

Course Guide

IBM Tivoli Netcool/Impact 7.1 Administration and Implementation

Course code TN045 ERC 2.1



February 2017 edition

NOTICES

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

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

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

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

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.

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.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

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

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.

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.

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

© Copyright International Business Machines Corporation 2017.

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

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

Contents

About this course	xiii
Course objectivesxv
Audience	xvi
Prerequisites	xvi
Agenda	xvii
Course agenda	xvii
Unit 1 Introduction to IBM Tivoli Netcool/Impact	1-1
Objectives	1-2
IBM Tivoli Netcool/Impact overview	1-3
Network Operations Insight	1-3
IBM Tivoli Netcool/Impact architecture	1-4
Software compatibility	1-5
Select a specific product	1-6
Compatibility report for Netcool/Impact V7.1	1-7
IBM Tivoli Netcool/Impact events	1-8
IBM Netcool/Impact event sources	1-9
IBM Tivoli Netcool/OMNibus	1-10
Netcool/OMNibus probes	1-11
Tivoli Netcool/OMNibus architecture overview	1-12
Netcool/Impact event readers and listeners	1-14
Netcool/Impact policy engine	1-15
Netcool/Impact policies	1-16
Netcool/Impact data source adapters (DSAs)	1-17
IBM Netcool/Impact operator views	1-18
Netcool/Impact V7.1 embedded products	1-19
IBM Tivoli Netcool/Impact concepts	1-20
Event suppression	1-21
Event correlation	1-22
Event visualization	1-23
Event aggregation	1-24
Event enrichment	1-25
The event enrichment life cycle	1-26
Student exercises	1-27
Review questions	1-28
Review answers	1-29
Summary	1-30
Unit 2 The Netcool/Impact user interface	2-1
Objectives	2-2
Overview of the Netcool/Impact user interface	2-3

IBM Tivoli Netcool/Impact Welcome screen	2-4
The Tivoli Netcool/Impact view	2-5
The Project view versus the Global view	2-6
Benefits of using projects	2-7
The project list	2-8
Prebuilt projects	2-9
Project elements	2-10
Creating a project	2-11
The global repository	2-12
Global repository elements	2-13
Version control	2-14
Version control interface	2-15
Version control script	2-16
Student exercises	2-17
Review questions	2-18
Review answers	2-19
Summary	2-20

Unit 3 The Netcool/Impact data model 3-1

Objectives	3-2
Netcool/Impact solutions	3-3
Data models	3-4
Data model architecture	3-5
Data sources	3-6
SQL database data sources	3-7
Lightweight directory access protocol (LDAP) data sources	3-8
Mediator data sources	3-9
The internal data repository	3-10
Available data sources	3-11
Creating a data source	3-12
Data types	3-13
The purpose of data types	3-14
SQL and LDAP data types	3-15
Mediator data types	3-16
Internal data types	3-17
Creating an SQL data type	3-18
SQL data type configuration	3-19
Data items	3-20
New Field window	3-21
Links	3-22
Database Schema example	3-23
Types of links	3-24
Static links	3-25
Creating a static link	3-26
Dynamic links	3-27
Creating a Dynamic Link Example: Link by Filter	3-28
Link by Filter window	3-29
Composite links: Virtual data types	3-30

Adding the composite field	3-31
Viewing a data type with composite fields	3-32
Event sources	3-33
Types of event sources	3-34
Event source architecture	3-35
Student exercises	3-36
Review questions	3-37
Review answers	3-38
Summary	3-39
Unit 4 Policies	4-1
Objectives	4-2
Lesson 1 Netcool/Impact policy structure	4-3
Policies overview	4-4
Netcool/Impact policy language and JavaScript	4-5
IPL basic constructs	4-6
IPL data types	4-7
Simple data types	4-8
Simple data type example policy	4-9
Complex data types	4-10
Contexts	4-11
Context example	4-12
Arrays	4-13
Array example	4-14
Data items	4-15
Data item example	4-16
Event containers	4-17
Event container example	4-18
Variables	4-19
Built-in variables	4-20
The EventContainer variable	4-21
Dot notation example	4-22
Event state variables	4-23
The DataItems variable	4-24
The DataItem variable	4-25
The Num variable	4-26
User-defined variables	4-27
User-defined variables example	4-28
Control structures	4-29
If statement logic	4-30
While statement logic	4-31
Lesson 2 Policy functions	4-32
Policy function overview	4-33
Action functions	4-34
Action function example	4-35
Parser functions	4-36
Parser function examples	4-37
The Length function	4-38

Length function example	4-39
The Int function	4-40
Int function example	4-41
The Eval function	4-42
The Extract function	4-43
The UpdateEventQueue function	4-44
UpdateEventQueue example	4-45
Function libraries	4-46
Configuring function libraries	4-47
Calling function libraries	4-48
Lesson 3 Error handling and other policy topics	4-49
Exceptions	4-50
Handling exceptions	4-51
Sample exception handler IPL	4-52
Sample exception handler JavaScript	4-53
The DefaultExceptionHandler	4-54
The failed event exception handler	4-55
Local transactions	4-56
Local transactions considerations	4-57
Clear cache	4-58
Continuation character	4-59
Student exercises	4-60
Review questions	4-61
Review answers	4-62
Summary	4-63
Unit 5 Services.....	5-1
Objectives	5-2
Services	5-3
Types of services	5-4
Services: Default	5-5
Services: Global	5-6
Netcool/Impact predefined services	5-7
Netcool/Impact internal and user-defined services	5-8
The OMNIbus event reader	5-9
Configuring the OMNIbusEventReader	5-10
General Settings: Restrict Fields	5-11
General Settings: Optimize List	5-12
Event Mapping: Event Matching	5-13
Event Mapping: Actions	5-14
Event Mapping: Event Locking	5-15
Event Mapping: New Mapping	5-16
Analyze Filter	5-17
Filter analysis results	5-18
Running the OMNIbusEventReader	5-19
Default logging	5-20
The policy logger	5-21
Log files	5-23

Expanding logging capabilities	5-24
Other services and optional service selections	5-25
Database event reader	5-26
Database event reader General Settings	5-27
Database event reader Event Mapping	5-28
OMNIbus event listeners	5-29
OMNIbusEventListener configuration	5-30
Service startup and service logging	5-31
Review questions	5-32
Review answers	5-33
Summary	5-34
Unit 6 The Enrichment policy.....	6-1
Objectives	6-2
Netcool/Impact solution types	6-3
Event enrichment solutions	6-4
Policy creation	6-5
The Policies tab	6-6
Importing policies	6-7
Policy uploader	6-8
The File Upload window	6-9
Updated policy upload window	6-10
Policies tab after the upload	6-11
The policy editor	6-12
The Insert Function list	6-13
Configuring functions	6-14
Policy editor toolbar functions	6-15
The event enrichment policy	6-17
The event enrichment policy logical flow	6-18
Launching policies for testing	6-19
Testing from the Policies tab	6-20
Policy triggering	6-21
The OMNIbusEventReaderService	6-22
Testing the policy	6-23
Expected policy results	6-24
Policy debugging	6-25
Error handling	6-26
Review questions	6-27
Review answers	6-28
Student exercises	6-29
Summary	6-30
Unit 7 Controlling policy execution sequence	7-1
Objectives	7-2
Threading	7-3
Policy execution	7-4
Policy chaining	7-5
Policy chaining eliminates inefficiencies	7-6

Configuring policy chaining	7-7
Configuring threading	7-8
Controlling threading processing	7-9
Configuring event locking	7-10
Synchronized blocks	7-11
Synchronized blocks syntax	7-12
Sample policies without synchