946	12/2/14 7:57:06 PM	host2.tivoli.edu	simnet probe on host2.tivo
12823	10/23/14 7:33:26 PM	host1.tivoli.edu	Probe Going Down ... (Pro

© Copyright IBM Corporation 2016

37

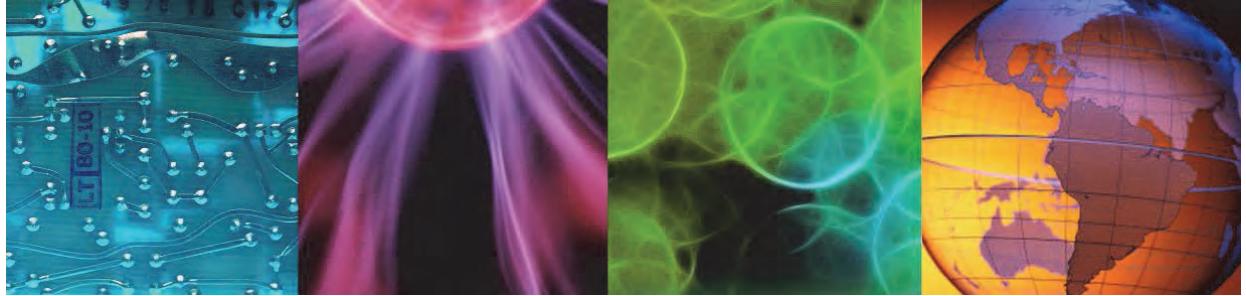
Adding a map to a page, continued

Select your map from the list of available maps, and save the property settings. Then, save the page, and exit the page editor. If you did not change the location when you created the map, the map is saved in the **Default** folder. Click the icon to open the folder, and select the page. The page opens, and the map appears.



Lesson 4 Gauges

Lesson 4 Gauges



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Configure a gauge metric
- Use the metric to define a gauge
- Add the gauge to a page

Gauges

Another aspect to visualizations is the gauge

A query of an ObjectServer table determines the content of the gauge

Netcool Web GUI includes sample gauge usage

An administrator can create more



39

Gauges

A gauge is a graphical display that portrays the value of one metric. The metric is computed based on an SQL query of an ObjectServer table. The query runs continuously based on a frequency, like every 5 minutes. You configure what data is retrieved, and how the data is used to compute the metric. The metric can be as simple as the count of records in a table. Or, the metric can be something like the average of a number of data points.

Netcool/OMNIbus includes several pages of gauge examples. The examples present the near-term values for various key performance indicators. The key performance indicators can be used to determine the overall health of the Netcool/OMNIbus environment.

Characteristics of a gauge

Gauges Layout



Type determines the appearance of the gauge

- There are six options currently available

Metric defines the SQL query for the ObjectServer table

- Metric is defined separately

© Copyright IBM Corporation 2016

40

Characteristics of a gauge

A gauge consists of a number of configurable characteristics. One of these characteristics is the type of graphic that is used for the gauge. A gauge can represent three levels of metric state: good, marginal, or bad. The state is determined based on threshold values. The graphic for the gauge varies based on the current state of the metric.

The choices for type of gauge are as follows:

- Dial
- Weather
- Emoticon
- Traffic Lights
- Thermometer
- Status Button

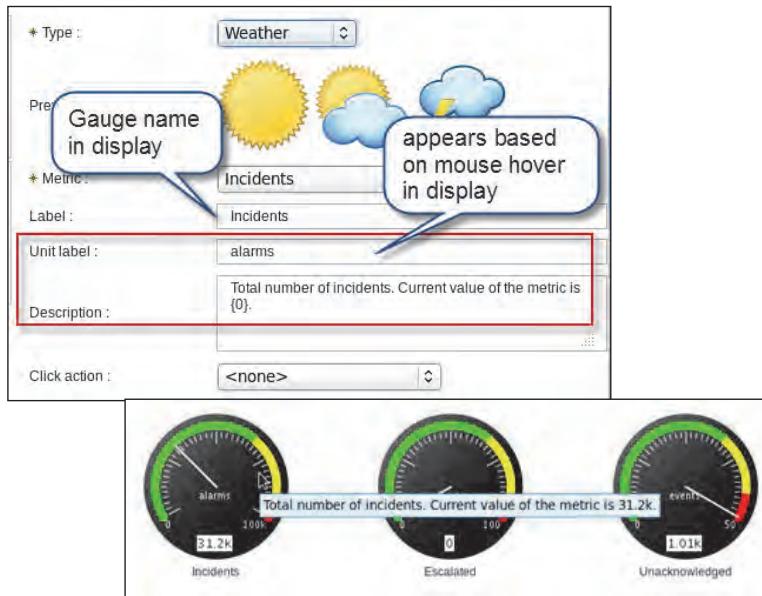
Characteristics of a gauge, continued

Unit label identifies the unit of the metric

Description text appears when the mouse hovers over the gauge

Click action defines optional action that occurs when gauge is clicked

- Run script
- Send an event to another widget
- Open a URL



© Copyright IBM Corporation 2016

41

Characteristics of a gauge, continued

The Label contains the textual name for the gauge, which appears in the user view.

The Unit label identifies the units of the metric, for example, percentage, count, events, and others.

The text in Description appears when the user hovers the mouse over the gauge.

You can optionally define an action that occurs when the gauge is clicked. The choices are as follows:

- Run a script
- Send an event to another widget
- Open a URL



Note: The event action is specific to Dashboard Application Services Hub widgets. One widget generates an event. Another widget takes some action when the event is generated.

Creating a metric



Definition contains

- Unique metric name
- Label for display
- Minimum and maximum display values
- Data range for bad condition
- Data range for warning condition
- SQL query

The 'Create New Metric' dialog box. It includes fields for 'Name' (TotalEvents), 'Display Name' (Event Count), 'Description', 'Units' (events), 'Maximum Value on Gauge' (10000), 'Minimum Value on Gauge' (0), 'Upper Threshold' (80), and 'Lower Threshold' (41). There's also a 'SQL Query' section with the query: 'select count(Serial) from alerts.status;'. Buttons at the bottom include 'Create Metric' and 'Cancel'.

© Copyright IBM Corporation 2016

42

Creating a metric

To create a metric definition, open the Administration feature, and select **Metrics**. The list of available metrics opens.



Hint: The existing metrics provide examples of how to query various tables in the ObjectServer. Use these samples as the basis for creating your own metric definitions.

Enter a unique value for the metric name, which cannot contain spaces or special characters, except for the underscore. The display name, and description values are used when the gauge is defined. These values are the same values that are discussed in the previous slide.

The use of the remaining values varies based on the type of gauge display. For example, when you portray a gauge with the Dial graphic, there is a scale on the face of the dial. The minimum and maximum values in the metric definition are used to configure the scale on the dial. These same values do not have any relevance when you use a different gauge graphic, like the Status Button.

The threshold settings apply to all gauge types because they establish the state of the metric. You have two options for how to configure the threshold values. You can enter a specific value for the metric, or you can select a percentage. The percentage is applied to the value for the maximum gauge value, and the result appears in the definition. The percentage option is convenient if you anticipate the need to adjust the maximum value for the gauge scale.

Adding a gauge to a page



Edit the page

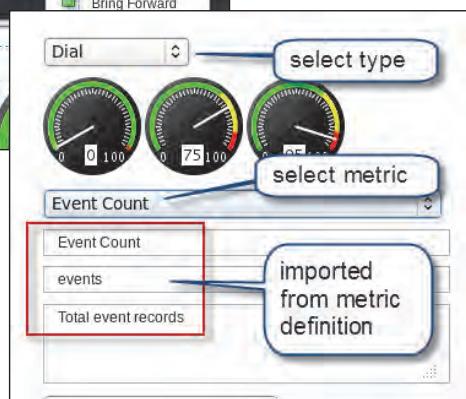
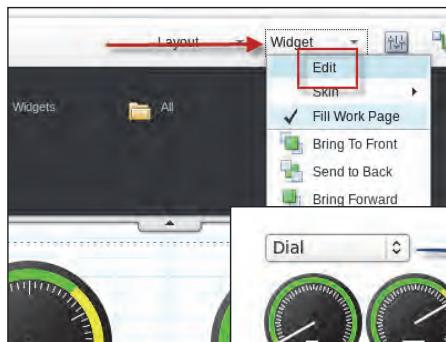
Edit the widget preferences

Select gauge display type

Select metric from list

- Text values are imported from metric definition

Click Add gauge



© Copyright IBM Corporation 2016

43

Adding a gauge to a page

You have two choices for how to use a new gauge. One choice is to add the gauge to one of the included pages. This technique is probably the easiest because the page exists, and you need only to modify the page. The second choice is to create a new page, and add the gauge. If you decide to use an existing page, it is suggested that you make a copy of the page, and save the copy with a new name. By copying the page, you eliminate any issues if the existing page is modified in a subsequent software upgrade.

You must use a user with the *iscadmins* role to add or modify a page. To edit an existing page, open the page. Then, click the *page actions* icon in the right corner of the page, and select **Edit Page**.

The page opens in the page editor. There is a small, faint blue line near the top of the page. Click the line, and a dotted line appears around the page. At the top of the page click **Widget**, and select

Edit. The widget property editor opens.



Note: The page that with the gauge displays contains a single widget. You must click the blue line to select the widget on the page.

Scroll to the right in the widget property editor. Select a gauge type, and select the metric from the pull-down list. The remaining values are populated from the metric definition. Scroll down and click **Add Gauge**. The gauge is added to the widget.

Adding a gauge to a page, continued



Drag gauge to change location
Save the page

© Copyright IBM Corporation 2016

44

Adding a gauge to a page, continued

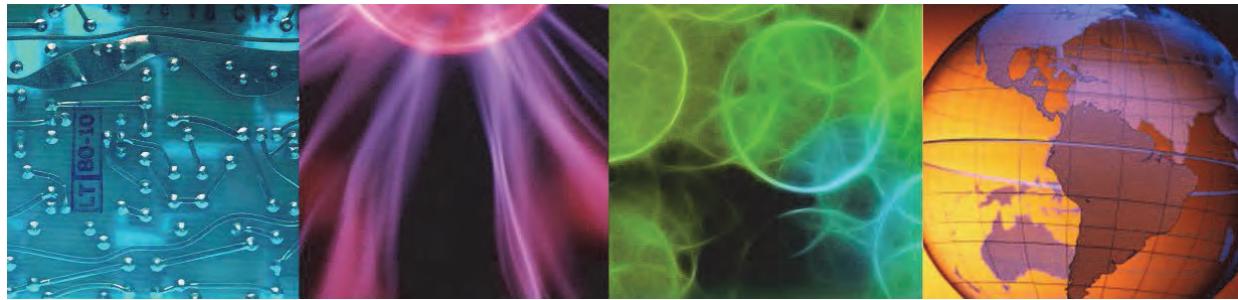
After the gauge is added to the widget, it appears in the list of widgets. Drag a gauge to change the location. Click the X in the corner of a gauge to remove it from the widget. When complete, click **Save** to save the widget property settings. Then, click **Save and Close** to save the page, and exit the page editor. The updated page opens.



Lesson 5 Basic dashboard creation



Lesson 5 Basic dashboard creation



© Copyright IBM Corporation 2016

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 a simple dashboard page.

Dashboard defined

A dashboard is a Dashboard Application Services Hub *page*

The user opens and views a *page*

The page contains one or more *widgets*

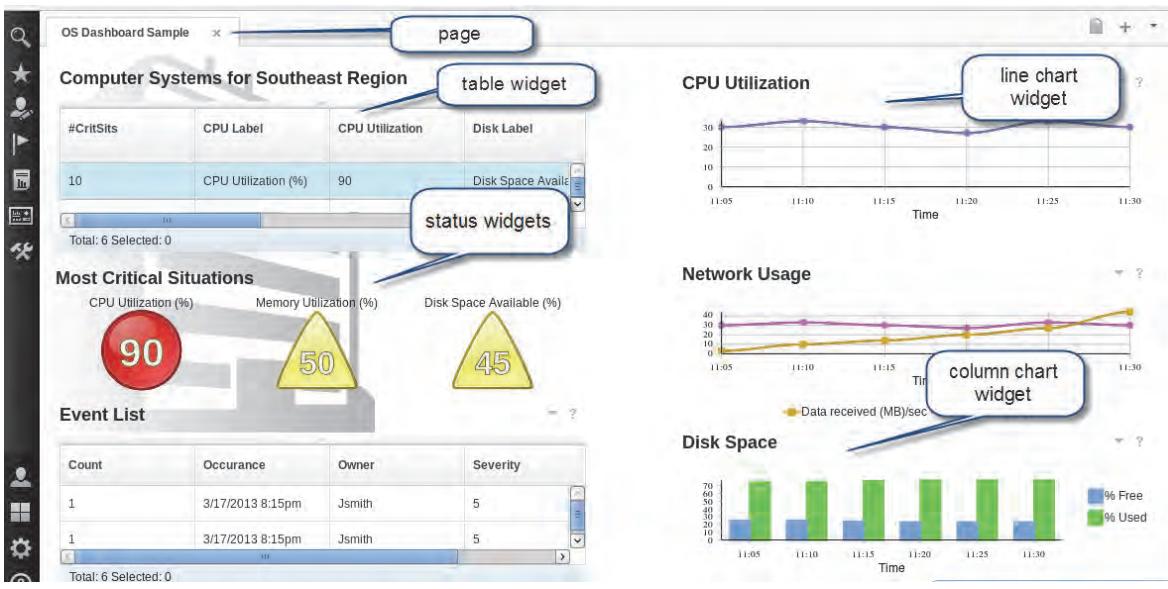
A *widget* is an application that displays data, such as:

- Event data
 - Active Event List
 - Event Viewer
 - Event dashboard
 - Web GUI map
- Non-event data
 - Gauge
 - Chart

Dashboard defined

Jazz for Service Management is a framework that facilitates the integration of data from various applications. The system that is used in the class integrates Netcool/OMNibus event data, by using Web GUI, and Tivoli Common Reporting. In a production environment, generally many more applications are incorporated within the Jazz for Service Management framework. The visualization component is called Dashboard Application Services Hub. A dashboard, in the context of this lesson, is a page that contains data from one or more applications. The data is displayed on the page through widgets. A widget is an application that presents data in some form. For example, there are table widgets, which display in tabular form. Charting widgets are used to present the data in various styles of graphs, such as: pie chart, bar chart, and line chart.

Sample page



© Copyright IBM Corporation 2016

47

Sample page

This slide shows a screen capture of a sample dashboard page. Dashboard Application Services Hub includes several examples of dashboard pages. This example page uses several types of widgets, including table widget, status widget, line chart widget, and column chart widget.

Creating a page

User requires *iscadmins* role to create or change a page

Enter page name

Select location to save page

Page Settings dialog:

- * Required field
- * Page name: MyNewPage
- Page location: console/Default/
click to change location
- Page Layout:
 - Proportional - Place and overlay widgets anywhere that will scale on work page.
 - Freeform - Place and overlay widgets anywhere on work page.
 - Fluid - Tile widgets on the page. Great for mobile.
- Optional setting

© Copyright IBM Corporation 2016

48

Creating a page

The first step in creating a dashboard is to create a Dashboard Application Services Hub page. The user who creates the page must have the *iscadmins* role.

Hint: The **smadmin** user has the *iscadmins* role by default. However, the smadmin user does not have any of the Web GUI roles, like: *ncw_admin* or *ncw_user*. Therefore, the smadmin user can create a page, but the user cannot use any of the widgets that access the ObjectServer. The **ncoadmin** user has access to the ObjectServer, and has the *iscadmins* role. Therefore, the ncoadmin user can create a page that contains Web GUI widgets.

To create a page, you open the Console Settings feature and select **Pages**. The list of available pages opens. Pages are organized into folders. To create a folder, click **New Folder**. To create a page, click **New Page**. The Page Settings page opens.

Enter a value for the page name. Page names *can* contain spaces. You can change the location where the page is saved if necessary. The default location is a folder that is called **Default**.

Creating a page, continued

The screenshot shows the 'Create Page' dialog. On the left, under 'Page Layout', the 'Freeform' option is selected. Below it, the 'Optional setting' section is expanded, showing an 'Add...' button. On the right, the 'Available Roles' panel lists three roles: 'administrator', 'all authenticated portal users' (which is checked), and 'chartAdministrator'. A red arrow points from the 'Add...' button to the 'all authenticated portal users' row.

Select page layout

Expand Optional setting

Add one or more roles to grant access to the page

© Copyright IBM Corporation 2016

49

Creating a page, continued

Select the type of page layout. The type of page layout controls how widgets are placed on the page, with the following choices:

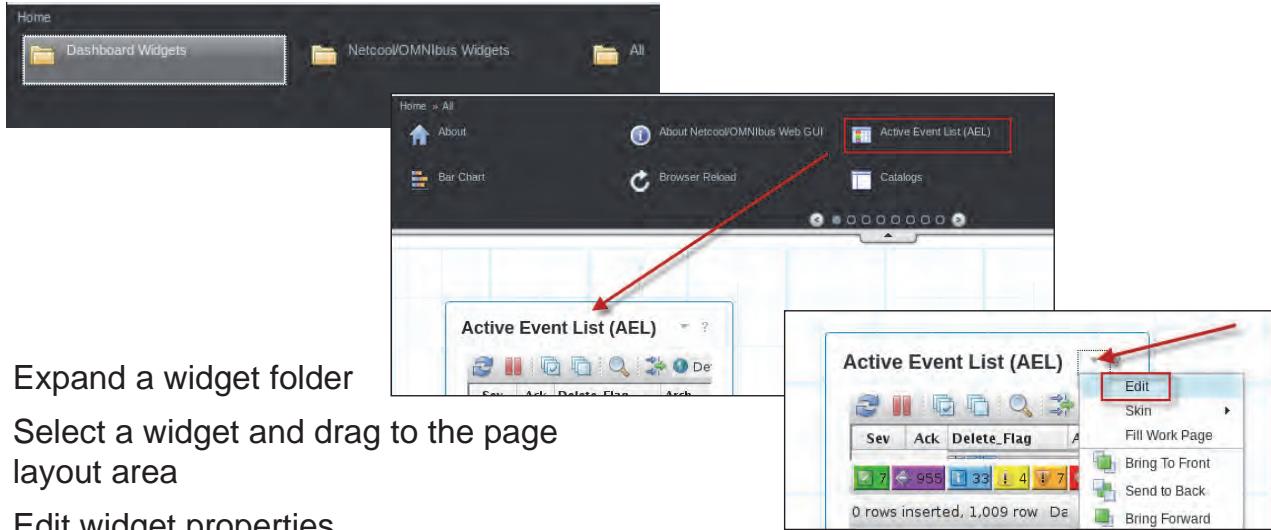
- Proportional: Widgets can be positioned anywhere. Widgets proportionally resize based on the resolution of the screen that is used. Widgets can be overlaid. Proportional layout does not support scrolling.
- Freeform: Widgets can be positioned anywhere. Widgets can be overlaid. Freeform dashboards scroll horizontally or vertically, depending on the size of screen that is used.
- Fluid: Widgets reorder their position on the dashboard, depending on the type of screen. You cannot overlay widgets. This layout type supports vertical scrolling only. When a screen is narrow, widgets are displayed vertically. This layout type is the most flexible for mobile device screens, for example, tablets and smartphone screens.

Expand **Optional setting**, and click **Add**. You use this feature to control access to the page. You grant access to the page by selecting one or more roles. To allow access by all user, select *all authenticated portal users*. Click **OK** to save the access information. Click **OK** to open the page editor.



Important: The roles that you assign here control access to the page. A user with access to the page might need more roles to view the data on the page. A user requires a Web GUI role to see event data. If a user that does not have a Web GUI role opens a page that contains event data, the page opens, but access to the event data is denied.

Adding a widget to a page



© Copyright IBM Corporation 2016

50

Adding a widget to a page

The page editor has two functional areas. The top of the editor contains the available widgets. The widgets are organized into folders. You click a folder to view the contents. The lower portion of the page editor contains the page layout area. To add a widget to a page, you locate the widget, and drag the widget into the page layout area. As you drag the widget into the layout area, you might see the widget turn red. The red color is an indication that you cannot drop the widget in the current location. Continue dragging the widget until it turns green. When the widget is released, it snaps to the nearest grid lines. You can drag the widget to change location. You can click an edge of the widget, and drag the edge to resize the widget.

Every widget has a collection of properties. The properties vary by type of widget. Click the arrow, and select **Edit** to configure widget property values.

Widget properties

Widget properties vary by widget type

For event list widget types, like Active Event List

- Select event filter
- Select event view
- Select data source

Enter text for widget title

Save changes

Active Event List

General Settings

Show System Filter : MyNewFilter

Filter : **select filter**

Filter SQL :

View : **select view**

Transient filter name :

Event list single-click action : <none>

Event list double-click action : Information

Data Sources

Select | Name

OMNIBUS

Appearance of the AEL

Portlet Title : **Active Event List**

Use Customizer :

A yellow callout bubble points from the "title on page" label to the Portlet Title field.

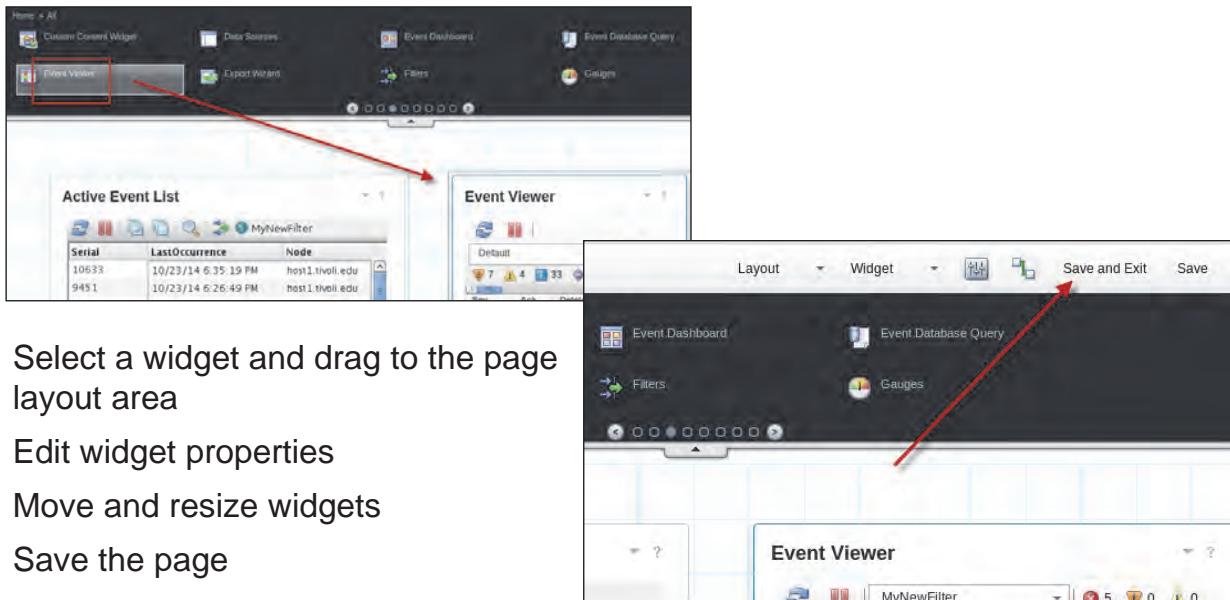
Widget properties

The example that is shown here is based on the Active Event List widget. For this widget, you must select a filter, a view, and an ObjectServer. Enter text for the portlet title. This text appears on the page. Click **Save** to save the settings.



Important: Any user that views the page with the Active Event List widget must have one of the Web GUI roles: *ncw_admin* or *ncw_user*. This requirement is true for all event list widgets.

Adding another widget to a page



Select a widget and drag to the page layout area

Edit widget properties

Move and resize widgets

Save the page

Adding another widget to a page

After you save the properties for the Active Event List widget, the event data appears on the page. The event records are limited based on the filter, and the view defines the column names.

To add another widget to the page, you repeat the previous steps. Locate the widget, drag the widget to the page, and configure the properties. You can add as many widgets as necessary.

You can add a title for the page with the **Text** widget. You can add a background to the page with the **Image** widget.

When the page is complete, click **Save and Close**. The page editor closes, and the finished page opens.

Accessing the page

The screenshot shows the Dashboard Application Services Hub interface. On the left, there is a navigation tree with a folder icon labeled 'MyNewPage' highlighted with a red box and a red arrow pointing to it from below. The main area contains two windows: 'Active Event List' and 'Event Viewer'. The 'Active Event List' window shows a table with columns 'Serial', 'LastOccurrence', and 'Node', containing five rows of event data. The 'Event Viewer' window shows a table with columns 'Group', 'Count', 'Sev', 'Serial', and 'LastOccurrence', also containing five rows of event data.

The default location is the **Default** folder

© Copyright IBM Corporation 2016

53

Accessing the page

A user can find the page in the **Default** folder, unless you saved the page to a different location. Users can add the page to their startup pages. Any page in the list of startup pages opens when the user logs in to Dashboard Application Services Hub.



Lesson 6 Web GUI administrative API

Lesson 6 Web GUI administrative API



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Install the Web GUI administrative API client
- Configure the client
- Use the client to perform Web GUI administrative functions

Using the WAAPI client

Use the Web GUI Administration Application Program Interface (WAAPI) client to locally or remotely deploy configurations and settings

The WAAPI client is a Java based utility

Configuration instructions are written in XML and are stored in a text command file

The WAAPI client sends the command file directly to the web GUI server

The WAAPI client authenticates against the server

The WAAPI client is installed with the Web GUI server by default

The client can be installed manually on a user workstation

Using the WAAPI client

The Web GUI administrative API is used to modify an instance of Web GUI through a command-line interface. A client application provides access to the API. The client reads configuration commands from an XML file and applies the changes to the target Web GUI server. The Web GUI server can be running on the same system as the client or on a remote system.

Many of the Web GUI administrative functions that are discussed previously in this unit can be applied with the Web GUI API client. For example, you can use the client to create filters and views. You can use the client to create tools, menus, and prompts.

Installing the WAAPI client on a remote host

To install the WAAPI client on a remote host, follow these steps:

1. Open a browser and connect to Dashboard Application Services Hub
2. Log in with a user with the *ncw_admin* role
3. Scroll to the bottom of the page and expand **WAAPI Client Information**.
4. Download the WAAPI client:
 - a. Click **waapi.tar.gz** for a UNIX workstation
 - b. Click **waapi.zip** for a Windows workstation
5. Expand the file

Configure the WAAPI client

1. `cd <install directory>/waapi/etc`
2. `gedit waapi.init`
3. Set the following values
 - `waapi.host:<Web GUI server name or IP address>`
 - `waapi.port:16310`
 - `waapi.user:<Web GUI user with ncw_admin role>`
 - `waapi.password:<password>`

© Copyright IBM Corporation 2016

56

Installing the WAAPI client on a remote host

The Web GUI API client software is included with Web GUI. To download the installation files, you open a browser, and log in to Dashboard Application Services Hub. In the lower portion of the Welcome page, you see **WAAPI Client Information**. Click the arrow to expand the list. Two files appear in the list. One file is used for Windows, and the other is used for UNIX. Click the appropriate file name, and specify the directory to save the file.

The client software is packaged as a compressed file. Expand the file with the appropriate utility based on the local operating system.

The client requires access information for the target Web GUI server. You can configure the access information in a property file, or provide the access details with command-line parameters when you run the client. The client requires the host name of the Web GUI server, the port number, a user name, and password. The user must be a valid Dashboard Application Services Hub user. The user must also have the Web GUI administration role, *ncw_admin*, assigned.

Using WAAPI instructions to administer the server

WAAPI instructions are contained within an XML command file and passed to the server with the WAAPI client

You do not have to restart the server after the commands are issued

As an administrator, you should have some XML knowledge

Sample WAAPI command files for typical administrative activities are provided with the WAAPI client and are in directory

```
/opt/IBM/netcool/omnibus_webgui/waapi/etc/samples
```

If you installed the WAAPI component on a system that does not have a Web GUI server installation, use the sample files that are in

```
<install_dir>/etc/samples/
```

Using WAAPI instructions to administer the server

The client software includes a collection of sample XML files. You can use the sample files as the basis to create your own administration files. The following list is a few of the sample files:

```
list_entity.xml  
list_filtercollection.xml  
list_filter.xml  
list_map.xml  
list_menu.xml  
list_metric.xml  
list_resource.xml  
list_tool.xml  
list_user.xml  
list_view.xml
```

Each sample file is documented with comments that describe what functions are provided by the file. For example, the samplerequest_tool.xml file contains the following text:

```
<!--  
This WAAPI command file contains samples to:  
- list tools in the server  
- delete tools  
- create tools  
- create or replace tools  
- modify tools  
-->
```

Starting the WAAPI client

Start the WAAPI client:

- If the WAAPI client is on a computer with a Web GUI installation:

```
/opt/IBM/netcool/omnibus_webgui/waapi/bin/runwaapi options
```

- If the WAAPI client is on a computer as a stand-alone utility:

- Set the `JAVA_HOME` environment variable to `jre_install_dir`.
`<install_dir>/waapi/bin/runwaapi options`

© Copyright IBM Corporation 2016

58

Starting the WAAPI client

You run the client with the following command:

```
/opt/IBM/netcool/omnibus_webgui/waapi/bin/runwaapi options
```

If you did not configure the Web GUI access information in the `waapi.init` file, you can provide the parameters when you run the client with the following command:

```
/opt/IBM/netcool/omnibus_webgui/waapi/bin/runwaapi -host <hostname> -port <port number> -user <user name> -password <password>
```

You must provide the path to the XML command file, for example:

```
/opt/IBM/netcool/omnibus_webgui/waapi/bin/runwaapi -file  
/opt/IBM/netcool/omnibus_webgui/waapi/etc/samples/webtop_report.xml
```

Sample output from webtop_report.xml

```
inetcool@host2 bin]$ ./runwaapi -file ..\etc\samples\webtop_report.xml | more
*****
WAAPIClient: Request sent to server on http://localhost:16310/ibm/console/webtop/...
Mon Dec 08 21:37:37 UTC 2014
*****
WAAPIClient: 1 method returned errors during execution:
Method 'webtopprobe.generateReport' returned:
*****
Tivoli Netcool/OMNIBus Web GUI DATA REPORT START
*****
Mon Dec 08 21:37:37 UTC 2014
Tivoli Netcool/OMNIBus Web GUI Server Version
*****
Tivoli Netcool/OMNIBus Web GUI: 8.1
IBM Tivoli Netcool/OMNIBus V8.1 Web GUI build 493 (16:50 Jun 03 2014)
Java Runtime Environment
*****
Java version: 1.6.0
Java location: /opt/IBM/WebSphere/AppServer/java/jre
Memory Usage
*****
JVM total memory: 655360K
JVM free memory: 240328K
Platform Information
*****
Platform: Linux amd64
Platform version: 2.6.32-279.22.1.el6.x86_64
```

© Copyright IBM Corporation 2016

59

Sample output from webtop_report.xml

The slide shows a portion of the output from the webtop_report.xml file.

Student exercises



Perform the exercises for this unit.

Summary

You now should be able to perform the following tasks:

- Create event filters and views
- Configure event grouping
- Create desktop tools
- Configure and create map pages
- Add a gauge and define thresholds
- Configure and use the Web GUI administrative API client



6 User administration



User administration



© Copyright IBM Corporation 2016

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

In this unit, you learn how to create users and groups.

References: SC27-6265-00 *Netcool/OMNibus Version 8 Release 1 Administration Guide*,
SC27-6505-00 *Web GUI Administration, and User's Guide*

Objectives

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

- Describe ObjectServer user administration
- Create ObjectServer users, groups, roles, and restriction filters
- Configure ObjectServer to use LDAP for password authentication
- Describe Web GUI user administration
- Create Web GUI users, roles, and groups
- Configure Web GUI user preferences
- Create default startup pages



Lesson 1 User administration overview



© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this lesson, you learn how you define users in the ObjectServer, and Web GUI.

Overview

User administration occurs in multiple components

- ObjectServer
- Web GUI
- Dashboard Application Services Hub
- WebSphere

Where user administration occurs is a function of user access requirements

User requires access to event records with the Native Desktop

- User must have a user ID defined within the ObjectServer

User requires access to event records with Web GUI

- User must have a user ID defined within the ObjectServer
- User must also have a user ID defined within WebSphere
- WebSphere user ID must have appropriate Dashboard Application Services Hub roles assigned

Overview

User information is defined in various components. The type of access generally determines where user information is defined. Any user that accesses the ObjectServer with the Native desktop must have a user ID that is defined in the ObjectServer. A user that accesses event records with the Web GUI component must have a user ID that is defined in the ObjectServer and a WebSphere Virtual Member Manager user repository.

ObjectServer user administration

Default ObjectServer contains two users

- root: System level user, with no password initially configured
- nobody

User definitions are stored in ObjectServer tables

- Security database

Password authentication source

- ObjectServer: Passwords are stored internally
- External: One of the following, 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 against the defined target (Operating System or LDAP)

User administration is configured with Netcool/OMNIbus Administrator utility

\$OMNIHOME/bin/nco_config

© Copyright IBM Corporation 2016

5

ObjectServer user administration

The Netcool/OMNIbus Administrator utility is used to create users, and groups in the ObjectServer. The user information is saved in ObjectServer tables. The administrator can decide how the ObjectServer performs password authentication. Two options are available. One option stores the password in an ObjectServer table. The second option uses an *external* source for password authentication. There are two options for an external source. One option uses the local operating system. The second option uses an LDAP server.

Netcool/OMNIbus Administrator utility

Users are managed from the **Users** tab

Note the two users

- root
- nobody

The screenshot shows the 'User' configuration screen for the PRIMARY instance on omnibhost:4100. The left sidebar has tabs for 'User', 'Groups', 'Roles', and 'Restriction Filters'. The main area is titled 'Users' and contains a table with two rows:

Name	Full Name	Type	ID	Extern...	Enabled
nobody	Nobody	Unknown	65534	<input checked="" type="checkbox"/> false	<input checked="" type="checkbox"/> false
root	Root User	Super User	0	<input checked="" type="checkbox"/> false	<input checked="" type="checkbox"/> true

© Copyright IBM Corporation 2016

6

Netcool/OMNIbus Administrator utility

The screen capture that is shown here illustrates the interface within the Netcool/OMNIbus Administrator utility that is used to create users and groups.

Web GUI user administration

Web GUI is accessed through Dashboard Application Services Hub

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

- Roles determine access to Dashboard Application Services Hub features

Dashboard Application Services Hub is based on WebSphere Application Server

WebSphere Virtual Member Manager (VMM) provides user authentication within Dashboard Application Services Hub

Three options for VMM repositories:

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

The default Web GUI installation configures file-based and ObjectServer

You cannot use ObjectServer and LDAP together

© Copyright IBM Corporation 2016

7

Web GUI user administration

The Web GUI component is a collection of features that are hosted within Dashboard Application Services Hub. Access to the features is controlled with Dashboard Application Services Hub roles. Dashboard Application Services Hub users are defined in a WebSphere Virtual Member Manager (VMM) repository. You have three options for the type of VMM repository. One option is the file-based repository. The default administrator user is defined in a file-based repository. A second option for VMM repository is an ObjectServer. The last option is an LDAP server. You can configure a file-based repository and one of the other options, but not both. The default configuration is a file-based repository and an ObjectServer repository.

ObjectServer user synchronization

ncoadmin and **ncouser** created in Dashboard Application Services Hub, and added to ObjectServer

root and **nobody** copied from ObjectServer, and to added Dashboard Application Services Hub

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 2016

8

ObjectServer user synchronization

When the ObjectServer is configured as a VMM repository, user information is stored in the ObjectServer. The WebSphere VMM plug-in retrieves user information from the ObjectServer. When a user is added to the ObjectServer, the user appears in the VMM repository. The user can access Dashboard Application Services Hub. The same holds true for ObjectServer groups. The WebSphere VMM plug-in reads the group information from the ObjectServer. If you add a group to the ObjectServer, it is immediately available within WebSphere. If you create a user within WebSphere, the user information is written 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

Name/	Description	ID
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
Operators	Level 1 operators	10
Probe	Permissions required for a probe user	4
Public	Public Group	0
System	System Group	1

© Copyright IBM Corporation 2016

9

ObjectServer group synchronization

The same synchronization technique holds for groups. ObjectServer group information appears within WebSphere. WebSphere group information is written to the ObjectServer.

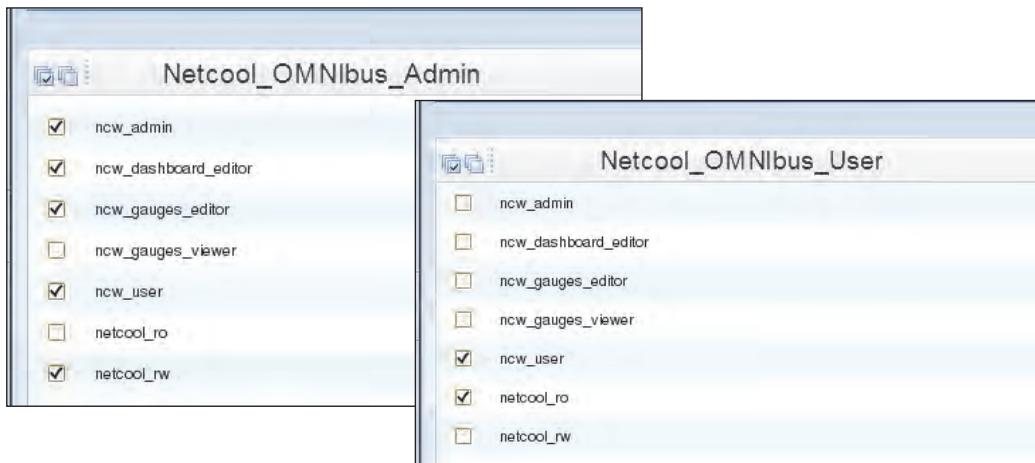
Web GUI roles



Web GUI roles

Dashboard Application Services Hub roles control access to application features, including the Web GUI component. When you install Web GUI, the installation creates a number of Dashboard Application Services Hub roles. Some of the roles control access to Dashboard Application Services Hub pages. Other roles control access to the tools that a user can access through the event list.

Web GUI group roles



The roles that are shown here are the default values

© Copyright IBM Corporation 2016

11

Web GUI group roles

When you install Web GUI, an installation utility creates two sample users and two sample groups. One sample group is defined with the roles that are required for administrative access. The other sample group is defined with the roles that are required for nonadministrative access. You can use the sample groups or create your own.

LDAP as an authentication source

LDAP is an option for user authentication

- Can be used for user authentication with Dashboard Application Services Hub
- Can be used for user authentication with 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 members of a specific LDAP subtree

LDAP as an authentication source

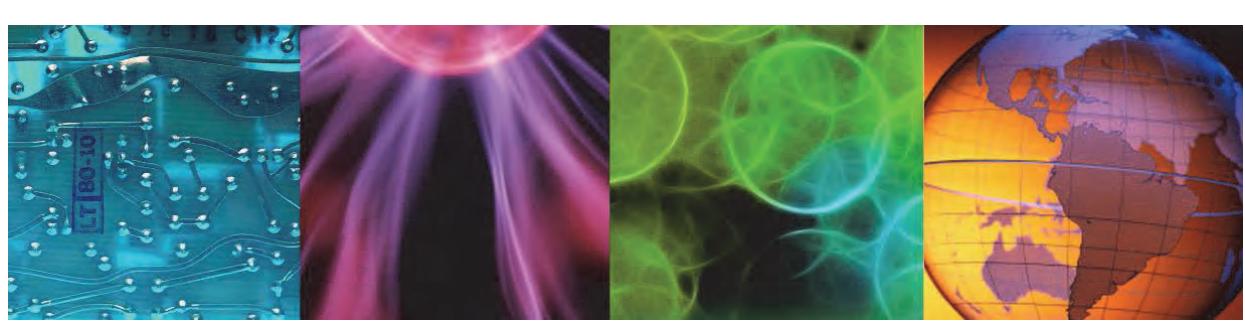
You can configure LDAP as a WebSphere VMM repository. The LDAP configuration must occur after you install Web GUI. You must decide whether you want to use LDAP or the ObjectServer as the VMM user repository. You cannot use both. The choice between the options is generally a function of the number of users and the type of access.

If the number of users is relatively small, the best option might be an ObjectServer user repository. If the number of users is large, and the users are defined in LDAP, then an LDAP user repository is a better choice.

When LDAP is configured as the VMM user repository, you generally configure WebSphere to access a specific subtree within LDAP. A large company might have many thousands of users that are defined within LDAP, but not all users require access the Dashboard Application Services Hub.



Lesson 2 ObjectServer user administration



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Create restriction filters
- Creates roles
- Create groups
- Create users

ObjectServer users

Three categories of users:

- System
 - Super user with authority to modify all ObjectServer objects
- Administrator
 - Authority to change some ObjectServer objects but not all
- Normal
 - Cannot modify the ObjectServer

Group membership defines user authority

- System user belongs to the System group
- Administrator user belongs to the Administrator group
- Normal user belongs to the Normal group

ObjectServer users

Three categories of ObjectServer users exist. A System user is the superuser. The System user can change any type of object in the ObjectServer. The second category is the Administrator. The Administrator user can change some ObjectServer objects. In general, the objects are related to user information. The last category of user is the Normal user. The Normal user can access event records but cannot change any ObjectServer objects.

The group definition defines the category of user.

User configuration requirements

Restriction filters

- An option that can be used to restrict the events that the user can access
- Applied at the user or group level

Roles

- Define database and table access permissions
Alter, delete, drop, insert, select, and update

Groups

- Used to associate common attributes to one or more users
Roles, restriction filters
- Can use to restrict desktop tool access

Users

- Username: Textual user ID
- Full Name: Long name for user
- Restriction filter: If used
- Assigned groups
- Password: Internal or external

© Copyright IBM Corporation 2016

15

User configuration requirements

There are four aspects to ObjectServer users. A restriction filter is used to limit access to event records. For example, you have event information that is related to databases, servers, and networks. You have database administrators that need access to database issues. You can define a restriction filter that limits access to only database events for the database administrators.

The ObjectServer role controls access to database actions. The type of database action determines whether a user can change a database table.

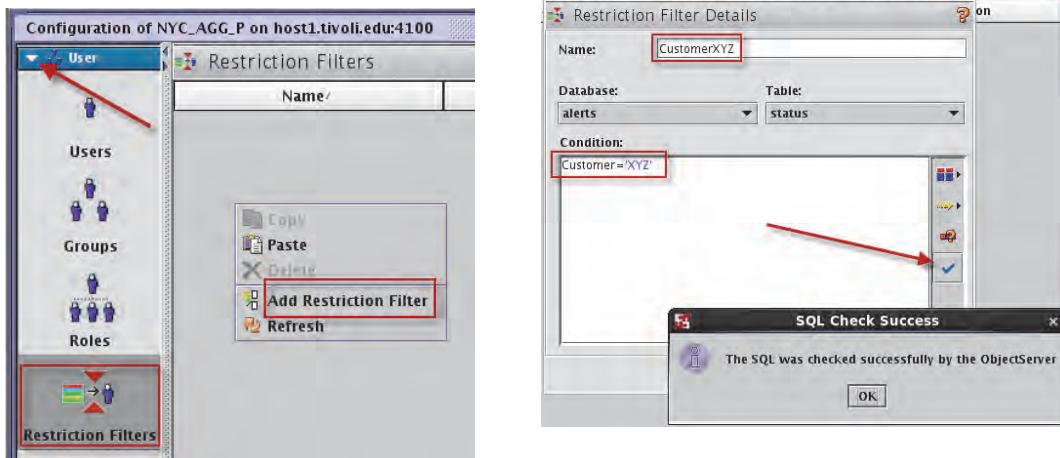
The ObjectServer group is used to organize users into operational entities. You assign roles to groups.

The ObjectServer user definition contains a user name and a Full Name. The user name is how the user accesses the ObjectServer. The Full Name is optional. Users are assigned to groups. The user record defines where the password is stored: in the ObjectServer or outside the ObjectServer.

Restriction filters

Restriction filters create a filter (SQL where clause) that is applied to groups and users

- Filter on any database or table
- Users inherit filters that are assigned to a group



© Copyright IBM Corporation 2016

16

Restriction filters

The ObjectServer does not contain any restriction filters by default. You create the restriction filter, and enter a name. The name cannot contain spaces or special characters, except the underscore. The filter definition is any valid SQL *where* clause. The filter can consist of a single condition or a combination of conditions.



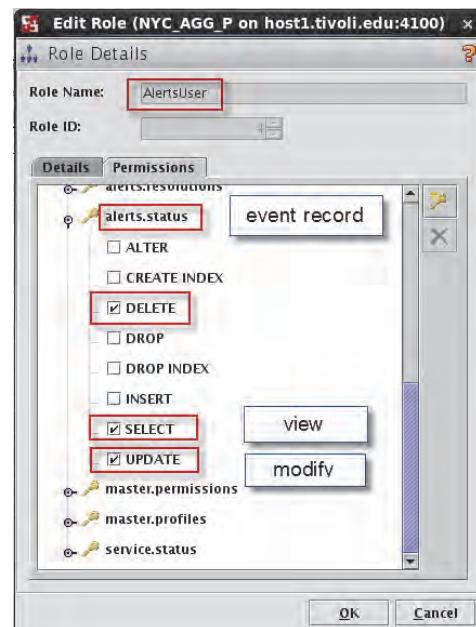
Hint: Click the *green check* icon to validate the syntax of the filter.

Roles

Roles allow actions within groups or users
Roles authorize access to database tables

Example: AlertsUser

Allowed to DELETE, SELECT, and UPDATE
records in alerts.status table



© Copyright IBM Corporation 2016

17

Roles

The ObjectServer includes a collection of roles. You can create more roles. The role definition consists of a database name, table name, and one or more database actions. Some database actions control whether a user can change the *structure* of a database table. For example, the ALTER action is required to add a column to a database table. Other database actions control whether a user can change the *contents* of a database table. For example, the UPDATE action is required to change the contents of a record in a database table.

Groups

Groups manage users efficiently

There are default

Netcool/OMNIbus groups

Groups have associated roles

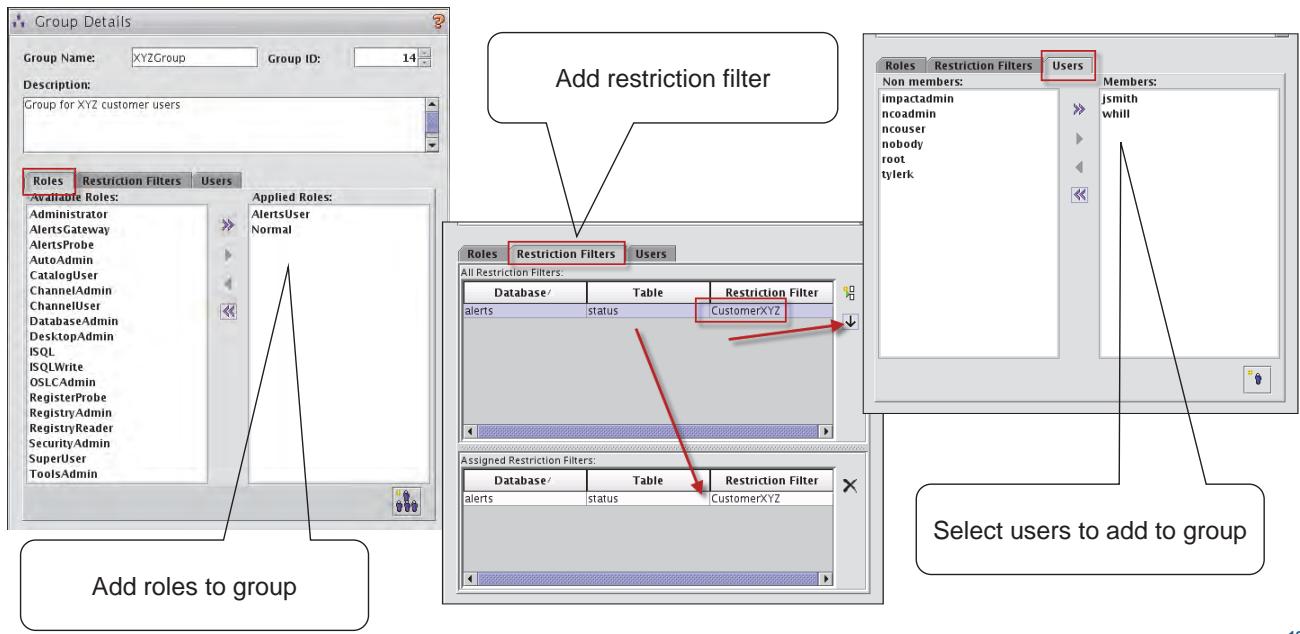
Groups have users as members

Configuration of NYC_<AGG>_P on host1.tivoli.edu:4100			
		Name / Description	ID
	Administrator	Admin Group	2
	Gateway	Permissions required for a gateway user	5
	iSQL	Read only iSQL access	7
	iSQLWrite	Write iSQL access	6
	Netcool_Admin	VMM synchronised groups	10
	Netcool_User	VMM synchronised groups	9
	Normal	Normal Group	3
	Operations	VMM synchronised groups	11
	Probe	Permissions required for a probe user	4
	Public	Public Group	0
	System	System Group	1
	vmmusers	Group to manage VMM users	8

Groups

You use groups to organize the users and assign access authority. The System group contains the roles that allow update access to all ObjectServer tables.

Creating a group



© Copyright IBM Corporation 2016

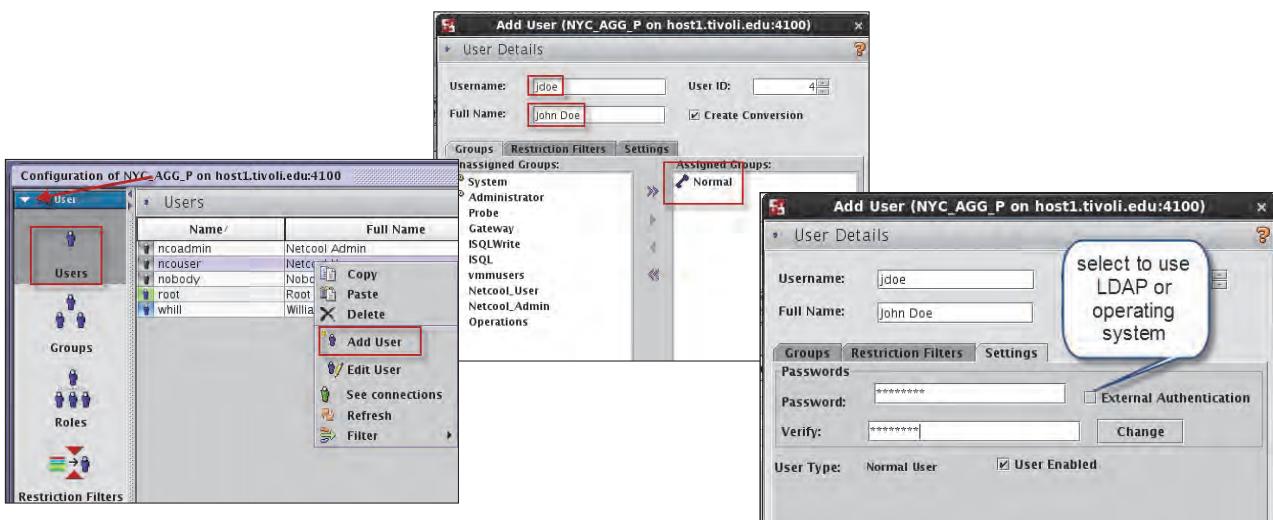
19

Creating a group

Create a group, and assign a name, which cannot contain spaces or special character except for the underscore. Select one or more roles. You can optionally assign a restriction filter. Add users to the group.

You can create the group without any users and add users later.

Adding new users



© Copyright IBM Corporation 2016

20

Adding new users

You create a user, and enter a user name. The user name is the user ID that is used to access the desktop. The user name cannot contain spaces or special characters except for the underscore. Assign one or more groups to the user. Optionally, assign a restriction filter. Click Settings, and configure the password. The password is saved in the ObjectServer by default. Select External Authentication to use an external source.



Note: You cannot select the type of external source: operating system or LDAP. The default configuration is operating system.

Configuring the ObjectServer to use LDAP for external authentication

User IDs are stored within the ObjectServer, and LDAP is used for password authentication

Configure the ObjectServer to use LDAP authentication by setting the **Sec.ExternalAuthentication** ObjectServer property to **LDAP**

LDAP access information is configured in a property file:

\$OMNIHOME/etc/ldap.props

- Requirements:

```
Hostname: 'omnihost.tivoli.edu'  
Port: 389  
LDAPSearchBase: "ou=tipusers,cn=tipRealm,dc=ibm,dc=com"  
LDAPSearchFilter: '(&(uid=%s))'
```

Restrictions:

- Tivoli Netcool/OMNibus is not intended to be used to manage user accounts in LDAP, and so does not provide the capability to change passwords in an LDAP server

© Copyright IBM Corporation 2016

21

Configuring the ObjectServer to use LDAP for external authentication

To configure the ObjectServer to use LDAP, you must modify an ObjectServer property. The property is:

Sec.ExternalAuthentication: LDAP

You configure the LDAP access information in a property file. The property file contains the host name, and port number for the LDAP server. The file also contains the filter criteria to limit the LDAP search to a specific subtree.

The ObjectServer cannot update LDAP so it cannot change a password in LDAP.

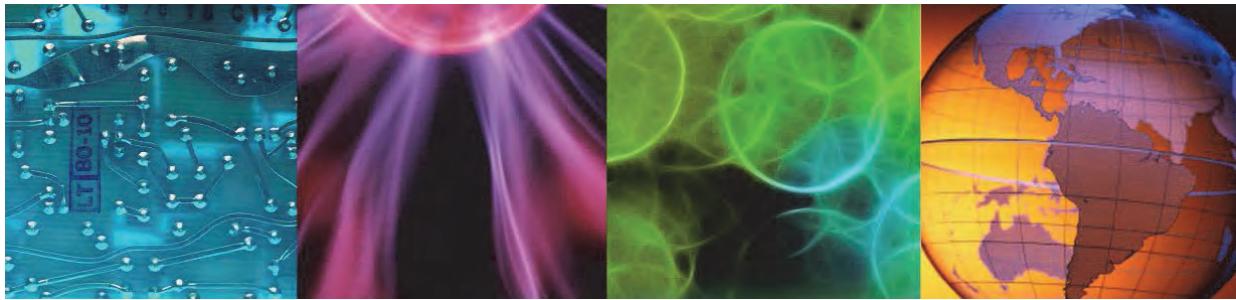
You can define a single LDAP server to the ObjectServer. If the LDAP server is down, the ObjectServer cannot authenticate users with passwords in LDAP.



Lesson 3 Web GUI user administration



Lesson 3 Web GUI user administration



© Copyright IBM Corporation 2016

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 perform the following tasks:

- Create users
- Create groups
- Configure user preferences

Web GUI user administration

Web GUI users are defined in WebSphere Virtual Member Manager repositories

User administration tasks:

- Create groups

- Create users

- Assign roles

These tasks require Dashboard Application Services Hub administrative authority

- User with *iscadmins* role

Web GUI users must also exist in the ObjectServer

How users are created, and managed is a function of how WebSphere Virtual Member Manager is configured:

- ObjectServer as the default user repository

- LDAP as the default user repository

© Copyright IBM Corporation 2016

23

Web GUI user administration

Web GUI users are Dashboard Application Services Hub users with specific roles that are assigned. The user information is defined in a WebSphere Virtual Member Manager repository. You create users and groups with the WebSphere Administrative Console. You assign roles to users and groups with Dashboard Application Services Hub.

A user with Web GUI access rights user must exist in the ObjectServer. User administration varies whether WebSphere is configured with an ObjectServer user repository or LDAP user repository.

ObjectServer user repository

Default configuration when installing Web GUI

Create a user in WebSphere, and user details are written to the ObjectServer

User does not physically exist in WebSphere

WebSphere reads user details from the ObjectServer

Create a user in the ObjectServer, and user details appear in WebSphere

User does not physically exist in WebSphere

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

Selected	User ID	First name	Last name	E-mail	Unique Name
<input checked="" type="checkbox"/>	ncoadmin	ncoadmin	tivoli		uid=ncoadmin,o=netcoolObjectServerRepository
<input type="checkbox"/>	ncouser	ncouser	tivoli		uid=ncouser,o=netcoolObjectServerRepository
<input type="checkbox"/>	nobody		Nobody		uid=nobody,o=netcoolObjectServerRepository
<input type="checkbox"/>	root		Root	User	uid=root,o=netcoolObjectServerRepository
<input type="checkbox"/>	smadmin	smadmin	smadmin		uid=smadmin,o=defaultWMFileBasedRealm

24

ObjectServer user repository

The ObjectServer user repository is the default configuration. When WebSphere is configured with an ObjectServer user repository, the user information is stored in only the ObjectServer. The WebSphere VMM plug-in reads the user information from the ObjectServer. If you create a user with the administrative console, the user information is written to the ObjectServer.



Important: The WebSphere VMM can be configured to use a high availability pair of ObjectServers. With the high availability configuration, if the primary ObjectServer is down, WebSphere can still access user information from the backup ObjectServer.

LDAP user repository

Cannot be configured when installing Web GUI

Must be configured postinstallation

Create a user in WebSphere, and user details are written to LDAP

User does not physically exist in WebSphere

WebSphere reads user details from LDAP

Create a user in LDAP, and user details appear in WebSphere

User does not physically exist in WebSphere

Option to create ObjectServer users and groups

© Copyright IBM Corporation 2016

25

LDAP user repository

The second option for VMM user repository is an LDAP server. When WebSphere is configured with an LDAP user repository, the user information is stored in only the LDAP server. The WebSphere VMM plug-in reads the user information from the LDAP server. If you create a user with the administrative console, the user information is written to the LDAP server.

Because WebSphere can update LDAP, users can change their passwords from Dashboard Application Services Hub, and the change is written to LDAP.

You can configure an option to synchronize changes with an ObjectServer. This option is a Web GUI feature, not a WebSphere feature.



Important: The WebSphere VMM can be configured to use a high availability pair of LDAP servers. With the high availability configuration, if the primary LDAP server is down, WebSphere can still access user information from the backup LDAP server.

Synchronizing LDAP users with the ObjectServer

Enable the user synchronization function:

- This function creates the LDAP users and groups 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

Modify two files 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 2016

26

Synchronizing LDAP users with the ObjectServer

To enable the synchronization feature, you must configure two files. You modify the Web GUI initialization file to enable synchronization.

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

Synchronization occurs based on a frequency. The default frequency is every hour. You can optionally change the frequency in the data sources definition file.

```
/opr/IBM/netcool/omnibus_webgui/etc/datasources/ncwDataSourceDefinition.xls  
<config maxAge="3600" />
```

Synchronizing LDAP users with the ObjectServer, continued

Not all WebSphere users are created in the ObjectServer

Only users with Web GUI roles: *nco_admin*, *nco_user*

ObjectServer users are not enabled by default

The user works with Web GUI even when disabled in the ObjectServer

The user cannot be used with the Native client unless it is enabled

ObjectServer roles are not assigned by default

The user does not need ObjectServer roles when used with Web GUI

The user cannot be used with the Native client unless ObjectServer roles are assigned

ObjectServer user password is not defined

When used with Web GUI, the user password is verified against LDAP

The user cannot be used with the Native client unless the ObjectServer is configured to use LDAP

© Copyright IBM Corporation 2016

27

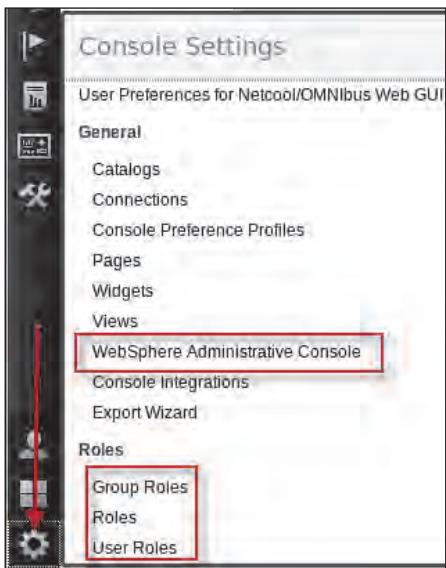
Synchronizing LDAP users with the ObjectServer, continued

The synchronization feature has some conditions, and limitations. First, synchronization creates ObjectServer users for the WebSphere users with Web GUI roles: *ncw_admin* or *ncw_user*. The WebSphere VMM plug-in might retrieve hundreds of users from LDAP. You might not need all the users in the ObjectServer. After you add the Web GUI roles to the users in WebSphere, the next synchronization creates the corresponding entries in the ObjectServer.

The ObjectServer users that created by synchronization are not enabled in the ObjectServer. You do not enable the user in the ObjectServer for Web GUI access. However, if that user needs to access the ObjectServer with the Native Desktop, then you must manually enable the user ID in the ObjectServer.

When synchronization creates a user in the ObjectServer, it does not configure password authentication. With this type of scenario, the user is defined in LDAP, and the password is stored in LDAP. Synchronization creates the user record in the ObjectServer, but it cannot configure password authentication. If you want an LDAP user to have access to Web GUI and the Native desktop, you must manually update the ObjectServer user definition to specify external authentication. The ObjectServer must be configured to use the same LDAP server for password authentication.

Dashboard Application Services Hub administrative authority



Granted through *iscadmins* role

© Copyright IBM Corporation 2016

28

Dashboard Application Services Hub administrative authority

Web GUI user administration occurs within two applications. You use WebSphere Administrative console for some parts and Dashboard Application Services Hub for other parts.

Creating groups and users

WebSphere Administrative Console

Must use **smadmin** user ID

© Copyright IBM Corporation 2016

29

Creating groups and users

Dashboard Application Services Hub contains a link to the WebSphere Administrative Console. WebSphere administrative console is a separate application. The administrative console does not use the WebSphere VMM repositories. You must use the **smadmin** user to access the administrative console.

You use the administrative console application to manager users and groups.

Creating a group

Creating a group:

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

© Copyright IBM Corporation 2016

30

Creating a group

You create a group, and enter a name. The group name cannot contain spaces or special characters except for the underscore.

Creating a user

Creating a 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 2016

31

Creating a user

You create a user, and enter a user name. The user name cannot contain spaces or special characters except for the underscore. You enter a first name, a last name, and a password.

You assign one or more groups to the user.

When an ObjectServer is configured as the VMM user repository, this information is written to the ObjectServer, including the password. When LDAP is used as the VMM user repository, this information is written to LDAP, including the password.

Assigning roles to a group

Defining group with read/write access:

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

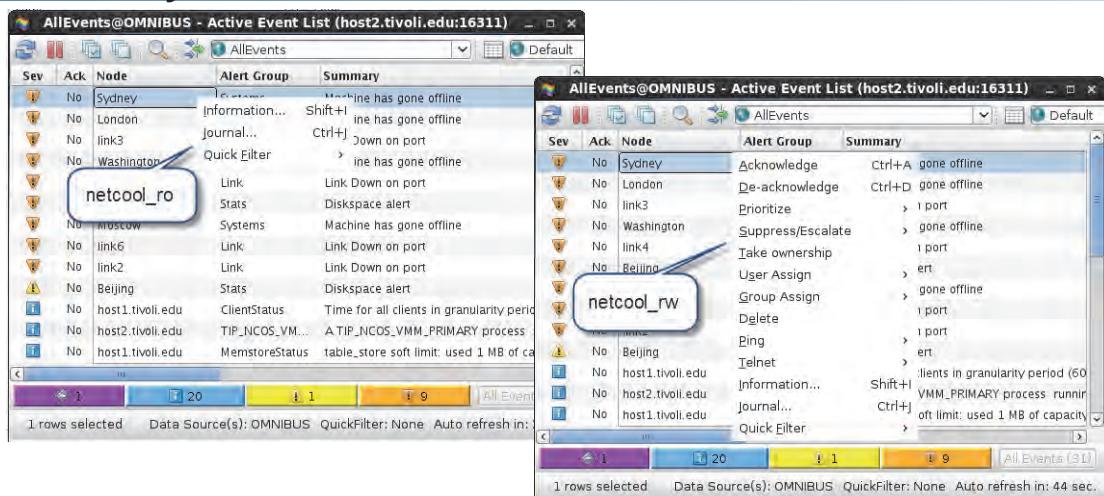
© Copyright IBM Corporation 2016

32

Assigning roles to a group

You create a group with administrative console, and assign users to the group. Then, you use Dashboard Application Services Hub to assign one or more roles to the group.

Read-only versus read/write access



Dashboard Application Services Hub roles *netcool_ro*, and *netcool_rw* control access to tools

If the user has access to the *Delete tool*, the user can delete events from the ObjectServer

Web GUI users do not need ObjectServer roles assigned

© Copyright IBM Corporation 2016

33

Read-only versus read/write access

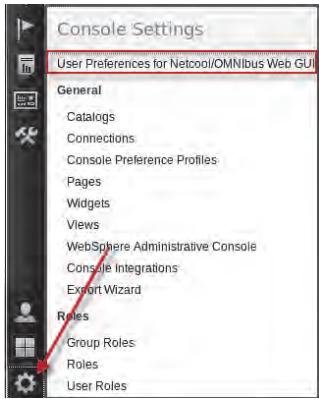
Dashboard Application Services Hub roles determine whether a user can modify an event record. The *netcool_ro* role restricts access to the desktop tools that modify event records. Without access to the tools, a user cannot modify an event record.

User preferences

Limit access to events

Define home page

Control access to Event List features



User Preferences for Netcool/OMNibus Web ...

User name: whill

User filter: Delete_Flag=0

User's home-page: /

Event List Configuration

Allow filter and view selection
 Allow filter builder access
 Allow view builder access
 Allow preference configuration
 Allow refresh rate configuration

Refresh rate (seconds):

Minimum refresh rate (seconds):

Allow event selection
 Show basic event information
 Show event details
 Show journals
 Edit journals (read write role)

© Copyright IBM Corporation 2016

34

User preferences

You configure preferences on a user by user basis. Preferences control access to various event list features.

User preferences explained 1 of 3

User filter

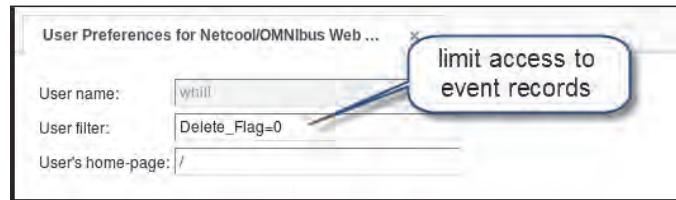
- SQL filter condition
- Restricts access to event records that meet the condition

User's home-page

- Define default home page for user
- Example:
`/ibm/console/webtop/mypage.html`

Page must be saved to:

`/opt/IBM/JazzSM/profile/installApps/JazzSMNode01Cell/isc.ear/OMNIbusWebGUI.war`



© Copyright IBM Corporation 2016

35

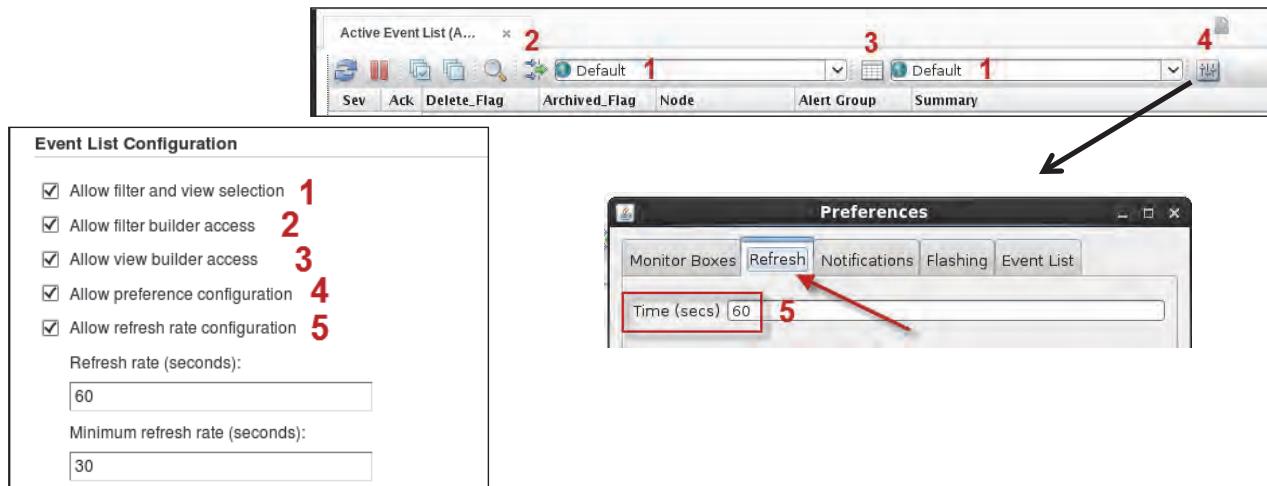
User preferences explained 1 of 3

The user filter provides the same function as the ObjectServer restriction filter. It is an SQL where clause that limits access to event records.

You can configure a default home page for the user. You can enter the home page as part of a URL, or a complete path. If the information is entered as part of a URL, the HTML page must be saved in the following directory:

`/opt/IBM/JazzSM/profile/installApps/JazzSMNode01Cell/isc.ear/OMNIbusWebGUI.war`

User preferences explained 2 of 3



© Copyright IBM Corporation 2016

36

User preferences explained 2 of 3

The next set of preferences, as shown here, control access to features on the event list. You can configure the user so that they can perform the following tasks:

- Change the filter in the event list
- Change the view in the event list
- Use the filter builder to change the configuration of a filter
- Use the view builder to change the configuration of a view
- Access the preferences option to change the event list preferences
- Change the refresh rate of the event list

User preferences explained 3 of 3

The screenshot illustrates the relationship between user preferences and event management. On the left, a configuration dialog shows settings for 'Minimum refresh rate (seconds)' (30) and five checked options: 'Allow event selection' (1), 'Show basic event information' (2), 'Show event details' (3), 'Show journals' (4), and 'Edit journals (read write role)' (5). To the right, three windows demonstrate the results of these preferences:

- Active Event List:** A grid showing event records. The first record has 'Delete_Flag' set to 'No' (highlighted with a red box labeled 1). A callout bubble labeled 'selected event' points to this row.
- Alert Status for Serial Number 26599 - Fields View:** A table showing event details. The 'Node' field is set to 'TestNode' (highlighted with a red box labeled 2). The 'Severity' field is set to 'No' (highlighted with a red box labeled 3). The 'LastOccurrence' field shows the date '12/17/14 4:03:24 PM' (highlighted with a red box labeled 4).
- Alert Status for Serial Number 26599 - Journal View:** A journal entry window. It contains a message from 'whill' at 'Tue Dec 30 14:30:15 GMT 2014' stating 'This is my entry' (highlighted with a red box labeled 5). An arrow points from the 'Add to journal...' button to the journal entry area.

© Copyright IBM Corporation 2016

37

User preferences explained 3 of 3

The next set of preference settings control what information that is related to the event that the user can access, including these examples:

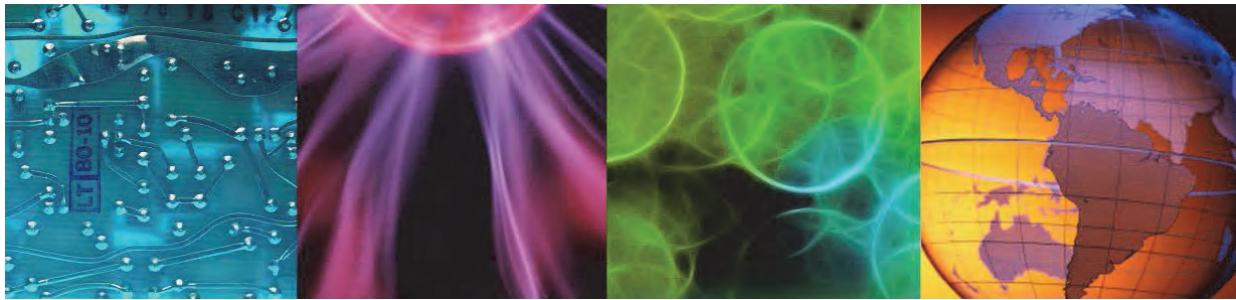
- Whether the user can select an event record in the desktop
- Whether the user can open the event information view
- Whether the user can view event details
- Whether the user can view event journals
- Whether the user can add an event journal record



Lesson 4 Creating Web GUI startup pages



Lesson 4 Creating Web GUI startup pages



© Copyright IBM Corporation 2016

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 default startup pages for users.

Overview

Startup pages open when user logs in

Assigned to users or user groups based on their role

Can hide links to other pages from the navigation

Steps to create a startup page:

1. Create a role
2. Create a startup page, and assign access for the role
3. Create a view, and assign the startup page to the view
4. Create a console preference profile, and assign the view to the profile
5. Assign the role to the profile
6. Assign the role to the user or group

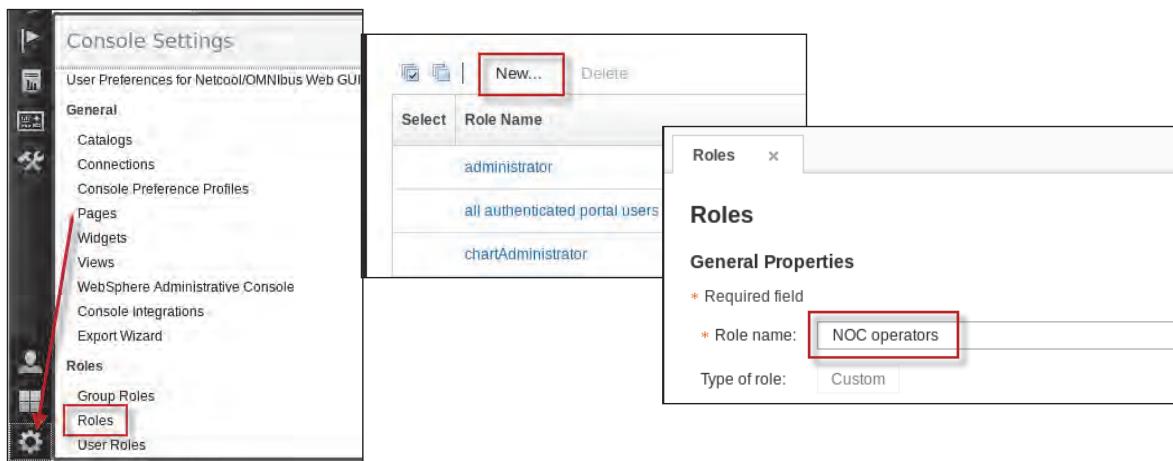
Overview

A startup page opens automatically when a user logs in to Dashboard Application Services Hub. Startup pages are assigned to users, or groups based on a role. You can configure the feature to hide access to other pages, which effectively constrains the users to just the defined startup pages.

Creating a role

Users are granted access to resources based on assigned roles

All roles that are created in the portal have a resource type of *Custom*



© Copyright IBM Corporation 2016

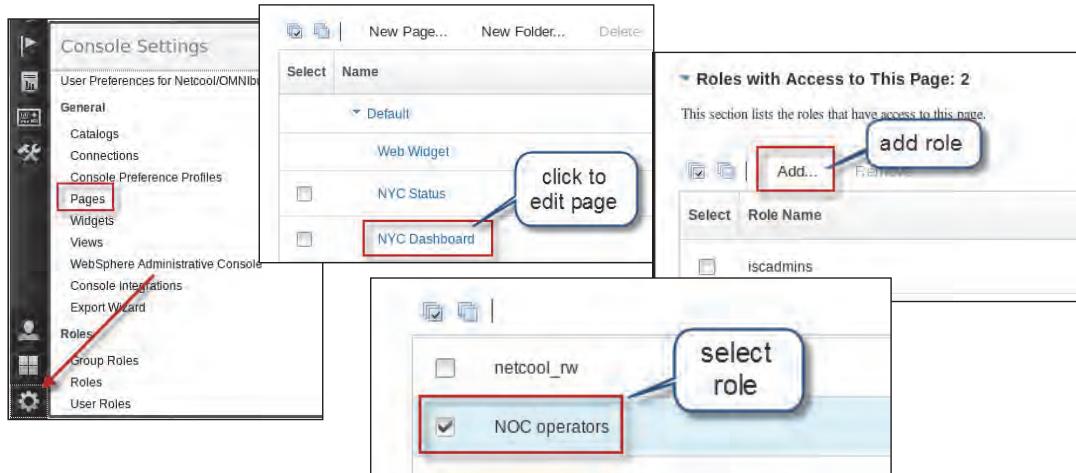
40

Creating a role

The ability to configure a default startup page is based on a role. You can use an existing role or create a role for just the startup feature.

You create a role, and assign a name. The name can contain spaces.

Assigning role to page



Repeat these steps to add role to all startup pages

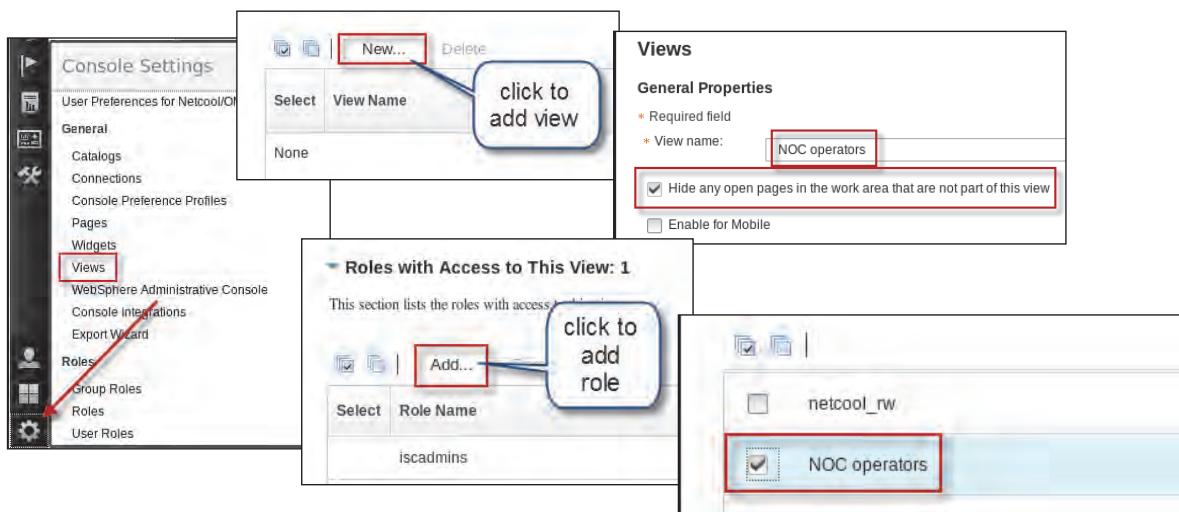
© Copyright IBM Corporation 2016

41

Assigning role to page

Next, you assign the role to one or more pages. The pages might exist, or you might create the pages. The screen captures that are shown on this slide illustrate how to add a role to an existing page.

Creating a view 1 of 2



© Copyright IBM Corporation 2016

42

Creating a view 1 of 2

Next, you create a view. The view name can contain spaces. The view name does not have to be the same as the role name, as shown in the example. Assign the role to the view.

Creating a view 2 of 2

The first screenshot shows a 'Pages in This View' section with a 'Set all pages in this view' link and an 'Add...' button highlighted with a red box and a callout 'click to add pages'.

The second screenshot shows a list of pages: 'NYC Status' and 'NYC Dashboard'. Both have checkboxes checked. A callout says 'select one or more pages'.

The third screenshot shows a configuration table for the selected pages. The 'Launch' column has checkboxes for both pages, which are also highlighted with red boxes. A callout says 'select pages to open automatically'.

© Copyright IBM Corporation 2016

43

Creating a view 2 of 2

You add pages to the view. You can add one or more pages. After you add the pages, you configure the pages to open automatically. The column that is labeled Default is used to select which page is in focus. The example contains two pages. Both pages are configured to open automatically. The first page is the page that the user sees when the user logs in. Users can switch between the two open pages after they log in.

Creating a console preference profile 1 of 2

Console Settings

User Preferences for Netcool/OMNIbus Web

General

Catalogs

Connections

Console Preference Profiles

Pages

Widgets

Views

WebSphere Administrative Console

Console Integrations

Export Wizard

Roles

Group Roles

Roles

User Roles

Console Preference Profiles

General Properties

* Required field

* Preference profile name: **NOC operators**

Preference profile unique name: 1117794812

Theme: **IBM Design Signature** Preview select for dark background

Console Bidirection Options

Component direction: Default

Console view options

Show view selector **Hide view selector** disable option to change views

Select the view options the user has access to in the View drop down menu in the banner.

Required view (at least one required view must be selected)

All tasks System and custom views Core views show only custom views

You can allow the user to change views, or limit the user to the custom view

Creating a console preference profile 1 of 2

The next part of the console preference configuration contains an option to select a page *theme*. The theme controls the overall appearance of the page. The default option is a page with a white background. The selectable option displays the page with a dark background.

The other options in this section control access to other pages. You can configure the feature so that the user can select from one or more views. You can also configure the option to hide the other views, and the user sees only the startup pages.

Creating a console preference profile 2 of 2



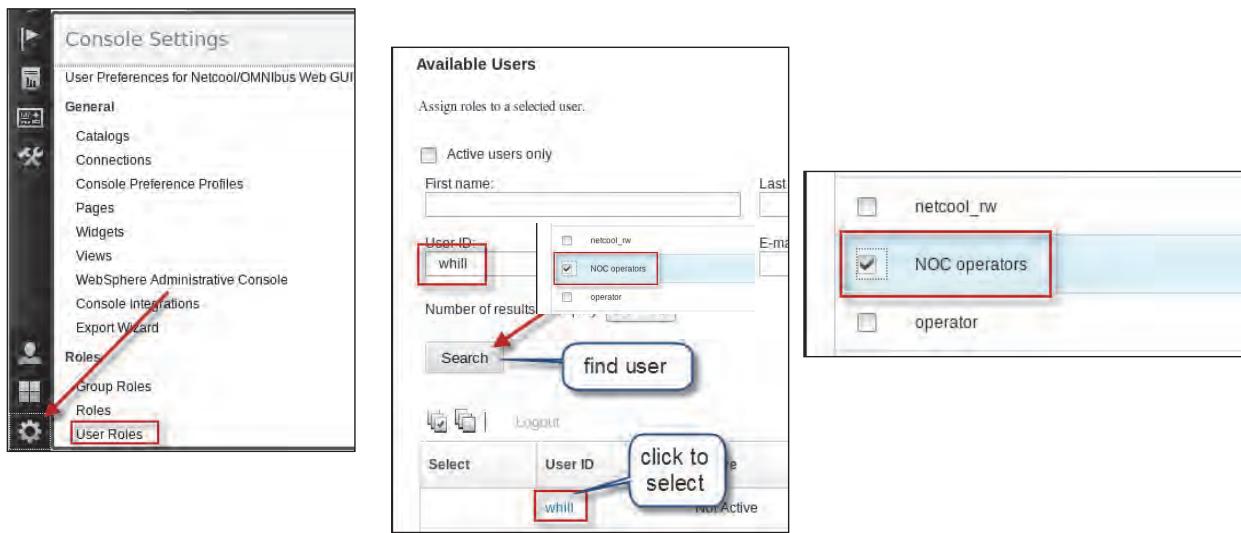
© Copyright IBM Corporation 2016

45

Creating a console preference profile 2 of 2

You add the role as the last step in the configuration.

Assigning role to user



© Copyright IBM Corporation 2016

46

Assigning role to user

The role is assigned to one or more pages. The role is assigned to a view. The view contains the pages. The role is assigned to a console preference. The last step is to assign the same role to a user or a group. It is the role that links the user to the startup pages.

Result



User navigation is restricted

- Access to pages in view

User cannot select another view or other pages

© Copyright IBM Corporation 2016

47

Result

When the user logs in to Dashboard Application Services Hub, the startup pages open automatically. The page that is selected in the console preference is the page that the user sees.

In this example, the console preference is configured to prevent access to other views. This user can see only the pages that are defined in this view.



Student exercises

© Copyright IBM Corporation 2016

48

Student exercises

Summary

You now should be able to perform the following tasks:

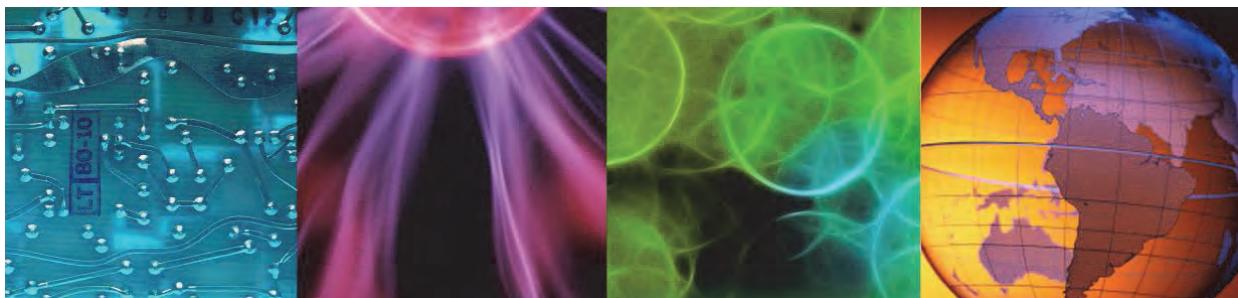
- Describe ObjectServer user administration
- Create ObjectServer users, groups, roles, and restriction filters
- Configure ObjectServer to use LDAP for password authentication
- Describe Web GUI user administration
- Create Web GUI users, roles, and groups
- Configure Web GUI user preferences
- Create default startup pages



7 Customizing Tivoli Common Reporting reports



Customizing Tivoli Common Reporting reports



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

In this unit, you learn how to modify the Cognos® data model to accommodate extra columns in the Netcool/OMNibus event record.

References: *IBM Cognos Framework Manager Version 10.2.0 User Guide*

Objectives

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

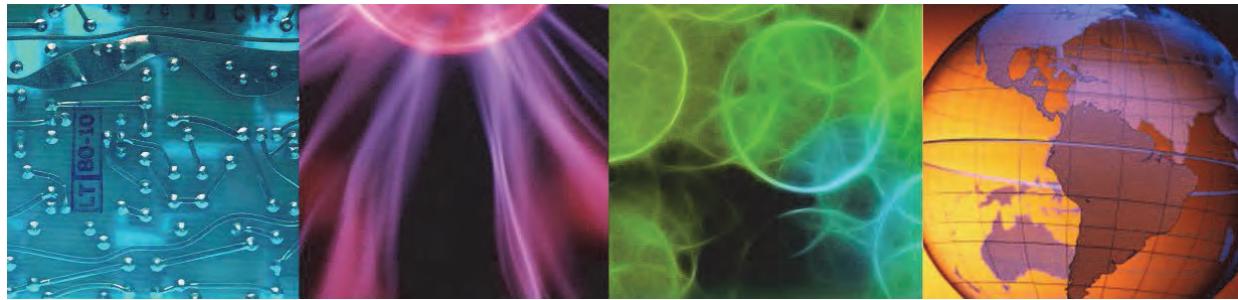
- Install and configure IBM Cognos Business Intelligence Modeling, also known as Framework Manager
- Use Framework Manager to modify the existing Cognos model for the event history database



Lesson 1 Overview



Lesson 1 Overview



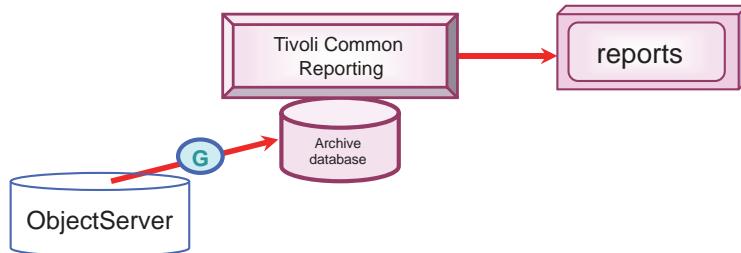
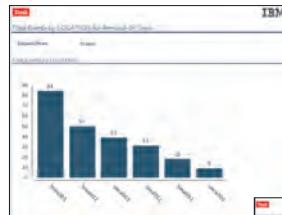
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this lesson, you learn about the components of historical event reporting.

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 2015

4

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: REPORTER

- Customer provided
- DB2, Oracle, Sybase, MS SQL

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

- Included with Netcool/OMNIbus

© Copyright IBM Corporation 2015

5

Components

The database that retains event records is customer-provided. The following databases are supported:

- DB2
- Oracle
- Sybase
- MS SQL
- Sybase

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.

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 that are used to convert integers to text
 - REPORTER_NAMES
 - REPORTER_GROUPS
 - REPORTER_MEMBERS
 - REPORTER_CLASSES
 - REPORTER_CONVERSIONS
- Tables that are used for prompts during report generation
 - REP_SEVERITY_TYPES
 - REP_TIME_PERIODS

© Copyright IBM Corporation 2015

6

REPORTER database configuration

The Netcool/OMNIbus database gateway installation package includes an SQL file. 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 2015

7

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 on column changes and update the respective AUDIT table

AUDIT tables

The database gateway supports two modes of operations: audit mode and reporting mode. 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 ObjectServer behavior with 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

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.

Modifying the event record

Event reporting components are configured based on the standard event record

ObjectServer table alerts.status

When you add columns to the event record, you must also modify

- Archive database
 - REPORTER.REPORTER_STATUS
- JDBC Gateway
 - Map file
- Tivoli Common Reporting
 - Data model
 - Reports

Modifying the event record

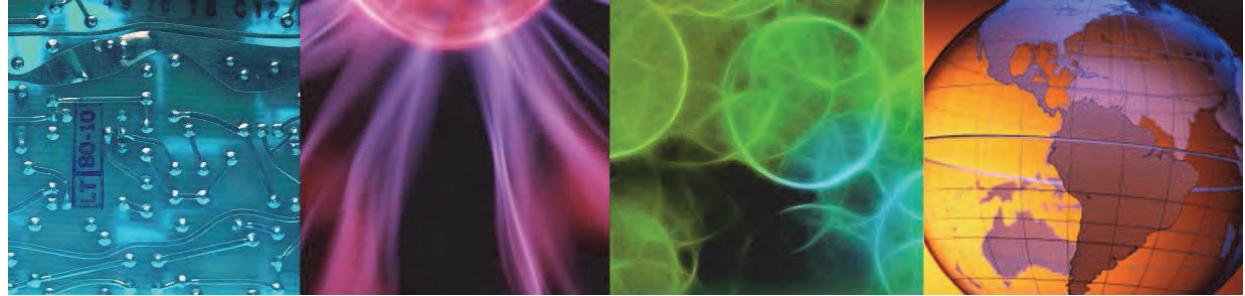
Most users modify the ObjectServer event record and add extra columns. When you add extra columns to the event record, you must modify other components, including these examples:

- Archive database: the supplied SQL file creates a table in the REPORTER database that corresponds to the ObjectServer event record. The SQL creates the same columns in the REPORTER_STATUS table as those columns in the ObjectServer event record. When you add a column to the event record, you must add a corresponding column to the archive database table.
- JDBC Gateway: the gateway is configured to copy selected columns from the event record to corresponding columns in the archive database. The column names are defined in the gateway map file. When you add a column to the event record, you must modify the map file and add a line for the new column.
- Cognos data model: Tivoli Common Reporting is based on IBM Cognos. IBM Cognos uses a component that is called a data model. When you change the archive database, you must also change the data model.



Lesson 2 Framework Manager

Lesson 2 Framework Manager



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 perform the following tasks:

- Describe the function of a data model
- Install Framework Manager
- Use Framework Manager to modify a data model

Definition of a data model

Abstraction of data in a database

- Can be a subset of database
- Logic for joining tables with shared keys

Standardized dimensions to enable reporting across products

- Ensure consistency in structure, naming, and functionality

Relational model: Used for reporting and querying

- Commonly suffixed with (query) after the model name
 - This class

Dimensional model: Used for online analytical processing (OLAP)-style analysis and reporting

- Commonly suffixed with (analysis) after the model name
 - Beyond the scope of this class

© Copyright IBM Corporation 2016

1-12

Definition of a data model

A Cognos data model is a modeled version of the data that is collected by Tivoli products in their databases. Standardized dimensions that are shared across Tivoli are used in the data models to enable cross-product reporting and ensure consistency in structure, naming, and functions of the model. The model metadata is stored in XML format. Two examples of types of models are as follows:

- Relational model: Used for reporting and querying (commonly has a suffix of the phrase “(query)” after the model name)
- Dimensional model: Used for OLAP style analysis and reporting (commonly has a suffix of the phrase “(analysis)” after the model name)

Why model your data?

Drag and drop report creation without writing SQL queries

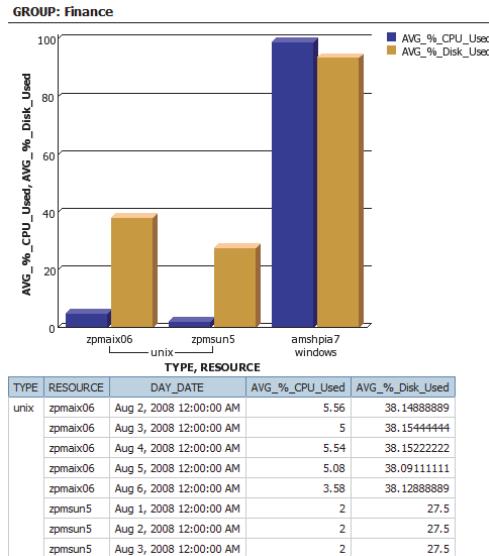
Group resources such as departments, managed system type, and more

Provide unique standard identifiers that enable reporting across products

Allow automatic drill-through creation for hierarchical data

Use dimensional models for advanced analytics and OLAP modeling with Analysis Studio

- Note: Analysis Studio is not included with Tivoli Common Reporting



© Copyright IBM Corporation 2016

1-13

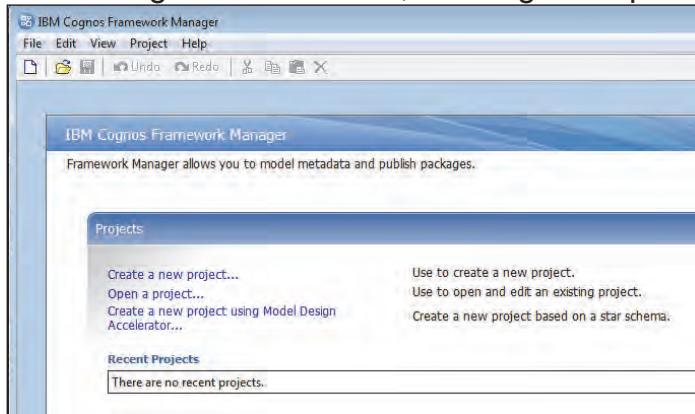
Why model your data?

The data model contains all the logic that is needed to retrieve data from the tables. The report user does not need to know any SQL or HTML to author and run reports.

Framework Manager

Metadata modeling tool that drives query generation

Requires access to Cognos Connection, the Cognos report engine



- Not installed with Tivoli Common Reporting by default
- Runs on Windows operating systems only

© Copyright IBM Corporation 2016

1-14

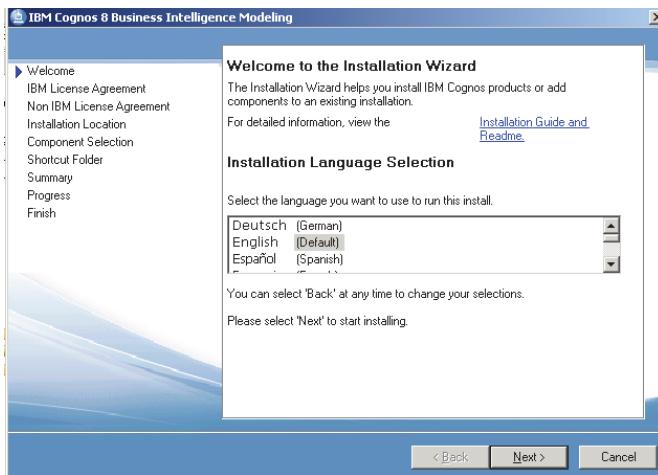
Framework Manager

Framework Manager is a metadata modeling tool that drives query generation for IBM Cognos. A model is a collection of metadata that includes physical information and business information for one or more data sources. IBM Cognos enables performance management on normalized and denormalized relational data sources and also various OLAP data sources. When you add security and multilingual capabilities, one model can serve the reporting, ad hoc querying, and analysis needs of many groups of users around the globe.



Important: Framework Manager is supported on only Microsoft Windows.

Installing Framework Manager



\CognosModeling\win32\issetup.exe

© Copyright IBM Corporation 2016

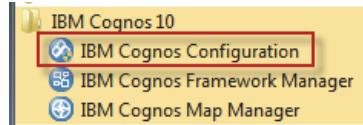
1-15

Installing Framework Manager

Framework Manager is not installed when you install Tivoli Common Reporting. The installation image is available on the Passport Advantage® website. Double-click issetup.exe to start the installation process. Specify the installation language and location, and click Install. The installation runs unattended.

Configuring Framework Manager

1. Start IBM Cognos Configuration



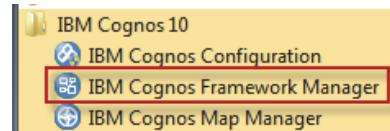
2. Set Gateway URI to match Cognos server

- Gateway URI
- Dispatcher URI

Environment - Group Properties

Name	Value
Data files location	./data
* Map files location	./maps
Temporary files location	./temp
Encrypt temporary files?	False
Sort buffer size in MB	4
* IP Version for Host Name Resolution	Use IPv4 addresses
Gateway Settings	
* Gateway URI	<input type="text" value="https://host2.tivoli.edu:16311/tarf/servlet/dispatch"/>
Other URI Settings	
* Dispatcher URI for external applications	<input type="text" value="http://host2.tivoli.edu:16310/tarf/servlet/dispatch/ext"/>
Font Settings	

3. Start Framework Manager



© Copyright IBM Corporation 2016

1-16

Configuring Framework Manager

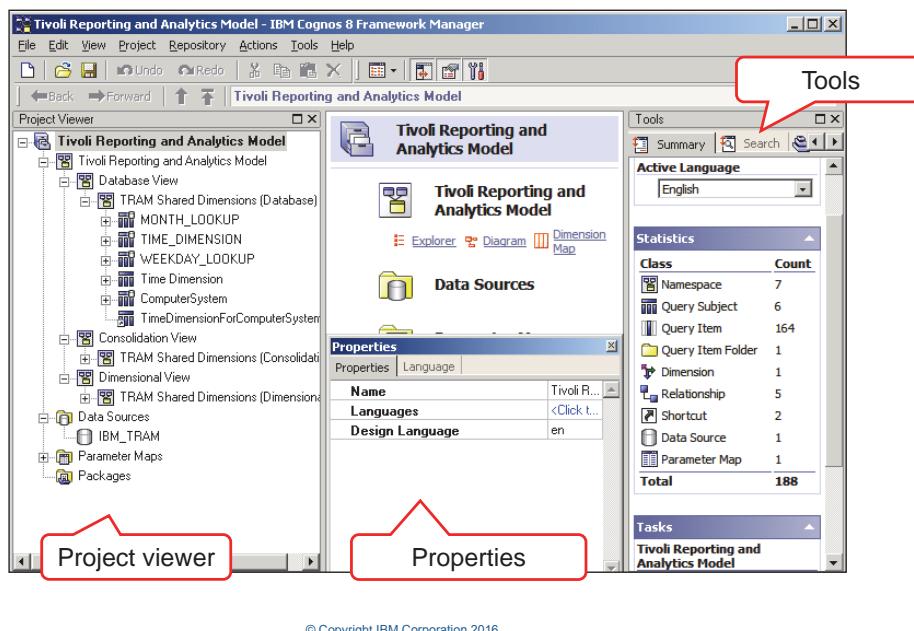
Framework Manager accesses the Cognos engine with a URL. On a single-server installation, the Gateway URI must match the URI in the Tivoli Common Reporting Cognos configuration. You can use the Tivoli Common Reporting configuration utility to validate the Gateway URI.

The configuration utility is included with Tivoli Common Reporting. On the server where Tivoli Common Reporting is installed, run the following command:

```
/opt/IBM/JazzSM/reporting/cognos/tcr_config.sh
```

The user interface for the utility looks the same as what is shown on the slide. Select Local > Environment, and copy the value for the Gateway URI. Paste the value to Framework Manager's configuration.

Framework Manager user interface



© Copyright IBM Corporation 2016

1-17

Framework Manager user interface

The Project Viewer shows the objects in a project in a hierarchical view. You can use the Project Viewer to view, modify, and create objects.

The Properties pane shows the properties of the objects that you last selected in the Project Viewer, Explorer tab, Diagram tab, Dimension Map tab, Dependencies tab, Search pane, or Summary pane. Object properties are set during import, and some property values can be modified during modeling. You can use the Properties pane to add, modify, or delete the properties of objects.

The Tools pane contains the **Summary** tab, the **Search** tab, and the **Dependencies** tab.

- The **Summary** tab shows the language, statistics, and tasks available for the selected object in the Project Viewer.
- Use the **Search** tab to quickly find objects by applying different search criteria, such as the location, the class, a condition, or a property.
- The **Dependencies** tab shows the objects that depend on a selected object.

Framework Manager terminology

Project: A set of files that are used to define the Framework Manager model

Model: Query objects that are defined by the modeler in the project

- Organized into namespaces and folders
- Has a data source for accessing database

Package: All or a subset of query objects that are published to Cognos Connection for use by report authors and business analysts

- Saved as acpf file

Framework Manager terminology

A project contains a model, namespaces, packages, data sources, and related information for maintaining and sharing model information. A single project can span many data sources or tables.

A model is the set of related dimensions, query subjects, and other objects that are required for one or more related reporting applications.

A package is a subset of the dimensions, query subjects, and other objects that are defined in the project. A package is what is published to the IBM Cognos Business Intelligence server, and it is used to create reports, analyses, and ad hoc queries.

Framework Manager terminology, continued

Namespace: Container for organizing model objects

- Unique within a model

Folder: Organize objects within a namespace

Query Subject: Reporting object or result set (equivalent to data sets in BIRT)

- Data source: maps to a corresponding object in the data source and uses a modifiable SQL statement to retrieve the data
- Model: maps to existing metadata in the model
- Stored procedure: Runs a database stored procedure to retrieve or update the data

Query Item: Reporting object that is contained within a query subject

- Usually maps to one or more corresponding objects in the data source (equivalent to data set row in BIRT)

Identifiable by their icons

- *Framework Manager User Guide > Getting Started with Framework Manager > The Project Page > The Project Viewer*

Framework Manager terminology, continued

A namespace must be unique with model. You can have objects of the same name in different namespaces.

Folders can contain namespaces, query subjects, or other folders. Folders must be unique within a namespace, but not within the data model.

A query subject can be a database table or similar object.

A query item is a row of data.

Individual icons identify objects in Cognos Business Intelligence products. For a reference to the icons and their explanations, refer to the following guide:

Framework Manager User Guide > Getting Started with Framework Manager > The Project Page > The Project Viewer

Facts and dimensions

Facts (KPI)

- Contain quantitative or factual data
 - The information that is queried (summarized historical data, for example)
- Usually numerical, additive measurements, and can consist of many columns, and millions or billions of rows
- For example, resource usage, availability metrics, monitored data

Dimensions

- Holds descriptive data that reflects the dimensions, or attributes, of a business
- For example, date, time, resource name

Organized into a star schema

Facts and dimensions

Facts are metrics that are measures or key performance indicators.

- Numeric or time interval
- Non-indexed columns
- Usually additive
- Query subjects with only 0..n or 1..n cardinalities

Fact tables contain foreign keys that relate to the dimensions. They can have different levels of granularity, such as hourly, daily, weekly.

Dimensions are descriptive information, such as Server Name or time. Query subjects with 0..1 or 1..1 are dimensions. In other words, query subjects on the one side of a one-to-many relationship are dimensions.

Conformed dimensions are applicable to multiple fact tables. A table in the database that contains descriptive attributes and corresponding names, meanings, and values that are agreed to across the enterprise are conformed dimensions. Time stamp is a conformed dimension.

Regular dimensions contain descriptive and business key information, and organize the information in a hierarchy, from the highest level of granularity to the lowest, allowing for OLAP-style queries.

Measure dimensions are a collection of measures (typically facts) for OLAP-style queries. Scope relationship exists between measure dimensions, and regular dimensions to define the level at which measures are available for reporting.

Identifiers and attributes

Identifiers

- Primary keys
 - Unique values
 - Date or Date/time
 - Any indexed column

Attributes

- Descriptive strings
 - Managed system name
 - Operating system type
 - String values are automatically categorized as Attributes

Usage determines how Framework Manager aggregates values

© Copyright IBM Corporation 2016

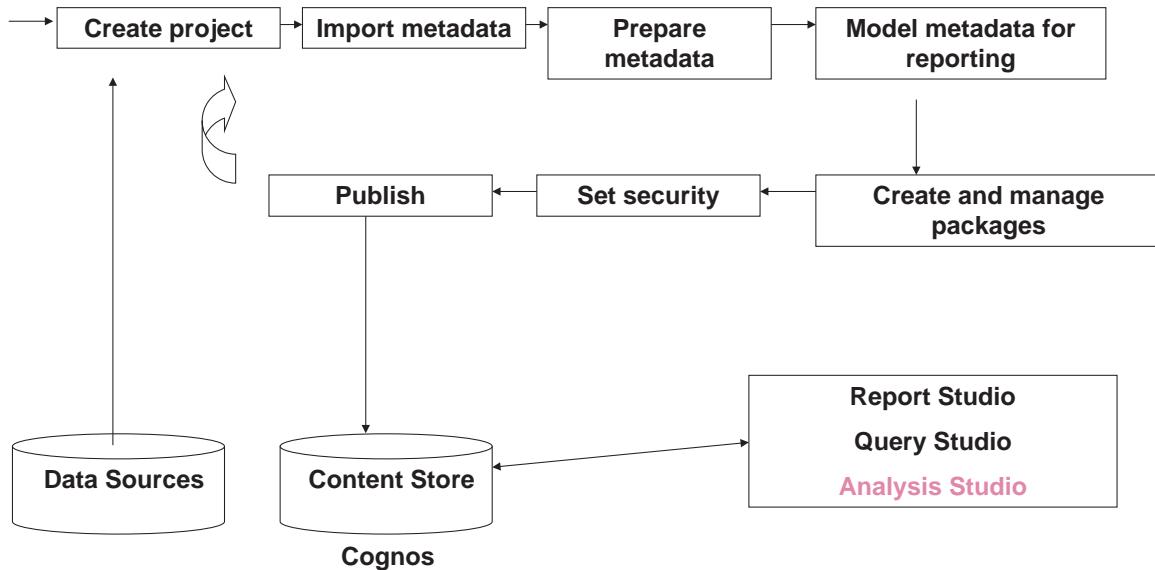
1-21

Identifiers and attributes

Every query subject needs to have one or more unique identifiers that are used to relate to other query subjects.

Ensure that attributes and identifiers are not confused. For example, in the disk tables, Disk_Name is an identifier, not an attribute. It is unique within the query subject. String values are automatically categorized as Attributes. You might need to change their usage to Identifier.

Framework Manager workflow



© Copyright IBM Corporation 2016

1-22

Framework Manager workflow

The following list defines the Framework Manager workflow:

- Import Metadata: Import objects such as tables, views, and procedures.
- Prepare Metadata: Examine, and modify properties, and relationships.
- Model Metadata for Reporting: Add business value specific to reporting requirements, and model for predictable results.
- Create, and Manage Packages: Identify subsets of metadata to be published.
- Set Security: Apply security at various levels to restrict access.
- Publish: Publish packages to the IBM Cognos servers for use by report authors and business analysts.



Lesson 3 Modifying the data model



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 use Framework Manager to model the Netcool/OMNIbus data model.

Requirements

To modify a data model, you need the following items:

- Framework Manager is installed
- Access to the report archive database
 - Framework Manager extracts database structure
- DB2 client libraries installed
 - Framework Manager uses the DB2 client to access the archive database
- Report package files that are used to create the data model

```
/opt/IBM/tivoli/netcool/omnibus/extensions/tcr_event_reports/Model/  
Netcool_OMNIbus_MODEL.cpf
```
- Access to Tivoli Common Reporting
 - Framework Manager connects to Tivoli Common Reporting to publish the revised data model

Framework Manager does not support a *stand-alone* mode

- Must connect to Tivoli Common Reporting
- Must connect to archive database

Requirements

The Framework Manager application requires real-time access to the Netcool/OMNIbus event archive database, and Tivoli Common Reporting. In most situations, these components are remote from the Framework Manager application.

The data model that is used for event reporting is included with Netcool/OMNIbus. The data model is contained within a report package. You can find the report package in the following location:

```
/opt/IBM/tivoli/netcool/omnibus/extensions/tcr_event_reports/Model
```

You must copy all of the files from the **Model** directory to the workstation where Framework Manager is installed.

Accessing report archive database

Framework Manager uses the Tivoli Common Reporting *data source name* to access the archive database

- Data source name is configured in Tivoli Common Reporting
- Data source name is configured in report package
- Data source name for the event archive is REPORTER

Framework Manager uses database client to access the archive database

- Install database client
- Define database connection to archive database
- Requires:
 - Server name
 - Port number
 - User name
 - Password

Accessing report archive database

Tivoli Common Reporting references a database with a data source definition. The data source name for the event archive database is REPORTER. Framework Manager access the archive database with the appropriate database client libraries, for example, DB2 or Oracle. The client libraries must be installed on the workstation with Framework Manager. In addition, you create a database alias with the name REPORTER. The database alias points to the actual archive database. To define the alias, you need the server, port number, user name, and password for the archive database.

Report package file

Netcool/OMNIbus includes the report package file

```
/opt/IBM/tivoli/netcool/omnibus/extensions/tcr_event_reports/Model/  
Netcool_OMNIbus_MODEL.cpf
```

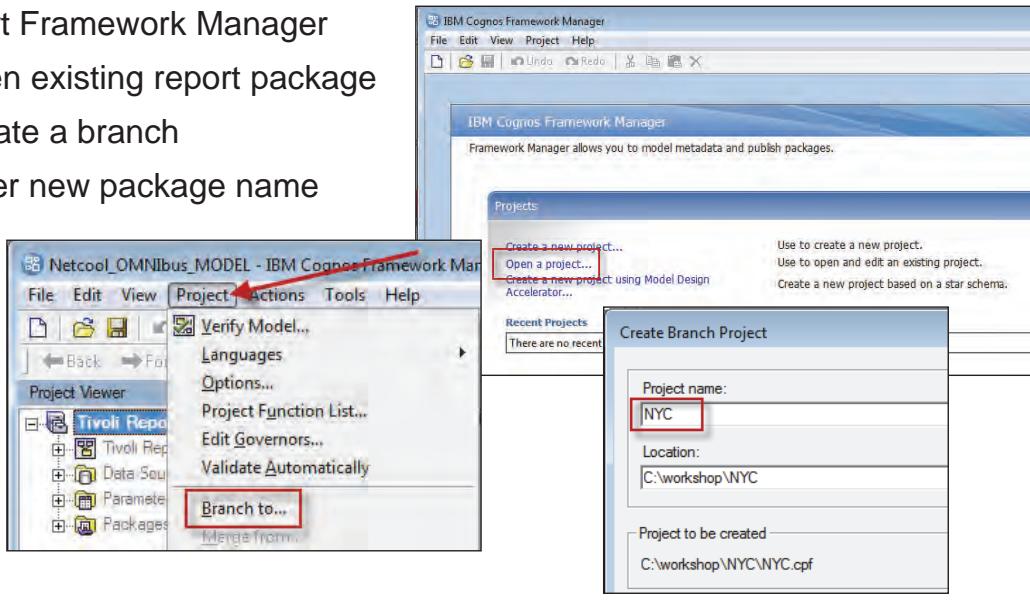
- You need all of the files in the Model directory

Report package file

The report package file is the output from Framework Manager. The file has a suffix of cpf. The existing package file is included with Netcool/OMNIbus. You must copy all of the files from the Model directory, not just the package file.

Copying the original model

1. Start Framework Manager
2. Open existing report package
3. Create a branch
4. Enter new package name



© Copyright IBM Corporation 2016

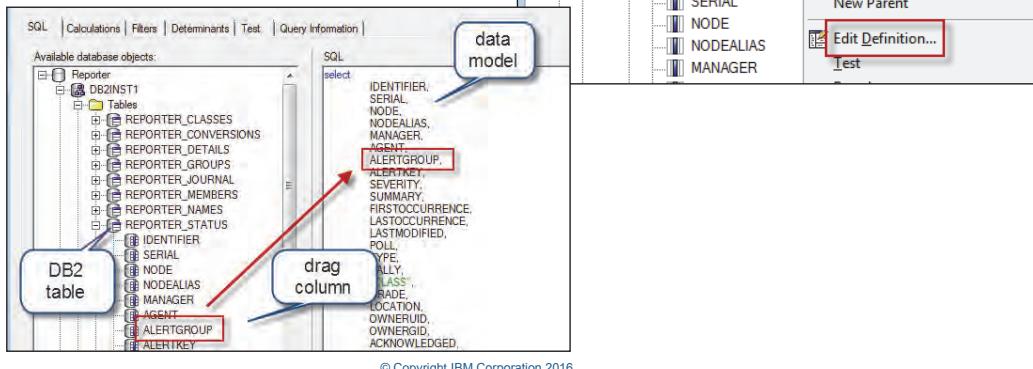
1-27

Copying the original model

To preserve the original data model, you create a copy of the report package. You start Framework Manager, and open a project. You browse the location where you copied the original files, and open the report package. The original data model opens in Framework Manager. You create what Framework Manager calls a *branch*, and provide a name. The name cannot contain spaces or special characters, except for the underscore. Then, you close Framework Manager.

Modifying the copy, Database View

1. Start Framework Manager
2. Open the new package
3. Expand Database View
4. Edit REPORTER_STATUS
5. Drag columns



1-28

Modifying the copy, Database View

You start Framework Manager, and open the copy of the package. The model contains four views:

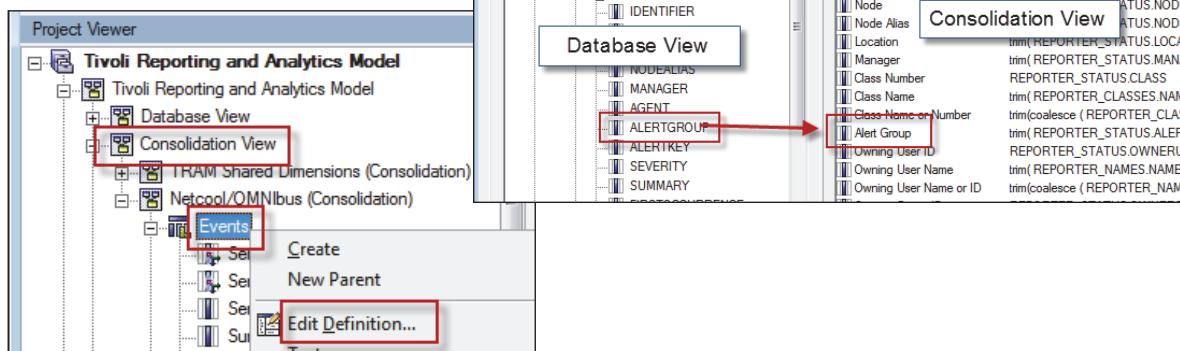
- Database view
- Consolidation view
- Dimensional view, which is not used
- Presentation view

The Database view is the physical representation of the database structure. The Database view contains the same objects as the physical database. You start the process of modifying the model by adding the extra columns to the Database view. You right-click the REPORTER_STATUS table in the Database view, and select Edit.

A window opens with two panes. The left pane represents the physical database. Framework Manager connects to the archive database, and extracts the physical structure in real time. The right pane represents what is contained in the data model. To add columns to the model, you drag the column from the left, and drop the column on the right. When complete, you click OK to save the changes. The Database view now contains the extra columns.

Modifying the copy, Consolidation View

1. Expand Consolidation View
2. Edit Events
3. Drag columns



© Copyright IBM Corporation 2016

1-29

Modifying the copy, Consolidation View

The Consolidation view is where the data model typically starts to hide the physical database structure. For example, the database might contain several tables. Each of the tables is represented in the Database view. In the Consolidation view, the data model can contain a single *table*, which is created from several database tables. You added columns to the Database view. Now you must add the same columns to the Consolidation view. You right-click the Events table in the Consolidation view, and select Edit.

A window opens with two panes. The left pane represents the Database view. The right pane represents the Consolidation view. To add columns, you drag a column from the left pane, and drop the column in the right pane. When complete, you click OK to save the changes.

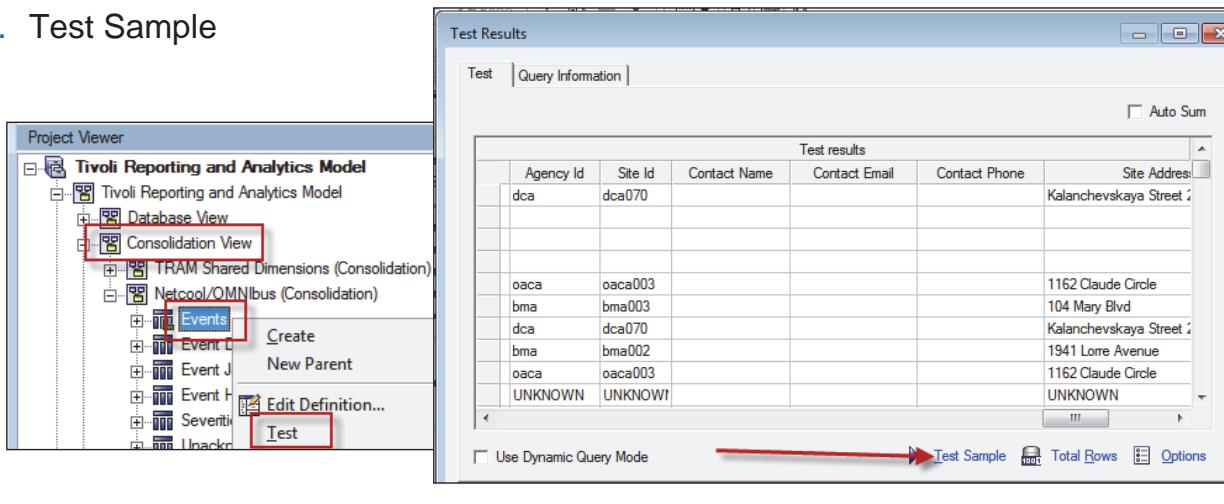
You typically change the column names in the Consolidation view to make them more meaningful. For example, database column names cannot contain spaces. You can change the column name in the Consolidation view, and add spaces.



Note: In the Database view, the event table is called REPORTER_STATUS. This name is the actual database table name. The corresponding table in the Consolidation view is called Events. The Events table in the Consolidation view contains data from multiple database tables, not just REPORTER_STATUS.

Modifying the copy, Testing the changes

1. Test Consolidation View
2. Test Sample



© Copyright IBM Corporation 2016

1-30

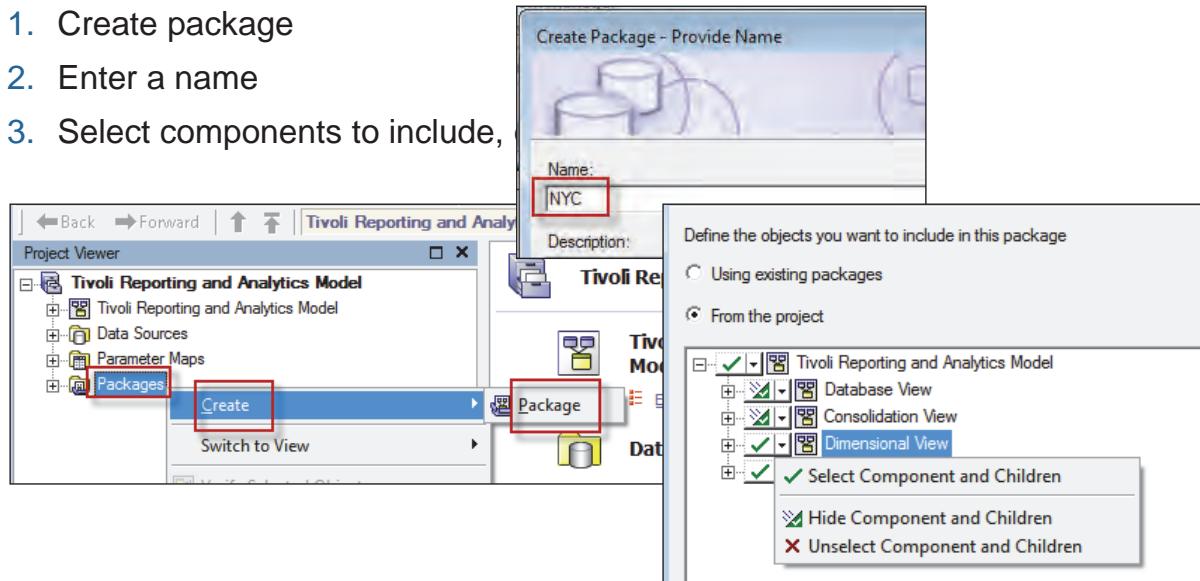
Modifying the copy, Testing the changes

You can verify whether your changes are correct with a test query of the database. Right-click the Events table in the Consolidation view, and select Test. A window opens. Click Test Sample on the bottom of the window. Framework Manager issues a query to the archive database to retrieve 25 records. The results appear in the window. The window contains every column name that is contained in the Consolidation view, not the Database view.

You can change the test configuration to retrieve more than 25 records if necessary.

Creating updated package

1. Create package
2. Enter a name
3. Select components to include,



© Copyright IBM Corporation 2016

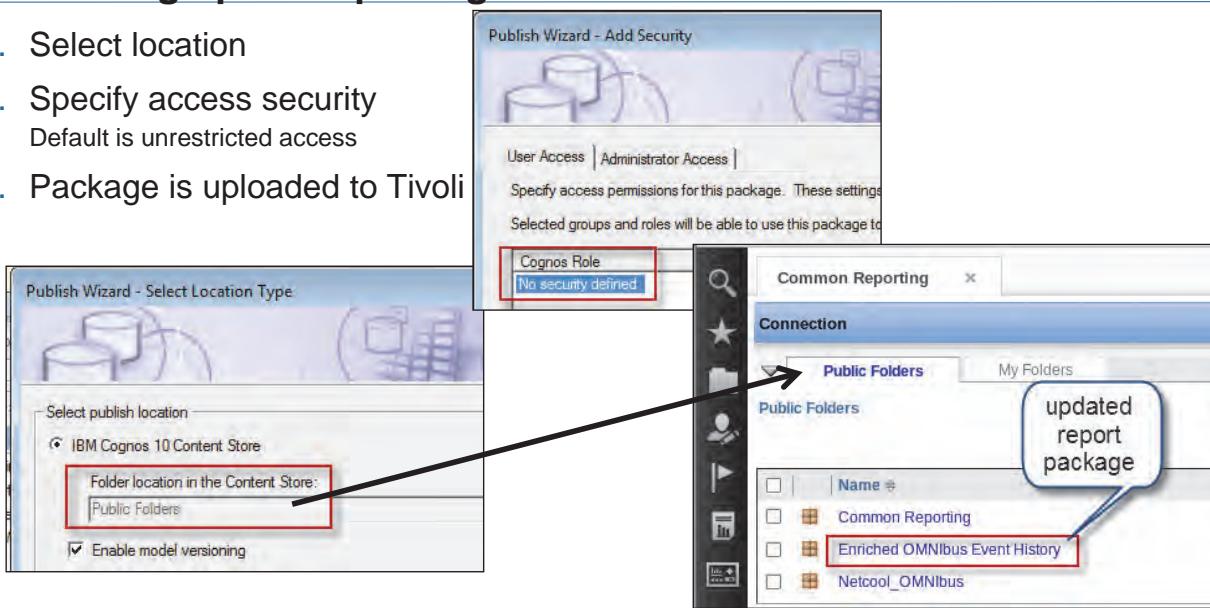
1-31

Creating updated package

After the views are modified, you create a report package. The report package is what users see when they use one of the Tivoli Common Reporting report author tools. You can control what components the user sees. For example, you do not want the user to see the physical database structure. Therefore, you *hide* the Database view in the package. You also *hide* the Consolidation view in the package. In the case of the event reporting data model, the user sees only the Presentation view.

Publishing updated package

1. Select location
2. Specify access security
Default is unrestricted access
3. Package is uploaded to Tivoli



© Copyright IBM Corporation 2016

1-32

Publishing updated package

After the package is created, you publish the package. It is the publish process that copies the package to Tivoli Common Reporting. After the package is published, it is immediately available for use within Tivoli Common Reporting.

Student exercises



© Copyright IBM Corporation 2015

33

Student exercises

Summary

You now should be able to perform the following tasks:

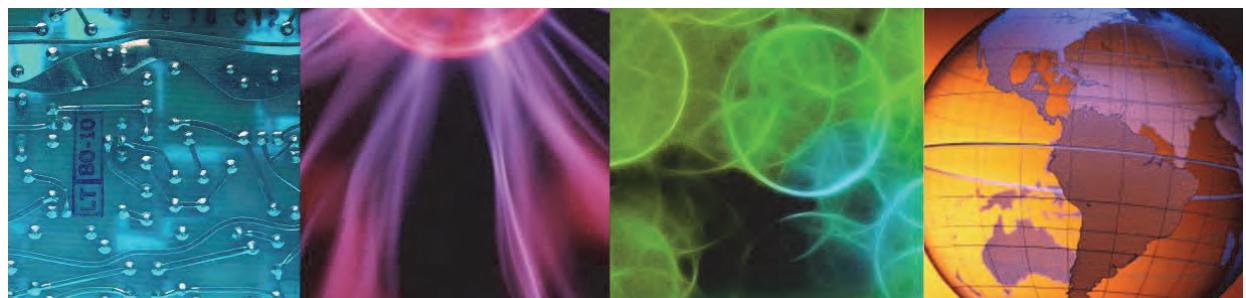
- Install and configure IBM Cognos Business Intelligence Modeling, also known as Framework Manager
- Use Framework Manager to modify the existing Cognos model for the event history database



8 Web GUI high availability



Web GUI high availability



© Copyright IBM Corporation 2016

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

In this unit, you learn how to create a WebSphere cluster, and implement Web GUI high availability.

References: SC27-6505-00 *Netcool/OMNibus Version 8 Release 1 Web GUI Administration, and User's Guide*

Objectives

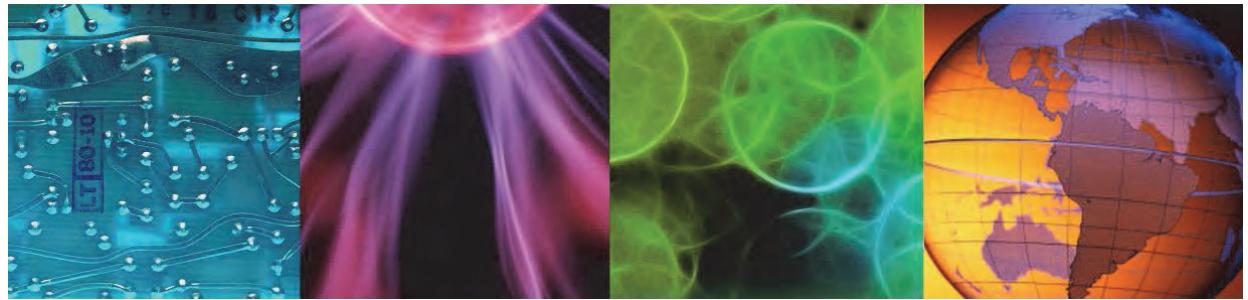
In this unit, you learn to perform the following tasks:

- Describe the components of a load balancing cluster
- Configure a load balancing cluster
- Use supplied tools to administer the cluster

Lesson 1 Dashboard Application Services Hub



Lesson 1 Dashboard Application Services Hub



© Copyright IBM Corporation 2016

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

In this lesson, you learn how a WebSphere cluster operates.

Load balancing environment

Group of servers that are linked together and operate as a single server

The name for the group of servers is a *cluster*

The cluster is a WebSphere cluster

Each of the servers is known as a *node*

Each node is running Dashboard Application Services Hub within WebSphere

The primary benefits of a cluster are as follows:

- Load balancing
 - Workload of servicing user requests is spread among the nodes
 - Improves the overall performance of the system
- Availability
 - Maintain the availability of network monitoring even if some cluster nodes are unavailable
 - For example, they are shut down for maintenance

Load balancing environment

A WebSphere cluster is a group of servers that are configured to operate in a common manner. The cluster consists of two or more servers, which are called nodes. The primary benefit is high availability of the applications that are supported within WebSphere. When a load balancer is implemented, the solution also offers improved performance due to the distribution of users across the nodes in the cluster.

Load balancing requirements

Lightweight Directory Access Protocol (LDAP) must be installed and configured as the user repository for each node in the cluster

A front-end Network Dispatcher (for example, IBM HTTP Server) must be set up to handle and distribute all incoming session requests

DB2 Version 10.1+ must be installed within the network to synchronize the global repositories for the console cluster

Each node in the cluster must be enabled to use the same LDAP with the same user and group configuration

All console nodes in load balancing cluster must be installed in the same cell name

All console nodes in load balancing cluster must run with synchronized clocks

The WebSphere Application Server and Jazz for Service Management application server versions must have the same release level, including any fix packs

Each node must use the same file-based repository user ID, which is assigned the role of *iscadmins*

Load balancing requirements

A number of requirements are necessary to implement a cluster. The primary requirements are DB2, and LDAP. DB2 is used to store the configurations from each node in the cluster. LDAP is used for a common user repository for all nodes in the cluster.

Synchronized data

Changes in the console are stored in global repositories

Changes are synchronized to all of the nodes in the cluster that uses a common database

The following actions cause changes to the global repositories:

- Creating, restoring, editing, or deleting a view
- Creating, editing, or deleting a preference profile
- Deploying preference profiles from the command line
- Copying a widget entity or deleting a widget copy
- Changing access to a widget entity, dashboard, external URL, or view
- Creating, editing, or deleting a role
- Changes to widget preferences or defaults
- Changes from the Users and Groups applications, including assigning users and groups to roles

Synchronized data

The cluster uses a DB2 database to save the configuration for each node in the cluster. When a change occurs on one node in the cluster, the change is written to node-specific tables in the database. After the database is updated, the other nodes are notified of the change. The other nodes retrieve the change from the database and apply the change to their configuration. Changes to most configuration objects that are used in WebSphere are saved in the database. Two notable exceptions are users and groups. Users and groups are maintained in LDAP so are not saved in the DB2 database.

Normal operation

During normal operation within a cluster:

1. Updates that require synchronization are first committed to the database
2. The node that submits the update for the global repositories notifies all other nodes in the cluster about the change
3. As the nodes are notified, they get the updates from the database and commit the change to the local configuration

If data fails to be committed on a node:

1. A warning message is logged in the log file
2. The node is prevented from making its own updates to the database

Restarting the Jazz for Service Management application server instance on the node rectifies most synchronization issues

© Copyright IBM Corporation 2016

7

Normal operation

Synchronization is automated and transparent to users. When a change occurs on a node in the cluster, the change is committed to the DB2 database. After the database is updated, the other nodes in the cluster are notified of the change. Each node retrieves the change from the database, and updates their configuration. When a change is made to a node, the database is also updated, and the change is committed to the node-specific tables.

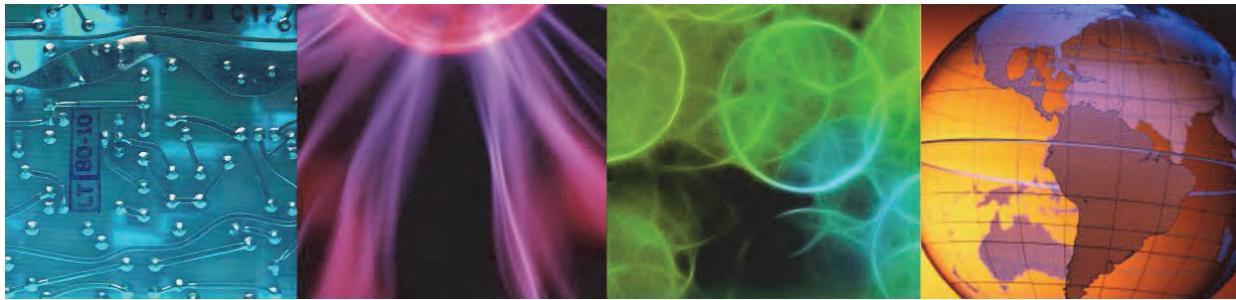
If a node is unable to commit a change, an entry is logged, and the node is prevented from updating the database until the problem is resolved.



Lesson 2 Web GUI



Lesson 2 Web GUI



© Copyright IBM Corporation 2016

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 Web GUI synchronization.

Web GUI in a load balancing environment

Web GUI is accessed through Dashboard Application Services Hub

Web GUI access is distributed across Dashboard Application Services Hub nodes in a cluster

The cluster maintains Dashboard Application Services Hub configuration data across all nodes

Web GUI has its own configuration data that must be maintained across nodes

© Copyright IBM Corporation 2016

9

Web GUI in a load balancing environment

You access the Web GUI application through Dashboard Application Services Hub. When WebSphere is configured as a cluster, Web GUI support is distributed across each node in the cluster. The WebSphere cluster automatically synchronizes WebSphere changes between nodes in the cluster. The cluster does not synchronize Web GUI changes.

For example, a Web GUI filter determines which events appear in the Active Event List. A user views the Active Event List through a Dashboard Application Services Hub page. Any changes to the page are synchronized through the cluster. Any changes to the filter are not synchronized through the cluster. The filter is specific to Web GUI.

Configuration data

Configuration data defines how a Web GUI server operates

It is held differently in a cluster than it is in a stand-alone server

- A Web GUI server holds its configuration data in the local file system
- In a cluster, the DB2 database holds the configuration data for the entire cluster
- The database is the master copy of the data that all the cluster nodes share
- A single set of configuration data means that each node is configured identically
- No configuration data is specific to a cluster node

Although the database holds the master copy, each node also has a copy in its local file system

- Allows the cluster to continue operation should the configuration database become unavailable
- When a node starts, it reads a complete set of configuration data from the database into the local file system and loads it into memory to improve performance

Configuration data

When a single instance of Web GUI is deployed, the configuration data is maintained in a local file system. When multiple servers are configured as a cluster that runs Web GUI, the operation changes. The configuration data for all instances of Web GUI is saved in the DB2 database, which ensures that all instances of Web GUI are configured the same.

Even though there is a common copy of configuration data for all instances of Web GUI, each instance of Web GUI also maintains a local copy of the configuration data to improve performance.

Configuration data examples

Web GUI configuration data consists of:

- Data source definitions
- Filters and views
- AEL menus and menu configuration data
- Metrics for gauges
- Prompts
- Tools
- User preferences
- AEL preferences such as the refresh time and the number of rows to display
- Web GUI properties such as the default time zone and the timeout period
- Maps and resources, together with their properties
- Gauges and their properties
- Charts and their properties
- Others

© Copyright IBM Corporation 2016

11

Configuration data examples

This slide shows the types of Web GUI configuration objects that are synchronized.

Configuration data changes

The process for changing configuration data is as follows:

1. A user on a node changes an item in the configuration and requests that the node saves the change, for example, creates a filter and saves it
2. The node writes the new information (for example a configuration file) to the database
3. The node notifies all the other cluster nodes that there is revised configuration information
4. The node updates its local copy of the configuration data to reflect the change
5. The other cluster nodes read the new information from the database and update the copies in their local file systems
6. The cluster continues to operate with the new configuration settings

Configuration data changes

Web GUI uses its own technique to detect changes. However, after the change is detected, the process to synchronize the change across the cluster is essentially the same. A change on one instance of Web GUI is written to the database, and all other instances are notified. Each instance of Web GUI retrieves the change from the database, and applies the change to their local configuration.

Detecting configuration data changes

Revised configuration data is automatically applied when it occurs through:

- Web GUI timed tasks facility
- A file that lists the files to be monitored and an associated set of monitor processes

Timed tasks determine when each node loads changed files from the database

The file is named *WEBGUI_HOME/etc/system/stores.lst*

- Contains a list of all the configuration files that are kept in the database
- When a node starts or joins the cluster, it creates a set of processes that monitor each of the files that are listed in stores.list
- Whenever a change occurs to one of those files, the corresponding process propagates the changed file to the DB2 database
- Notifies other nodes of the change

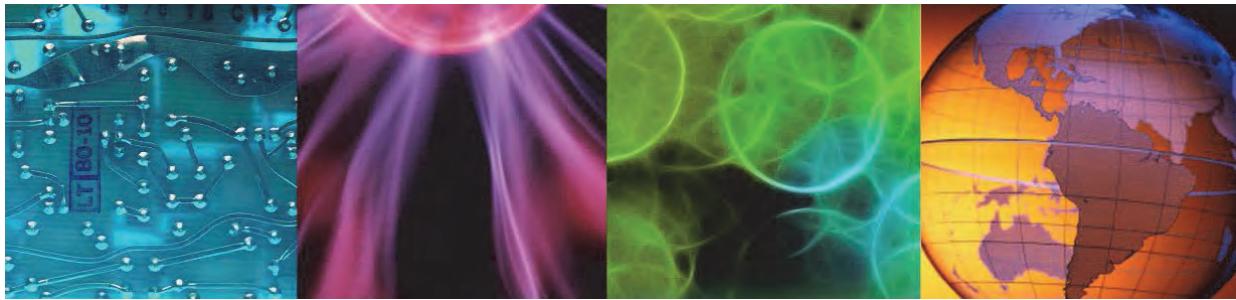
Detecting configuration data changes

Web GUI changes are detected by monitoring the local configuration files. A file contains of list of monitored configuration files. Users can add more files to this list. When a node starts, a process starts for every file in the list. Each process monitors an individual file. When a change is detected, the process commits the change to the database and notifies the other nodes in the cluster. The Web GUI component always commits changes to the local file system. It is the monitoring process that detects the changes, commits the changes to the database, and notifies the other instances of Web GUI.

Lesson 3 Setting up a load balanced cluster



Lesson 3 Setting up a load balanced cluster



© Copyright IBM Corporation 2016

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 deploy a WebSphere cluster and implement Web GUI synchronization.

Steps to create a Dashboard Application Services Hub cluster

1. Create the DB2 database that is used to save configuration data
2. Configure the high availability property file for the first server
3. Stop the first server
4. Create the cluster with the first server
5. Start the first server

The load balancing cluster is created and the console node is joined to the cluster as the first node
6. Add more nodes to the cluster
7. Configure server-to-server trust between all nodes in the cluster
8. Configure Web GUI load balancing
9. Install and configure load balancer

© Copyright IBM Corporation 2016

15

Steps to create a Dashboard Application Services Hub cluster

The high availability configuration consists of two distinct components: WebSphere cluster and Web GUI synchronization. You configure each component separately. This slide highlights the steps that are used to create a high availability configuration.

Creating the DB2 database

For the cluster to operate successfully, the DB2 database must be available

The database is the key coordination point of the cluster because it contains the configuration data

Create the database with any supported technique

For example, enter the following command through the DB2 command-line utility:

```
CREATE DATABASE LBCONFIG COLLATE USING SYSTEM PAGESIZE 8192 USER TABLESPACE  
MANAGED BY SYSTEM USING ('LBCONFIG') EXTENTSIZE 64 PREFETCHSIZE 32  
TEMPORARY TABLESPACE MANAGED BY SYSTEM USING ('LBCONFIGTemp') EXTENTSIZE 32  
PREFETCHSIZE 16
```

Consider configuring DB2 in a high availability configuration

Creating the DB2 database

The DB2 database is the most important component in the configuration. All changes are saved in the database. You create the database with any supported technique. There is nothing special about this database. Because it is a critical component for cluster operation, a high-availability cluster is suggested.

High availability properties

Dashboard Application Services Hub high availability property file

/opt/IBM/JazzSM/ui/bin/ha/tipha.properties

Change the following values:

- | | |
|--------------------|--|
| ▪ DBHost | the host name or IP address for the DB2 server |
| ▪ DBPort | the DB2 port number |
| ▪ DBName | configuration database name |
| ▪ DBJDBCDriverPath | location of the DB2 JDBC driver files |
| ▪ LocalHost | host name for the local copy of Dashboard Application Services Hub |
| ▪ LocalPort | port number for the local copy of Dashboard Application Services Hub |
| ▪ WasRoot | the location where WebSphere is installed |
| ▪ TipHome | the location where Dashboard Application Services Hub is installed |
| ▪ ProfilePath | the location of the Jazz for Service Management profile |
| ▪ ProfileName | the name of the Jazz for Service Management profile |
| ▪ CellName | the WebSphere Cell name |
| ▪ NodeName | the WebSphere host name |

Change the corresponding file on every server in the cluster

© Copyright IBM Corporation 2016

17

High availability properties

A property file defines the aspects of the WebSphere cluster. This property file contains information such as the access details for the DB2 database. The file also defines the WebSphere cell name. All nodes in the cluster must use the same cell name.

Creating the cluster

The cluster is created when you add the first server

Steps to add to the first server:

- 1.Modify the high availability property file
- 2.Stop Dashboard Application Services Hub
- 3.Add the server to the cluster

```
cd /opt/IBM/JazzSM/ui/bin/ha  
/opt/IBM/WebSphere/AppServer/bin/ws_ant.sh -f install.ant configHA -  
Dusername=<database user ID> -Dpassword=<database password> -  
DWAS_username=<WebSphere admin user> -DWAS_password=<WebSphere password>
```

Command creates and populates tables in the configuration database for the first server in the cluster

- 4.Start Dashboard Application Services Hub

The cluster contains the first node

© Copyright IBM Corporation 2016

18

Creating the cluster

To create the cluster, you add the first node. You configure the high availability property file, which was described previously, and enter the following command:

```
/opt/IBM/WebSphere/AppServer/bin/ws_ant.sh -f install.ant configHA  
-Dusername=<database user ID> -Dpassword=<database password>  
-DWAS_username=<WebSphere admin user> -DWAS_password=<WebSphere password>
```

The command creates the tables in the DB2 database for the corresponding node. After the command completes, you start Dashboard Application Services Hub. When the application starts, the configuration data is written to DB2, and the node joins the cluster.

Adding more nodes to the cluster

The process to add a node to the cluster is the same

Steps to add a server:

- 1.Modify the high availability property file
- 2.Stop Dashboard Application Services Hub
- 3.Add the server to the cluster

```
cd /opt/IBM/JazzSM/ui/bin/ha  
/opt/IBM/WebSphere/AppServer/bin/ws_ant.sh -f install.ant configHA -  
Dusername=<database user ID> -Dpassword=<database password> -  
DWAS_username=<WebSphere admin user> -DWAS_password=<WebSphere password>
```

Command creates and populates tables in the configuration database for the next server in the cluster

- 4.Start Dashboard Application Services Hub

The cluster contains two nodes

© Copyright IBM Corporation 2016

19

Adding more nodes to the cluster

You add more nodes to the cluster with the same process. You update the high availability property file and run the command. The command creates the tables for the node in DB2. You start the node, and the node joins the cluster. You repeat this process for every node in the cluster.

Configuring server-to-server trust, 1 of 3

Enable load balanced nodes to connect to each other and send notifications

Modify the SSL client property file

/opt/IBM/JazzSM/profile/properties/ssl.client.props

Uncomment the following lines:

```
#  
# Another SSL configuration (this is a template, uncomment and modify)  
# You can configure the dynamicSelectionInfo OR reference this alias  
# from another protocol (e.g., soap.client.props or sas.client.props)  
#  
com.ibm.ssl.alias=AnotherSSLSettings  
com.ibm.ssl.protocol=SSL_TLS  
com.ibm.ssl.securityLevel=HIGH  
com.ibm.ssl.trustManager=IbmX509  
com.ibm.ssl.keyManager=IbmX509  
com.ibm.ssl.contextProvider=IBMJSSE2  
com.ibm.ssl.enableSignerExchangePrompt=true  
#com.ibm.ssl.keyStoreClientAlias=default  
#com.ibm.ssl.customTrustManagers=  
#com.ibm.ssl.customKeyManager=  
#com.ibm.ssl.dynamicSelectionInfo=  
#com.ibm.ssl.enabledCipherSuites=
```

uncomment
these lines

Configuring server-to-server trust, 1 of 3

You must configure server-to-server trust between all nodes in the cluster. Server-to-server trust is a requirement of load balancing that enables the nodes in the cluster to connect to each other, and send notifications. The process involves two steps. In step one, you configure a property file on each node in the cluster as shown on this slide, and the following slide. In step two, you exchange certificates between all nodes in the cluster.

Configuring server-to-server trust, 2 of 3

Uncomment the following lines:

```
# TrustStore information
com.ibm.ssl.trustStoreName=AnotherTrustStore
com.ibm.ssl.trustStore=${user.root}/etc/trust.p12
com.ibm.ssl.trustStorePassword={xor}CDo9Hgw=
com.ibm.ssl.trustStoreType=PKCS12
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=true
com.ibm.ssl.trustStoreReadOnly=false
```

uncomment
these lines

Modify the following line:

```
# TrustStore information
com.ibm.ssl.trustStoreName=AnotherTrustStore
#com.ibm.ssl.trustStore=${user.root}/etc/trust.p12
com.ibm.ssl.trustStore=${user.root}/config/cells/TIPCell/nodes/TIPNode/trust.p12
com.ibm.ssl.trustStorePassword={xor}CDo9Hgw=
com.ibm.ssl.trustStoreType=PKCS12
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=true
```

Save the changes

Repeat this process for all nodes in the cluster

© Copyright IBM Corporation 2016

21

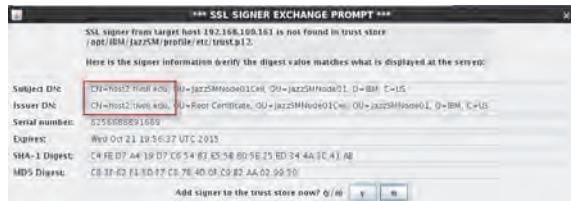
Configuring server-to-server trust, 2 of 3

Configuring server-to-server trust, 3 of 3

Retrieve Signer Certificate from remote host

```
cd /opt/IBM/JazzSM/profile/bin
./retrieveSigners.sh NodeDefaultTrustStore AnotherTrustStore -host <remote server>
-port 16313
```

Click Y to save the certificate



Enter credentials for remote server



Repeat on every node in the cluster

Configuring server-to-server trust, 3 of 3

You must exchange certificates between all nodes in the cluster. The certificates are retrieved remotely from each node with the following command:

```
/opt/IBM/JazzSM/profile/bin/retrieveSigners.sh NodeDefaultTrustStore
AnotherTrustStore -host <remote server> -port 16313
```

The host property identifies the remote server. When the command runs, a prompt asks for the user credentials of the remote server. The user that is required is the WebSphere administrator user.

After server-to-server trust is configured, the WebSphere cluster is complete.

Configuring Web GUI load balancing

Modify list of monitored files, if necessary

`WEBGUI_HOME/etc/system/stores.lst`

You can add files to the list, but do not remove any files from the list

Modify the server initialization file as follows:

```
WEBGUI_HOME/etc/server.init
cluster.mode:on
cluster.hostname:<local host name or IP address>
cluster.port:16311
timedtasks.enabled:true
timedtasks.default.startdelay: 120
timedtasks.default.runperiod: 60
```

Number of seconds to wait before starting to
monitor for changes
Frequency of checks

Restart Dashboard Application Services Hub

© Copyright IBM Corporation 2016

23

Configuring Web GUI load balancing

The configuration of Web GUI synchronization is a separate process. You enable synchronization on each node with modifications to the server.init file. After you change the file, you must restart Dashboard Application Services Hub.

IBM HTTP Server

Software load balancer

- Optional component
- Hardware load balancers are also supported

Based on Apache HTTP Server 2.2.8, with extra fixes

Installation files are included with WebSphere

- IBM WebSphere Application Server V8.5 Supplements (1 of 3)
- IBM WebSphere Application Server V8.5 Supplements (2 of 3)
- IBM WebSphere Application Server V8.5 Supplements (3 of 3)

Installed with IBM Installation Manager

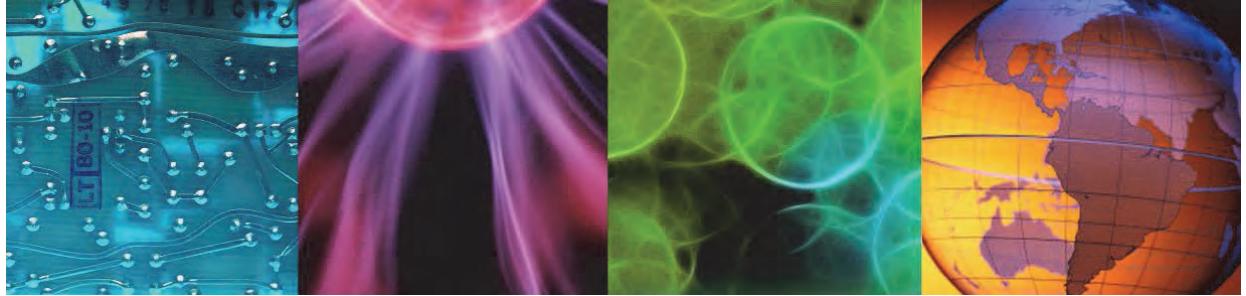
IBM HTTP Server

The cluster ensures that changes on one node are propagated to all nodes. However, if a node fails, the cluster does not redirect users of that node to another node. To redirect users between nodes in the cluster you must use a load balancer. You can use a hardware load balancer, or a software load balancer. The IBM HTTP Server application is one option for a software load balancer.



Lesson 4 Cluster administration

Lesson 4 Cluster administration



© Copyright IBM Corporation 2016

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 use the available commands to administer a cluster.

Cluster administration for WebSphere

A cluster requires little administration

Two parts to the cluster:

- WebSphere
- Web GUI

Separate tools to administer each part

- WebSphere tools
 - Monitoring the state of the cluster

```
DASH_HOME/bin/ha/HATool.sh modules <DB2 username> <DB2 password> -byNodes -showAll
```
 - Removing a node from the cluster

```
JazzSM_WAS_Profile/bin/ws_ant.sh -f uninstall.ant
disjoin -DdeleteExistingDataSource=true -Dusername=DB2_username
-Dpassword=DB2password -DWAS_username=WAS_admin_username
-DWAS_password=WAS_admin_password
```

Cluster administration for WebSphere

You use separate tools to administer the WebSphere cluster and Web GUI synchronization. There are three administrative functions for WebSphere cluster management:

1. Adding a node to the cluster, which was described previously.
2. Removing a node from the cluster.
3. Monitoring the state of the cluster.

Before you change a node in the cluster, you must remove the node from the cluster. For example, you install a fix pack to Dashboard Application Services Hub. You use the following command to remove the node:

```
JazzSM_WAS_Profile/bin/ws_ant.sh -f uninstall.ant
disjoin -DdeleteExistingDataSource=true -Dusername=DB2_username
-Dpassword=DB2password -DWAS_username=WAS_admin_username
-DWAS_password=WAS_admin_password
```

The command removes the node from the cluster, and removes the corresponding database tables. After the maintenance is complete, you can add the node to the cluster.

You use the following command to monitor the status of the cluster:

```
DASH_HOME/bin/ha/HATool.sh modules <DB2 username> <DB2 password> -byNodes
-showAll
```

The command queries the DB2 database to determine the state of the cluster, and the member nodes.

Cluster administration for Web GUI

Removing a node from the cluster

1. Run Web GUI API command

```
WEBGUI_HOME/waapi/etc/samples/cluster_removenode.xml
```

Removes the node from a load balancing cluster

If this node is the last node in the cluster, the method also removes the load balancing configuration data

2. Modify server.init

```
WEBGUI_HOME/etc/server.init
cluster.mode:off
timedtasks.enabled:false
```

3. Restart Dashboard Application Services Hub

Cluster administration for Web GUI

Web GUI synchronization ensures that all instances in the cluster are configured alike. You use the Web GUI Administrative API (WAAPI) utility to remove an instance of Web GUI from the cluster. If you want to remove the instance permanently, you must also modify the server.init file as shown.

Web GUI load balancing best practices

Timed tasks

- Ensures that all changes are detected and loaded into the server, without the need to restart the server
- Ensure that the **timedtasks.enabled** property in the server.init file is set to **true**

The configuration database

- Always ensure that the database is available before you change the configuration
- Use the Web GUI itself to change the configuration
- The Web GUI always checks that the database is available before you are allowed to save any changes

Custom web content

- Place any custom web content, such as HTML file, in subdirectories of `JazzSM_HOME/installedApps/JazzSMNode01Cell/isc.ear/OMNIbusWebGUI.war`
- In addition, add these directories to the list of files to maintain in the database

© Copyright IBM Corporation 2016

28

Web GUI load balancing best practices

The slide highlights some of the suggested best practices for Web GUI load balancing.

The timed tasks feature ensures that changes are propagated. Make sure that you enable the feature on all instances.

When Web GUI is configured as part of cluster, you cannot save a change if the DB2 database is not available. For example, you use the filter builder in Dashboard Application Services Hub to change a filter configuration. If the database is not available, you cannot save the modified filter. If you are an administrator, you know immediately that there is an issue with the database. If you use the Web GUI Administrative API utility to modify a filter, the change is applied to the local configuration file. However, the change is not committed to the database, and the other nodes are not notified of the change.



Note: The example that is used here references a change to a filter. The same holds true for a change to any Web GUI object.

Student exercises



Perform the exercises for these lessons.

Summary

You now should be able to perform the following tasks:

- Describe the components of a load balancing cluster
- Configure a load balancing cluster
- Use supplied tools to administer the cluster



9 Security



Security



© Copyright IBM Corporation 2016

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 features of Netcool/OMNibus related to security.

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

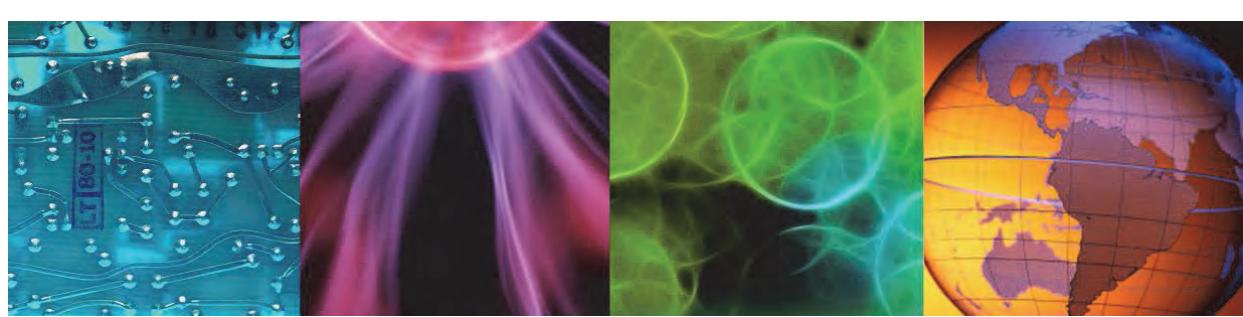
Objectives

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

- Implement AES, and AES_FIPS property value encryption
- Configure, and use SSL communications on core components
 - Create a certificate authority
 - Create Certificate requests
 - Sign, and accept certificate requests
- Configure FIPS-140-2 security compliance on core components



Lesson 1 Netcool/OMNIbus security elements



© Copyright IBM Corporation 2016

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

In this lesson, you learn how security is implemented within Netcool/OMNIbus.

Main components

User access security

- Protects Tivoli Netcool/OMNibus system from accidental or deliberate damage that is caused by users or potential users of the system
 - Authentication verifies the identity of the user
 - Authorization verifies the rights to view and modify information

Communication security

- Protects connections between different components of the Tivoli Netcool/OMNibus system
 - Secure Socket Layer (SSL/TLS) protocol provides confidentiality, authenticity, and integrity of information between components

Main components

The two primary aspects of security are user access security, and communication security. User access security prevents deliberate or accidental damage to the system. User security implements authentication, and authorization. Authentication establishes the identity of the user, which involves password verification. The assumption is that a valid user, and correct password establish the identity of the user. After a user gains access to the system, authorization controls what the user can do.

Communication security protects the information that passes between Netcool/OMNibus components. Netcool/OMNibus servers can be configured to support access with Secure Socket Layer (SSL) protocol. The SSL protocol provides two primary features. First, the protocol ensures the identity of the communication components. A certificate exchange establishes the identity of the server component. The second feature of the protocol is encryption. The contents of the transmission are encrypted. If some one captures the packets in a communication stream, encryption renders the contents unreadable.

User authentication

All Tivoli Netcool/OMNIbus users have a user name with an associated password

The user name and password combination is authenticated before a connection is made to the ObjectServer

Password encryption provides more security

Users can be authenticated in the ObjectServer, or externally authenticated

- On UNIX and Linux systems, ObjectServer users can be externally authenticated with Pluggable Authentication Modules (PAM) or a Lightweight Directory Access Protocol (LDAP) system
- On Windows systems, users can be externally authenticated by using LDAP

User authentication

All users must have a user name and password. The user name, and password is verified before the user gains access to the ObjectServer. A user name is stored in a table within the ObjectServer. The corresponding password can be stored *internally* within the table, or *externally*. For UNIX, and Linux deployments, the choice for external storage is either the host operating system, or LDAP. For Windows deployments, LDAP is the only option for external authentication.

The password values are encrypted to prevent some one from discovering their value.

Secure mode authentication

If the ObjectServer is **not** running in secure mode, probes and gateways connect without a user name and password

Secure mode is enabled with an ObjectServer property

SecureMode: TRUE

Default setting is FALSE

Probes must provide an ObjectServer user name and password

AuthPassword : 'object00'

AuthUserName : 'root'

Gateways must provide an ObjectServer user name and password

Gate.RdrWtr.Username : 'root'

Gate.RdrWtr.Password : 'object00'

Example for the JDBC gateway

Passwords can be encrypted for enhanced security

© Copyright IBM Corporation 2016

6

Secure mode authentication

Probes, and gateways do not require a user name, and password to access the ObjectServer unless the ObjectServer is configured for *secure mode*. Secure mode is an option, and is enabled through an ObjectServer property. When the ObjectServer is configured for secure mode, a probe or gateway must provide an ObjectServer user name, and password. The user name, and password strings are configured in the probe or gateway property file.

It is not necessary to encrypt the password string in the property file. The string can appear in the file in clear text. However, for added security, you can encrypt the values if wanted.

Configuring ObjectServer user authentication

If users are authenticated in the ObjectServer, you can configure

- The encryption algorithm for passwords
- The password format

Encryption algorithm

- Data Encryption Standard
 - PasswordEncryption: 'DES'
 - Only the first 8 characters are read
- Advanced Encryption Standard
 - PasswordEncryption: 'AES'
 - Only the first 16 characters are read

Password format

- Forces password to conform to minimum length, and context
- RestrictPasswords: TRUE
PasswordFormat: '8:1:1:0'

© Copyright IBM Corporation 2016

7

Configuring ObjectServer user authentication

When users are authenticated with the ObjectServer, the password values are always encrypted before they are saved to the table. The default encryption algorithm is based on the Data Encryption Standard (DES). When DES is used for encryption, only the first 8 characters of the password string are read. The remaining characters are ignored. The second option for encryption algorithm is based on the Advanced Encryption Standard (AES). When AES is used for encryption, only the first 16 characters are read, and the remaining characters are ignored. The choice of algorithm is configured in the PasswordEncryption ObjectServer property.

A second feature that is related to passwords is used to force password values to comply with minimum standards. The option is enabled with the RestrictPasswords property. The option is configured with the PasswordFormat property. Specify the value of the property as a colon-separated set of integer values. Each value defines a password requirement. The format is:

min_len:alpha_num:digit_num:punct_num

- min_len is the password length.
- alpha_num is the minimum number of alphabetic characters.
- digit_num is the minimum number of numeric characters.
- punct_num is the minimum number of punctuation characters.

User authorization

Permissions control access to objects and data in the ObjectServer

For example, INSERT, UPDATE, ALTER, DROP

Roles combine permissions

Groups combine roles into logical entities

For example, System group contains the roles that grant permission to change all objects

Users belong to one or more groups

- Inherit the corresponding permissions

Example read-only user

- Create role READONLY with SELECT permission for alerts.status table
- Assign role to READONLY group
- Assign user to READONLY group

User authorization

Each user is assigned to one or more groups. You can assign permissions to each group to perform actions on database objects by granting one or more roles to the group. You can create logical groupings such as super users or system administrators, physical groupings such as London or New York NOCs, or any other groupings to simplify your security setup.

For example, creating automations requires knowledge of Tivoli Netcool/OMNibus operations and the way that a particular ObjectServer is configured. You do not typically want all of your users to create or modify automations. One solution is to create a role that is named AutoAdmin, with permissions to create and modify trigger groups, files, SQL and external procedures, and signals. You can then grant that role to a group of administrators who create, and update automations.

Auditing authentication

An important aspect of user security is verification

Activate logging to track authentication issues

ObjectServer properties:

```
Sec.AuditLog $NHOME/omnibus/log/servername/audit.log  
Sec.AuditLevel [ debug, info, warn, error ]
```

The default is warn

The debug and info levels generate messages for authentication successes and failures

The warn and error levels generate messages for authentication failures only

© Copyright IBM Corporation 2016

9

Auditing authentication

When you secure applications, it is important to monitor, or audit, the effectiveness of your security. One way to monitor effectiveness is to configure the system to log certain types of messages. Then, you can monitor the logs to see whether anything of interest or concern occurred.

The ObjectServer is created with auditing enabled. The default level is warn, which produces an event for every authentication failure, for example:

```
2015-01-21T20:33:16: Error: E-SEC-010-002: authentication failure - cannot  
authenticate user "abraman" : Not authenticated
```

Property value encryption

Encrypt string values in a properties file or configuration file so that the strings cannot be read without a key

Prevents a human from seeing the property value in clear text

For example, passwords

When the process that uses the properties file or configuration file starts up, the strings are decrypted

The encryption mechanism is used in ObjectServer, proxy server, probe, gateway properties files, and process agent configuration files

The encryption mechanism uses:

- Advanced Encryption Standard (AES)
- Supports keys of 128, 192, and 256 bits
- Command-line key generator (nco_keygen)
- Encryption utility (nco_aes_crypt)

© Copyright IBM Corporation 2016

10

Property value encryption

To enhance the level of security, you can encrypt property values, which prevents a human from determining the value. Property value encryption uses a key, which is stored in a file. You use the key to encrypt the value, and the component uses the key to decrypt the value. The most common property values that are encrypted are typically password strings.

The encryption process is local to any component, like a probe. You can encrypt the value of a password string, for example, and paste the value in a probe property file. When the probe starts, it decrypts the password back to clear text. When the probe communicates with an ObjectServer that is configured in secure mode, the probe sends the user name, and password to the ObjectServer in clear text.



Important: Property value encryption is separate from ObjectServer password encryption.

Communication security

- Secure Socket Layer (SSL) or also known as Transport Layer Security (TLS) is a secure data transmission protocol
- SSL is a cryptographic system that uses two keys to encrypt data
 - Public key
 - Private or secret key
- SSL uses digital certificates for key exchange and authentication
 - When a client initiates an SSL connection, the server presents the client with a certificate signed by a certificate authority (CA)
 - The client then acts upon that information
- Several of these encryption standards are used in the java.security file

© Copyright IBM Corporation 2016

11

Communication security

Netcool/OMNIbus server components can be configured to communicate with clients through the Secure Socket Layer (SSL) protocol. The protocol ensures the integrity of the communication by using certificates, and encryption.

SSL concepts: Certificates

- Three actors in any SSL connection:
 - ▶ Client, Server, Authority
- The authority signs its own certificate and should be a trusted party.
- The authority signs the server's certificate with its private key
- The authority provides the client with its public key



- On connection establishment the server sends its signed public key to the client
- The client checks this public key against the authority certificate to verify that the server is trusted
- Session key exchange can now take place using the server public key as a basis for secrecy

© Copyright IBM Corporation 2016

12

SSL concepts: Certificates

SSL uses digital certificates for key exchange and authentication. When a client initiates an SSL connection, the server presents the client with a certificate that signed by a certificate authority (CA). A certificate authority is a trusted party that guarantees the identity of the certificate and its creator. The server certificate contains the identity of the server, the public key, and the digital signature of the certificate issuer.

By reading the server certificate, the client can determine whether the server is a trusted source, and then accept or reject the connection. To verify the signature on the server certificate, the client requires the public key of the issuing CA. Because public keys are distributed in certificates, the client must have a certificate for the issuing CA. The CA must sign this certificate.

Server certificates can be generated for ObjectServers, process agents, and proxy servers.

Certificates serve two specific purposes:

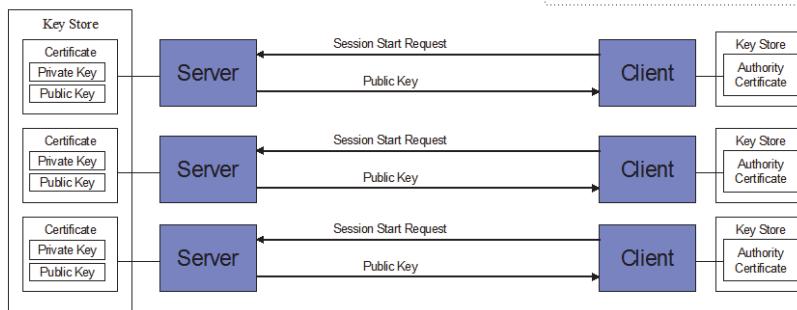
- When a client connects to a server, the certificate proves to the client that the company that installed the certificate owns the server.
- They contain the public key that the client uses to establish an encrypted connection to the server.

FIPS 140-2 approved cryptographic providers supply all encryption, and key generation functions that are required for the secured SSL connections.

SSL concepts: Keystores

SSL Concepts: Key Stores

- ▶ Server and trusted certificates now stored in CMS keystores
- ▶ Single files representing a password protected key database
- ▶ Keys referenced by label
- ▶ Labels match objectserver or process agent name in omni.dat/sql.ini
- ▶ Integral to generating and using certificates



© Copyright IBM Corporation 2016

13

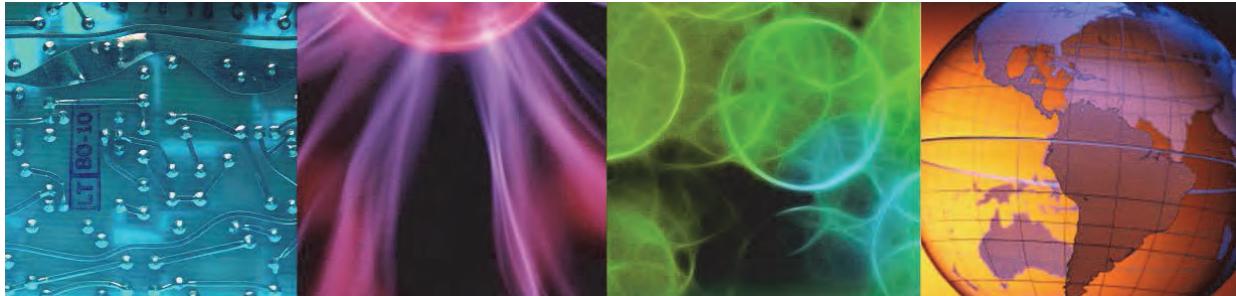
SSL concepts: Keystores

Every Netcool/OMNibus component that participates in SSL communications requires a local key database. The database contains the certificates that are used with SSL. You create the key database, and assign a password. The password prevents unauthorized access to the database.

Lesson 2 Using SSL for client and server communications



Lesson 2 Using SSL for client and server communications



© Copyright IBM Corporation 2016

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 SSL communication between the ObjectServer, and various clients.

Steps for configuring SSL

1. Create entries for SSL connections in the Server Editor
Update the interfaces file, and add SSL port numbers
2. If you are running in FIPS 140–2 mode, configure cryptographic properties for encryption, and key generation
3. Create, and distribute certificates, and keys to servers and clients
4. Configure clients for SSL access
5. Configure an SSL connection between Dashboard Application Services Hub, and the user repository
Only required if the ObjectServer is configured as a user repository
6. Configure SSL for the event feed to the Web GUI

© Copyright IBM Corporation 2016

15

Steps for configuring SSL

This slide illustrates the steps that are required to implement SSL communication between the ObjectServer, and clients.

In the following slides, the term *server* refers to an ObjectServer. The term *client* refers to a probe, gateway, or native desktop.

Modifying the interfaces file

Add an SSL port number for each ObjectServer, for example:

Server	Hostname	Port	SSL
NCOHS	host1.tivoli.edu	4100	4110
Backup1	host2.tivoli.edu	4100	4110
PSERV	host1.tivoli.edu	4100	4110
PSERV_B	host2.tivoli.edu	4100	4110

Configures ObjectServer to accept connections on either SSL or non-SSL ports

Modifying the interfaces file

The interfaces file defines the port numbers that are used for various Netcool/OMNIBus components, including the ObjectServer. The ObjectServer can support non-SSL or SSL communications, or both. The interfaces file defines what ports are used for each type of communication. If you want both types of communication available, you must define both port numbers. With both port numbers defined, the ObjectServer accepts connections on non-SSL or SSL ports.

The screen capture on this slide shows the configuration of a high availability pair of ObjectServers. The primary ObjectServer, PSERV, runs on host1.tivoli.edu, and listens for non-SSL connections on port 4100. The ObjectServer also listens for SSL connections on port 4110. The backup ObjectServer, PSERV_B, runs on host2.tivoli.edu, and listens for non-SSL connections on port 4100. The ObjectServer also listens for SSL connections on port 4110.

Configuring cryptographic properties

You must configure cryptographic properties for encryption, and key generation if you are running in FIPS 140–2 mode

1. Modify the `java.security` file
2. Update security providers exactly as follows:

```
security.provider.1=com.ibm.fips.jsse.IBMJSSEFIPSProvider      Add this line
security.provider.2=com.ibm.crypto.fips.provider.IBMJCEFIPS    Add this line
security.provider.3=com.ibm.jsse2.IBMJSSEProvider2
security.provider.4=com.ibm.crypto.provider.IBMJCE
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.security.sasl.IBMSASL
security.provider.8=com.ibm.xml.crypto.IBMXMLCryptoProvider
security.provider.9=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.10=org.apache.harmony.security.provider.PolicyProvider
security.provider.11=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
security.provider.12=com.ibm.security.cmskeystore.CMSProvider
```

© Copyright IBM Corporation 2016

17

Configuring cryptographic properties

To configure the Java Runtime Environment (JRE) supplied with Netcool/OMNIbus to work with FIPS 140–2 encryption, you must change the configuration of the `java.security` file. There are three areas within the file that require modification. In the first area, you must add definitions for two security providers, as shown in the slide.



Important: The changes to this file are required only if the ObjectServer runs in FIPS 140-2 compliance mode.

Configuring cryptographic properties, continued

3. Set Key, and Trust Manager and SSL Factory with the following entries:

```
ssl.KeyManagerFactory.algorithm=IbmX509  
ssl.TrustManagerFactory.algorithm=IbmX509  
...  
ssl.SocketFactory.provider=com.ibm.jsse2.SSLSocketFactoryImpl  
ssl.ServerSocketFactory.provider=com.ibm.jsse2.SSLServerSocketFactoryImpl
```

Configuring cryptographic properties, continued

In addition to the security providers, you must also modify two other sections of the file. You must set the key, trust manager, and SSL factory settings as shown on the slide.

Steps for creating, and distributing keys, and certificates

Perform the following steps on all computers that host an ObjectServer component

1. Create key database

2. Create self-signed certificates

Only necessary if you do not use an external certificate authority

3. Distribute self-signed certificates

Only necessary if you do not use an external certificate authority

4. Request and send certificate requests for all server components

If not using an external authority, use the self-signed certificate to sign the certificate requests

Required for every computer that hosts an ObjectServer, proxy server, or process agent

5. Receive the signed certificates

© Copyright IBM Corporation 2016

19

Steps for creating, and distributing keys, and certificates

This slide illustrates the steps for creating, and distributing certificates. Details are provided in subsequent slides.

Key management database

Used to store digital certificates

Required on every computer that hosts a Netcool/OMNibus component

If you run Tivoli Netcool/OMNibus in FIPS 140-2 mode, or if your network has Java based clients, use the `nc_gskcmd` utility to create the database

In FIPS 140-2 mode, passwords for key databases must meet the following requirements:

1. The minimum password length is 14 characters.
2. A password must have at least one lowercase character, one uppercase character, and one digit or special character.
3. Each character must not occur more than three times in a password.
4. No more than two consecutive characters of the password can be identical.
5. All characters are in the standard ASCII printable character set within the range from 0x20 to 0x7E inclusive.

Key management database

You must create a key database on every computer that hosts a Netcool/OMNibus component, which includes both *servers*, and *clients*. If the ObjectServer runs in FIPS 140-2 compliance mode, you must use the `nc_gskcmd` utility to create the database. When you create the database, you assign a password to protect access to the database. When FIPS 140-2 mode is used, the password must conform to specific standards, as illustrated on the slide.

Creating a key management database in FIPS 140-2 mode

To create a key database in FIPS 140-2 mode, run the following command:

```
$NCHOME/bin/nc_gskcmd -keydb -create -db "$NCHOME/etc/security/keys/omni.kdb"  
-pw password -stash -expire integer1
```

The *password* must meet the requirements described previously

The value for *integer1* is the number of days until the database expires

Maximum expiration is 7300 days, or 20 years

The *stash* parameter causes the password value to be saved as an encrypted file

```
$NCHOME/etc/security/keys/omni.sth
```

Required for Netcool/OMNIbus

Creating a key management database in FIPS 140-2 mode

This slide contains an example of the command that creates a key database.

Creating self-signed certificates

Used to set up a private trust network in which you are acting as the issuing certificate authority (CA) for your server certificates

Create a self-signed certificate in the key database of each server computer

The following example shows how to create a self-signed certificate:

```
$NCHOME/bin/nc_gskcmd -cert -create -db "$NCHOME/etc/security/keys/omni.kdb" -pw password  
-label "NCOMS_CA" -size 1024 -ca true  
-dn "CN=NCOMS_CA,O=IBM,OU=Support,L=SouthBank,ST=London,C=GB" -expire 366
```

Must specify **-ca true**

Results:

- Certificate that is called NCComs_CA
- Saved in "\$NCHOME/etc/security/keys/omni.kdb" key database
- Expires in 366 days

Creating self-signed certificates

If you want to set up a private trust network in which you act as the issuing CA for your server certificates, create a self-signed certificate in the key database of each server computer. The label parameter specifies a string that uniquely identifies the certificate within the database.

Distributing self-signed certificates

To use a self-signed certificate as a signer certificate, distribute the self-signed certificate to all *client* computers

1. Extract the self-signed certificate from the *server* computer

```
"$NCHOME/etc/security/keys/omni.kdb" -pw password -label "NCOMS"  
-target "$NCHOME/etc/security/keys/ncomscert.arm"
```

Creates a file: ncomscert.arm

2. Copy the file to each *client* computer

3. Add the certificate to the key database on each *client* computer

```
$NCHOME/bin/nc_gskcmd -cert -add -db "$NCHOME/etc/security/keys/omni.kdb"  
-pw password -label "NCOMS" -file "ncomscert.arm"
```

Distributing self-signed certificates

To use a self-signed certificate as a signer certificate, distribute the self-signed certificate to all *clients* by extracting the certificate from the server key database. Then, you add the extracted certificate to the key database on each *client* computer.

Creating certificate requests

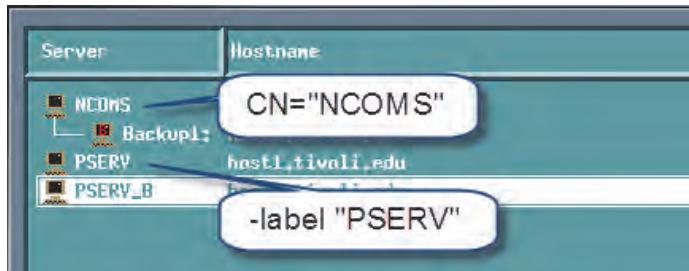
From each server computer, create a request for a digital certificate

```
$NCHOME/bin/nc_gskcmd -certreq -create -db "$NCHOME/etc/security/keys/omni.kdb"  
-pw password -label "PSERV" -size 1024  
-dn "CN=NCOMS,O=IBM,OU=Support,L=SouthBank,ST=London,C=GB"  
-file "$NCHOME/etc/security/keys/pservreq.arm"
```

ObjectServer high availability requires special attention

The **label** parameter identifies the *real* ObjectServer

The **CN** parameter identifies the *virtual* ObjectServer



24

Creating certificate requests

You must create a certificate request from each computer that hosts an ObjectServer. When ObjectServers are configured in high availability mode, the certificate request is created in a specific manner. The **label** parameter identifies the *real* ObjectServer, and the **CN** parameter identifies the *virtual* ObjectServer. **CN** stands for common name.

The command that creates the certificate request produces two forms of output. One form of output is a file. This file is sent to the certificate authority. The second form of output is an entry for the certificate request in the key database.

Signing certificate requests

External certificate authority

1. Send the certificate request to a trusted CA for authorization
2. The CA authorizes the certificate request and generates a server certificate
3. The CA returns the signed server certificate

Private trust network

To sign a certificate request file with a self-signed certificate:

```
$NCHOME/bin/nc_gskcmd -cert -sign -db "$NCHOME/etc/security/keys/omni.kdb"  
-pw password -label "NCOMS_CA" -target "$NCHOME/etc/security/keys/pservcert.arm"  
-file "$NCHOME/etc/security/certreq.arm"
```

The **label** parameter identifies the self-signed certificate

The **target** parameter identifies the certificate request file

The **file** parameter identifies the signed certificate file

© Copyright IBM Corporation 2016

25

Signing certificate requests

If you are using an external certificate authority, send the certificate request to a trusted CA for authorization. The CA authorizes the certificate request, and uses its self-signed root certificate to generate a server certificate. The CA then returns the signed server certificate.

If you are using a private trust authority, you sign the certificate request with the self-signed certificate created previously.

Receiving signed certificates

Receive the certificate into the key database for the *server*

The *server* certificate is used to authenticate the *server* side of Netcool/OMNIbus communications when a *client* requests a secure connection

The following example shows a certificate that is received for a primary ObjectServer called "PSERV"

```
$NCHOME/bin/nc_gskcmd -cert -receive -db "$NCHOME/etc/security/keys/omni.kdb"  
-pw password -file "$NCHOME/etc/security/keys/pservcert.arm"
```

The following example shows a certificate that is received for the backup ObjectServer called "BSERV"

```
$NCHOME/bin/nc_gskcmd -cert -receive -db "$NCHOME/etc/security/keys/omni.kdb"  
-pw password -file "$NCHOME/etc/security/keys/bservcert.arm"
```

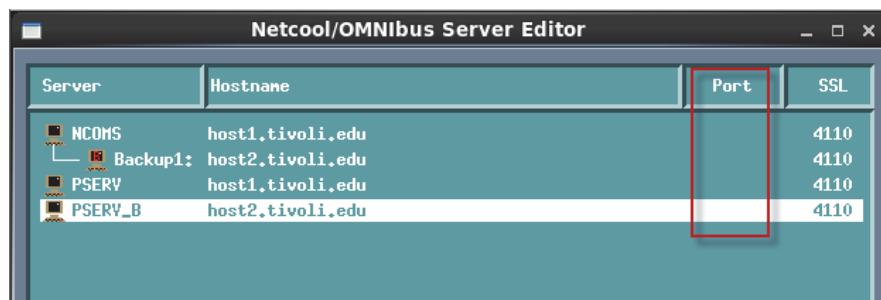
Receiving signed certificates

After the certificate request is signed, the certificate file must be added to the key database. The process is the same whether an external certificate authority created the certificate file, or it is produced internally.

When the signed certificate is added to the key database, the certificate is matched against the request entry that is stored in the key database. If the certificate matches the request, the certificate is added, and the request entry is removed. If the two entries do not match, an error is generated, and the certificate is not added to the database.

Configuring probes for SSL

Configure the local interfaces file with *only* SSL port numbers



Server	Hostname	Port	SSL
NCOMS	host1.tivoli.edu	4110	
Backup1:	host2.tivoli.edu	4110	
PSERV	host1.tivoli.edu	4110	
PSERV_B	host2.tivoli.edu	4110	

Forces probe to use the SSL ports

Configure probe properties

```
SSLServerCommonName : 'NCOMS'  
Server : 'PSERV'  
ServerBackup : 'PSERV_B'
```

© Copyright IBM Corporation 2016

27

Configuring probes for SSL

In most production environments, probes typically run on remote computers. On the computer that hosts the probe, you create an interfaces file that contains only SSL port numbers for the ObjectServer. The probe uses the ObjectServer name, and the interfaces file to determine how to communicate with the ObjectServer. If the interfaces file contains only the SSL port number, the probe must use the SSL port.

In an ObjectServer high availability configuration, the property values in the probe are important. The slide illustrates how to configure a probe to connect to a high availability pair of ObjectServers. The SSLServerCommonName contains the same text as the value for CN when the ObjectServer certificate was created. The Server parameter contains the same text as the value for label when the certificate for the primary ObjectServer was created. The ServerBackup parameter contains the same text as the value for label when the certificate for the backup ObjectServer was created.

Configuring bidirectional ObjectServer gateways for SSL

Configure the local interfaces file with *only* SSL port numbers

Netcool/OMNIbus Server Editor			
Server	Hostname	Port	SSL
NCOMS	host1.tivoli.edu	4110	
Backup1:	host2.tivoli.edu	4110	
PSERV	host1.tivoli.edu	4110	
PSERV_B	host2.tivoli.edu	4110	

Assumes that the gateway is not on the same computer with the ObjectServer

Forces gateway to use the SSL ports

Configure gateway properties

```

Gate.ObjectServerA.Server      : 'PSERV'
Gate.ObjectServerA.CommonNames : 'NCOMS'
Gate.ObjectServerB.Server      : 'PSERV_B'
Gate.ObjectServerB.CommonNames : 'NCOMS'

```

© Copyright IBM Corporation 2016

28

Configuring bidirectional ObjectServer gateways for SSL

When a bidirectional gateway is used to synchronize the ObjectServers in a high availability configuration, the gateway connects to each real ObjectServer. The slide illustrates the gateway property values that are required for this type of configuration.



Note: In most production environments, bidirectional gateways typically communicate between ObjectServers within a data center. If there is no communication outside of the data center, then SSL communication is generally not required.

Configuring JDBC gateways for SSL

Configure the local interfaces file with *only* SSL port numbers

Server	Hostname	Port	SSL
NCOMS	host1.tivoli.edu		4110
Backup1:	host2.tivoli.edu		4110
PSERV	host1.tivoli.edu		4110
PSERV_B	host2.tivoli.edu		4110

Assumes that the gateway is not on the same computer with the ObjectServer

Forces gateway to use the SSL ports

Configure gateway properties

```
Gate.RdrWtr.CommonNames : 'NCOMS'
```

Gateway connects to only the *virtual* ObjectServer

© Copyright IBM Corporation 2016

29

Configuring JDBC gateways for SSL

In a high availability configuration, the JDBC gateway connects to the *virtual* ObjectServer. The slide illustrates the gateway property value for this type of configuration.



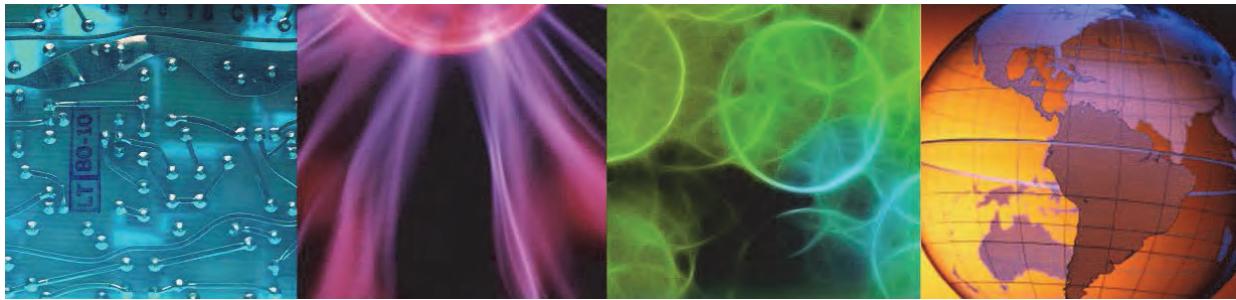
Note: In most production environments, the JDBC gateway typically communicates between an ObjectServer, and database within a data center. If there is no communication outside of the data center, then SSL communication is generally not required.



Lesson 3 Encryption and FIPS compliance



Lesson 3 Encryption and FIPS compliance



© Copyright IBM Corporation 2016

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 FIP 140-2 compliance.

ObjectServer configuration for FIPS 140-2

Create a FIPS configuration file

```
touch $NCHOME/etc/security/fips.conf
```

Modify ObjectServer property

```
PasswordEncryption AES
```

In this mode, the cryptographic functions of Tivoli Netcool/OMNIbus use cryptographic modules that are FIPS 140-2 approved

Changing the encryption type within the ObjectServer does not change stored user passwords

You must either change or reenter the current password to instantiate the new AES-based encryption string

ObjectServer configuration for FIPS 140-2

FIPS 140-2 is a US, and Canadian certification process. FIPS 140-2 validates the implementation of cryptographic algorithms. When you configure the ObjectServer for FIPS 140-2 mode, the Netcool/OMNIbus components use cryptographic algorithms that are FIPS 140-2 compliant.

Two steps are required to configure the ObjectServer to run in FIPS 140-2 mode. First, you create a file with a specific name in a specific directory. The file is empty. When the ObjectServer starts, it checks for the existence of the file. If the file exists, the ObjectServer runs in FIPS 140-2 mode.

The second step is to change the password encryption algorithm from DES to AES, which you do through a property setting. If you change the password encryption for an existing ObjectServer, any passwords that exist in the ObjectServer are not re-encrypted. You must change password values for all users to generate new encrypted strings.

Property value encryption

A major element that is used for FIPS 140-2 compliance

Requires a key file

Generate a key in a key file as follows:

```
$NCHOME/omnibus/bin/nco_keygen -o key_file [-l length | -k key]
```

-o key_file defines the name of the output file

-l length defines an optional key length: 128, 192, 256. Default is 128.

-k key defines optional value of the key in hexadecimal digits, as specified by you

If you use the *-o* command-line option to specify an output file name, and omit both the *-l* and *-k* options, a randomly generated 128-bit key is written to the file

For example

```
$NCHOME/omnibus/bin/nco_keygen -o $NCHOME/etc/security/keys/netcool.keygen
```

© Copyright IBM Corporation 2016

32

Property value encryption

FIPS140-2 deals only with the proper way by which a cryptographic module must operate, and be protected from attacks. Other security standards determine what types of data must be encrypted. The Netcool/OMNIbus solution includes components that can be used to encrypt property values in a FIPS 140-2 compliant manner.

To encrypt a property value, you must first generate a key, and save the key in a file. You create the key with the nco_keygen utility. You can optionally specify the length of the generated key. The default length is 128 bits.



Note: Property value encryption is not a requirement for FIPS 140-2 compliance. If you want encrypted property values, the Netcool/OMNIbus solution provides the mechanism to encrypt the values with a FIPS 140-2 compliant process.

Encrypting a string value

To encrypt a string value, run `nco_aes_crypt` utility as follows:

```
$OMNIHOME/bin/nco_aes_crypt -c cipher -k key_file string_value
```

`cipher` is either AES or AES_FIPS (FIPS mode)

`key_file` is the file path and name of the key

`string_value` is the string value that you want to encrypt

To encrypt the text of the password value object00, use the following command:

```
nco_aes_crypt -c AES -k $NCHOME/etc/security/keys/netcool.keygen object00
```

Result

```
@44:y7iNo3YPnE9e4ZaYc3Qrb945815sQZ7AKAd0m3yHtuY=@
```

Encrypting a string value

After the key file is created, the file is used to encrypt string values as shown on this slide. You use the `nco_aes_crypt` utility to encrypt the string.

ObjectServer properties

ObjectServer properties to configure property value encryption:

```
ConfigCryptoAlg: 'AES_FIPS'
```

```
ConfigKeyFile: '$NCHOME/etc/security/keys/netcool.keygen'
```

```
PA.Name: 'NCO_PA'
```

```
PA.Username: 'root'
```

```
PA.Password : '@44:D0Bk2i1+QzfxbzzBhHTaOvjZNz0yW4VqHbBbwgsulao=@'
```

ObjectServer properties

This slide illustrates how ObjectServer property encryption is configured.

The first two parameters configure the encryption mechanism. The first property configures the use of a FIPS 140-1 compliant algorithm. The second property identifies the key file that the ObjectServer users to decrypt property values.

The last three property values are used to define access to a process activity agent. The first property is the name of the agent, and the second property is the user name to use when communicating with the agent. The last property is the encrypted password string.

Probe properties

Property values to implement property encryption:

```
ConfigCryptoAlg: 'AES_FIPS'  
ConfigKeyFile: '$NCHOME/etc/security/keys/netcool.keygen'
```

If the target ObjectServer is running in *secure mode*, the probe requires a user name and password:

```
AuthPassword: '@44:dBt206oXIkuM9q4J262ybqruEAwE/2SRgMzzJFTBRE=@'  
AuthUserName: 'root'
```

The password string is encrypted as follows:

```
nco_aes_crypt -c AES_FIPS -k $NCHOME/etc/security/keys/netcool.keygen object00
```

Probe properties

This slide illustrates the properties that are used for encryption by a probe. The first two property values are the same as the ObjectServer.

Typically probes do not require a user name, and password to connect to an ObjectServer. If the ObjectServer is running in secure mode, the probe must provide a user name, and password. The second two properties on the slide contain the encrypted password string, and user name.

Bidirectional ObjectServer gateways

Property values to implement property encryption:

```
ConfigCryptoAlg : 'AES_FIPS'  
ConfigKeyFile : '$NCHOME/etc/security/keys/netcool.keygen'
```

Other related fields

```
Gate.ObjectServerA.Server : 'PRIMARY'  
Gate.ObjectServerA.Username : 'root'  
Gate.ObjectServerA.Password : '@44:dBt206oXIkumM9q4J262ybqruEAwE/2SRgMzzJFTBRE=@'  
Gate.ObjectServerB.Server : 'BACKUP'  
Gate.ObjectServerB.Username : 'root'  
Gate.ObjectServerB.Password : '@44:dBt206oXIkumM9q4J262ybqruEAwE/2SRgMzzJFTBRE=@'
```

The password string is encrypted as follows:

```
nco_aes_crypt -c AES_FIPS -k $NCHOME/etc/security/keys/netcool.keygen object00
```

Bidirectional ObjectServer gateways

This slide illustrates the properties that are used for encryption by a bidirectional gateway. The first two property values are the same as the ObjectServer.

Typically gateways do not require a user name, and password to connect to an ObjectServer. If the ObjectServer is running in secure mode, the gateway must provide a user name, and password for each ObjectServer. Here you see the encrypted password string for each ObjectServer.

Unidirectional ObjectServer gateways

Property values to implement property encryption:

```
ConfigCryptoAlg : 'AES_FIPS'  
ConfigKeyFile : '$NCHOME/etc/security/keys/netcool.keygen'
```

Other related fields

```
Gate.Reader.Server : 'PRIMARY'  
Gate.Reader.Username : 'root'  
Gate.Reader.Password : '@44:dBt206oXIkumM9q4J262ybqruEAwE/2SRgMzzJFTBRE=@'  
Gate.Writer.Server : 'BACKUP'  
Gate.Writer.Username : 'root'  
Gate.Writer.Password : '@44:dBt206oXIkumM9q4J262ybqruEAwE/2SRgMzzJFTBRE=@'
```

The password string is encrypted as follows:

```
nco_aes_crypt -c AES_FIPS -k $NCHOME/etc/security/keys/netcool.keygen object00
```

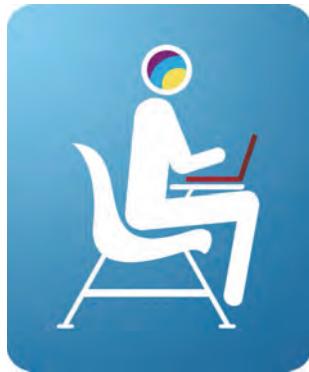
© Copyright IBM Corporation 2016

37

Unidirectional ObjectServer gateways

Unidirectional gateways are configured in a similar manner.

Student exercises



© Copyright IBM Corporation 2015

38

Student exercises

Summary

You now should be able to perform the following tasks:

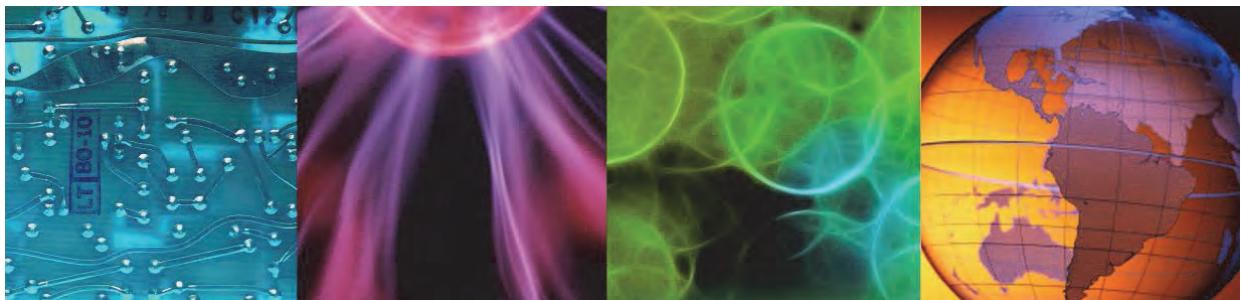
- Implement AES, and AES_FIPS property value encryption
- Configure, and use SSL communications on core components
 - Create a certificate authority
 - Create Certificate requests
 - Sign, and accept certificate requests
- Configure FIPS-140-2 security compliance on core components



10 Multitiered architecture



Multitiered architecture



© Copyright IBM Corporation 2016

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

In this unit, you learn how to create multitier architectures.

References: SC27-6264-00 *Netcool/OMNibus Version 8 Release 1 Installation and Deployment Guide*

Objectives

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

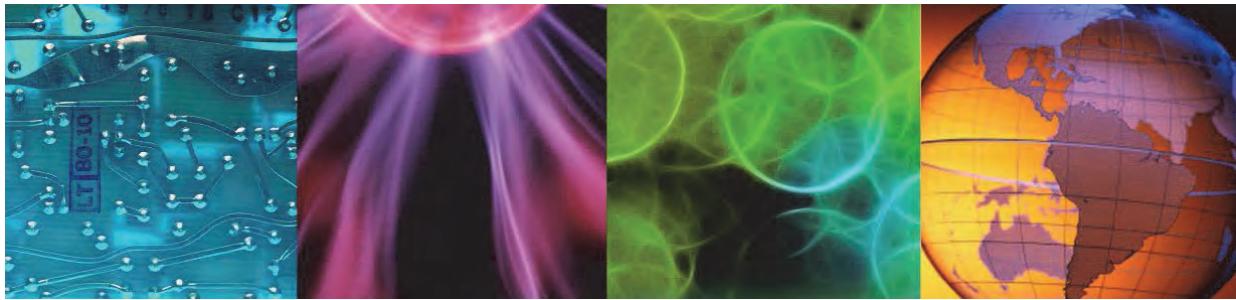
- Describe multitiered architectures and why they are used
- Implement a multitiered architecture



Lesson 1 Overview



Lesson 1 Overview



© Copyright IBM Corporation 2016

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 components of a multitier ObjectServer deployment.

Multitiered architecture

- Recommended practice to increase the scale of Netcool/OMNibus
- Implemented as layers of ObjectServers
 - The *Display layer* allows for the support of more users
 - The *Collection layer* allows for the support of higher incoming event rates — as well as having a layer where event pre-processing can be carried out
- Events pass between layers through ObjectServer gateways
- Netcool/OMNibus includes a set of baseline configurations
 - You can use these configurations to deploy a multitiered architecture
- Benefits of the included files
 - Reduces manual configuration effort, and associated potential for error
 - Files contain best practice recommended settings

© Copyright IBM Corporation 2016

4

Multitiered architecture

Since its release, Netcool/OMNibus has been pushed further, and further in terms of its event handling capacities. One of the many methods that are introduced to scale the product was the concept of multitiered Netcool/OMNibus architectures. By introducing a Display layer, IBM Netcool/OMNibus systems can support many more users. By introducing a Collection layer, IBM Netcool/OMNibus systems can support higher incoming event rates – as well as having a layer where event pre-processing can be carried out. However, by creating what was effectively a distributed database, the possibility of race conditions between ObjectServers was no trivial problem to solve.

In response to this problem, the Event Service Framework (ESF) was devised. The ESF was essentially a set of "best practice" configurations for ObjectServers and Gateways, which, when applied, would enable an engineer to rapidly deploy a multitiered IBM Netcool/OMNibus architecture. The ESF was a culmination of the best Netcool/OMNibus wisdom, and knowledge of the day and drew input from Development, Technical Support and individual that worked in the field. Since its creation, the ESF is extensively in deployments around the world.

Standard multitiered architecture

1 Collection layer

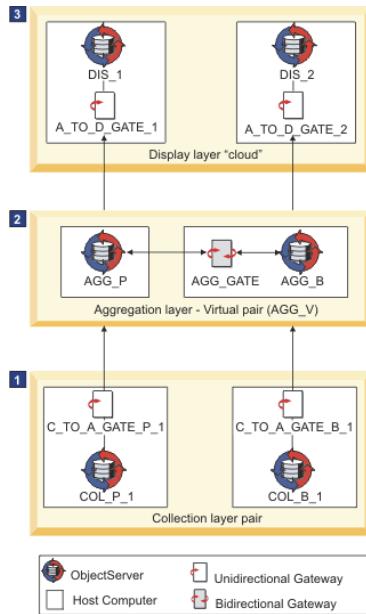
The collection layer includes a primary and backup pair of ObjectServers to which probes connect

2 Aggregation layer

The aggregation layer includes one pair of ObjectServers that are connected by a bidirectional ObjectServer gateway

3 Display layer

The display layer includes two display ObjectServers



© Copyright IBM Corporation 2016

5

Standard multitiered architecture

The components in the architecture sit within three tiers or layers: collection, aggregation, and display. Each layer shows the physical computers on which the ObjectServers, and associated ObjectServer Gateways reside.

In the figure, the unidirectional gateways transfer data in the direction of the arrow. The end of the gateway that connects to the source ObjectServer is known as the reader because it reads data from the source. The end of the gateway that connects to the destination is known as the writer because it writes data to the destination. The bidirectional gateway in the aggregation layer has a reader and a writer at each end because data flows in both directions.

Collection layer

- The collection layer includes a primary and backup pair of ObjectServers to which probes connect
 - The configuration shows one pair of collection layer ObjectServers, but further pairs can be added if required
- Each collection layer ObjectServer has its own dedicated unidirectional ObjectServer Gateway that connects the ObjectServer to the aggregation layer
 - Each collection gateway reader connects, and is fixed to, its dedicated collection ObjectServer
 - Each gateway's writer connects to the *virtual* aggregation ObjectServer pair
 - The writers can *fail over* and *fail back* between the primary and backup aggregation layer ObjectServers
 - The reader stays connected only to its dedicated collection ObjectServer

© Copyright IBM Corporation 2016

6

Collection layer

The ObjectServers at the collection layer receive events from probes. The ObjectServers reduce alarm volume through deduplication, and filtering. Events pass from the collection layer to the aggregation layer through unidirectional gateways. The gateways are configured to fail over between the high availability pair of aggregation ObjectServers.

After the events are forwarded to the aggregation layer, they are no longer needed in the collection layer ObjectServer. Triggers remove events periodically from the collection ObjectServers.

To increase the capacity to receive more events, you simply add more collection ObjectServers.

Aggregation layer

- The aggregation layer includes one pair of ObjectServers that are connected by a bidirectional ObjectServer Gateway to keep them synchronized
 - The bidirectional ObjectServer Gateway runs on the backup host
- All incoming *collection* gateway writers, and all outgoing *display* gateway readers connect to the virtual aggregation pair named AGG_V
 - The writers and readers can fail over and fail back if the primary aggregation ObjectServer computer becomes unavailable

Aggregation layer

The aggregation layer contains a single high availability pair of ObjectServers. Events from the collection ObjectServers enter the aggregation layer through unidirectional gateways. The aggregation layer is where advanced event management typically takes place. Processing at this layer includes things like event enrichment with Netcool/Impact, and integration with a ticketing system such as SmartCloud Control Desk.

If the number of event users is low enough, desktops can connect to the aggregation layer. If there are large numbers of users, you typically add a display layer to the architecture.

Display layer

- The display layer includes two display ObjectServers to which both desktop event list users and Web GUI users connect
- The configuration includes two display layer ObjectServers, but further display ObjectServers can be added if required
- Each display layer ObjectServer has its own dedicated unidirectional ObjectServer Gateway that connects the ObjectServer to the aggregation layer
 - Each display gateway reader connects to the virtual aggregation pair
 - Each gateway writer connects, and is fixed, to its dedicated display ObjectServer
 - The readers can fail over and fail back between the primary and backup aggregation layer ObjectServers
 - The writer stays connected only to its dedicated display ObjectServer
 - These gateway connections are the opposite of the gateway connections in the collection layer

© Copyright IBM Corporation 2016

8

Display layer

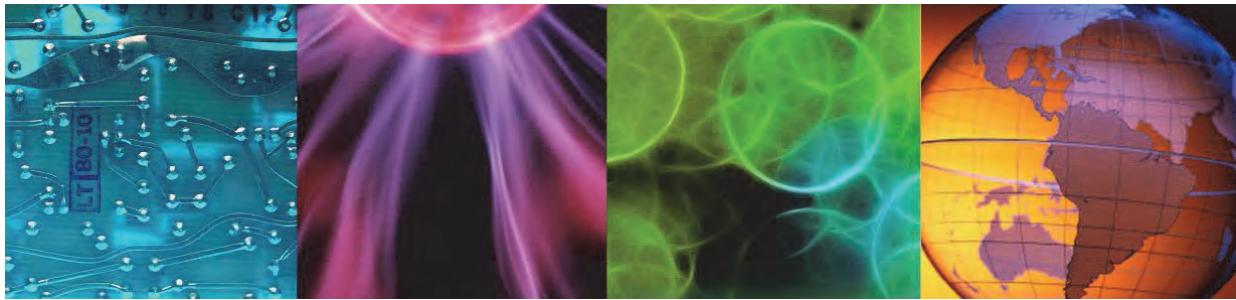
The display layer ObjectServers do nothing more than service desktop users. The desktops include both the native desktop, and Web GUI connections. Most of the processing that is required to service users is for servicing event requests. User desktops request events from the ObjectServer every 60 seconds by default. Most of the events are viewed, but never modified. So, typically, not much ObjectServer processing is required for updating events from users. The display layer ObjectServers eliminate the event request processing from the aggregation ObjectServers.

Events pass from the aggregation layer to the display layer through unidirectional gateways. When a user changes an event record, the desktop writes the event to the display layer ObjectServer, and writes the same changes to the aggregation ObjectServer. The technique is referred to as dual-write. The second write operation is how the changes to the event record get into the aggregation ObjectServer.

Lesson 2 Deploying the architecture



Lesson 2 Deploying the architecture



© Copyright IBM Corporation 2016

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 a multitier deployment.

Multitiered architecture and configuration

- Multitiered architecture uses a specific naming convention
 - Helps identify related components in each layer of the multitiered architecture
 - Helps identify the data flow within and across the layers
- Multitiered configuration files
 - All the configuration files that are provided for building the multitiered architecture are in the **\$NCHOME/omnibus/extensions/multitier** directory
 - The gateway subdirectory holds customized map definition files (**.map**), properties files (**.props**), and table replication definition files (**.tblrep.def**)
 - Gateway files are preconfigured with the required settings
 - The ObjectServer subdirectory holds customized SQL import files (**.sql**), which can update the database schemas

Multitiered architecture and configuration

Netcool/OMNIbus includes a collection of configuration files that can be used to create a multitier deployment. There are configuration files that are used to create ObjectServers for each layer and other configuration files for the gateways at each layer. If you use the supplied files, you must adhere to a component naming convention.

Naming conventions

Component	Description	Convention	Name provided	Additional name
Collection ObjectServers	Collection <i>primary</i> Objectserver <i>n</i>	COL_P_<i>n</i>	COL_P_1	COL_P_2 COL_B_2
	Collection <i>backup</i> Objectserver <i>n</i>	COL_B_<i>n</i>	COL_B_1	COL_P_3 COL_B_3
Collection-to-Aggregation Gateways	Collection <i>primary</i> Objectserver Gateway <i>n</i>	C_TO_A_GATE_P_<i>n</i>	C_TO_A_GATE_P_1	C_TO_A_GATE_P_2 C_TO_A_GATE_B_2
	Collection <i>backup</i> Objectserver Gateway <i>n</i>	C_TO_A_GATE_B_<i>n</i>	C_TO_A_GATE_B_1	C_TO_A_GATE_P_3 C_TO_A_GATE_B_3
Aggregation ObjectServers	Aggregation <i>primary</i> ObjectServer	AGG_P	AGG_P	Not applicable
	Aggregation <i>backup</i> ObjectServer	AGG_B	AGG_B	Not applicable
	Aggregation <i>virtual</i> pair	AGG_V	AGG_V	Not applicable
Aggregation Gateway	Aggregation Gateway	AGG_GATE	AGG_GATE	Not applicable
Aggregation-to-Display Gateways	Display Objectserver Gateway <i>n</i>	A_TO_D_GATE_<i>n</i>	A_TO_D_GATE_1	A_TO_D_GATE_3 A_TO_D_GATE_4
			A_TO_D_GATE_2	...
Display ObjectServers	Display ObjectServer <i>n</i>	DIS_<i>n</i>	DIS_1	DIS_3 DIS_4
			DIS_2	...

© Copyright IBM Corporation 2016

11

Naming conventions

The slide shows the component names for each layer. The components include ObjectServers, and gateways. The names help to identify the role of the component within the architecture. However, the real need for adhering to the names is that the names appear within the configuration files. For example, the property file for the synchronizer gateway at the aggregation layer contains the names for the primary, and backup ObjectServers: AGG_P, and AGG_B.

You can use your own ObjectServer names. However, you must modify the configuration files, and change the appropriate entries.

Multitiered configuration file locations

- All the configuration files that are provided for building the multitiered architecture are in the **\$NHOME/omnibus/extensions/multitier** directory
- The gateway subdirectory holds the following customized files:
 - **A_TO_D_GATE.map**
 - **A_TO_D_GATE.tblrep.def**
 - **A_TO_D_GATE_1.props**
 - **A_TO_D_GATE_2.props**
 - **AGG_GATE.map**
 - **AGG_GATE.props**
 - **AGG_GATE.tblrep.def**
 - **C_TO_A_GATE.map**
 - **C_TO_A_GATE_B_1.props**
 - **C_TO_A_GATE_B_1.tblrep.def**
 - **C_TO_A_GATE_P_1.props**
 - **C_TO_A_GATE_P_1.tblrep.def**

© Copyright IBM Corporation 2016

12

Multitiered configuration file locations

The gateway files are preconfigured with the required settings, and should be used as provided. The gateway files also require compliance with the naming conventions for ObjectServers and ObjectServer Gateways in the multitiered architecture.

Multitiered configuration file locations, continued

- The ObjectServer subdirectory holds the following customized SQL import files:
 - aggregation.sql
 - aggregation_rollback.sql
 - collection.sql
 - collection_rollback.sql
 - display.sql
 - display_rollback.sql
- The SQL files provide the following options:
 - Add relevant columns to the database tables
 - Create the automations that control the behavior of the ObjectServers, ObjectServer Gateways, and clients, and the flow of events across these components
 - Enable and disable triggers for the ObjectServers
 - Assign permissions to the triggers
 - Create conversions
 - Roll back the applied schema changes, if required

Multitiered configuration file locations, continued

Netcool/OMNIbus also includes a collection of ObjectServer SQL files. There is an SQL file for each type of ObjectServer. The SQL file is used when the ObjectServer is created with the nco_dbinit command. The SQL file contains changes that are applied to a default ObjectServer when the ObjectServer is created. The changes represent the best practice configurations for each type of ObjectServer.

Steps to deploy the architecture manually

1. Set up the interfaces file
2. Install the primary aggregation ObjectServer
3. Install the backup aggregation ObjectServer
4. Configure the bidirectional aggregation ObjectServer Gateway
5. Install the primary collection ObjectServer
6. Configure the unidirectional primary collection ObjectServer Gateway
7. Install the backup collection ObjectServer
8. Configure the unidirectional backup Collection ObjectServer Gateway
9. Install the display ObjectServer 1
10. Configure the unidirectional display ObjectServer 1 Gateway
11. Install the display ObjectServer 2
12. Configure the unidirectional display ObjectServer 2 Gateway

© Copyright IBM Corporation 2016

14

Steps to deploy the architecture manually

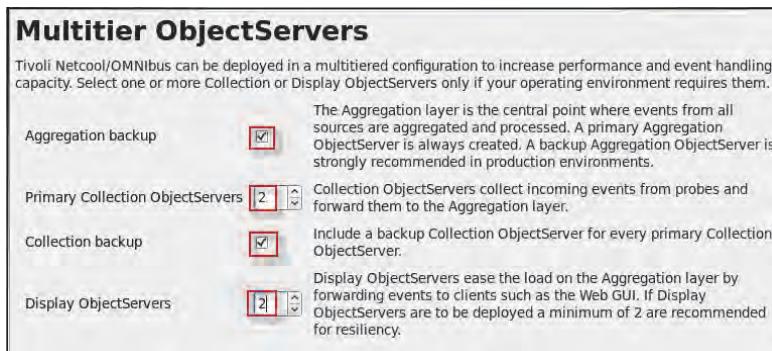
The supplied configuration files provide a significant benefit in terms of facilitating deployment of the multitier architecture. However, there is still a considerable amount of manual effort that is required to create the configuration. For example, you must create the interfaces file, which involves a manual edit process. You need to run the nco_dbinit command to create each ObjectServer. You must copy the supplied gateway configuration files to the \$OMNIHOME/etc directory. You must create the process activity configuration file for each computer in the architecture.

Steps to deploy the architecture automatically

- Run Initial Configuration Wizard utility

Define complete architecture

- Deploy architecture on each server



Steps to deploy the architecture automatically

The Initial Configuration Wizard can create the entire multitiered deployment with the least amount of effort. The following steps are required.

- You run the utility on one computer, and define the complete multitiered architecture. You define the following items:
 - Each computer by name
 - All ObjectServers and whether you want high availability
 - A process agent name for each computer
 The utility creates a configuration file.
- You copy the file to each computer in the architecture.
- You run the Initial Configuration Wizard on each computer, and apply the configuration. The utility performs the following steps:
 - Creates an interfaces file that contains entries for all components in the architecture
 - Creates the ObjectServer for the respective computer
 - Copies the gateway configuration files to the correct directory
 - Creates a process activity configuration file to manage the components on the respective computer
- Start the local process agent, and the agent starts the required components
- Repeat the previous steps on each computer in the deployment

Student exercises



© Copyright IBM Corporation 2016

16

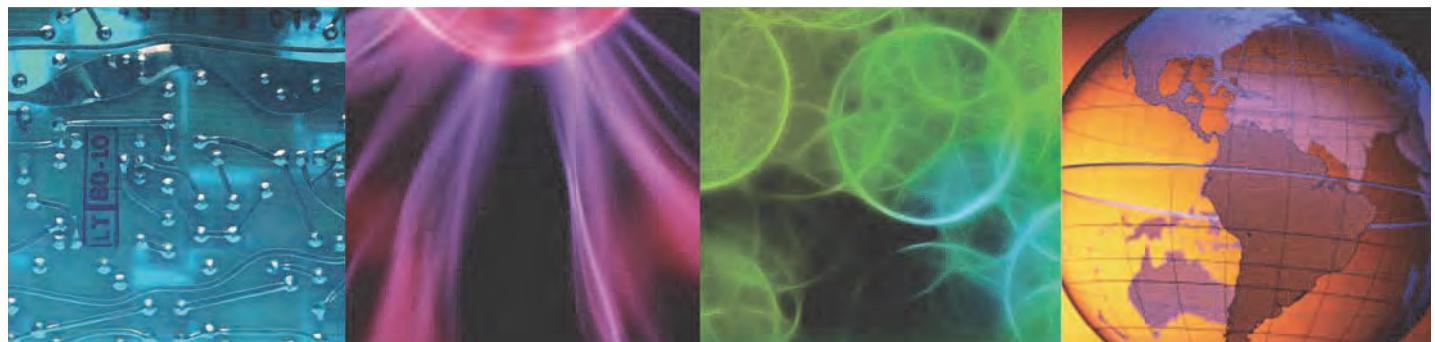
Student exercises

Summary

You now should be able to perform the following tasks:

- Describe multitiered architectures, and why they are used
- Implement a multitiered architecture

TN035 2.0



ibm.com/training

Authorized
IBM | Training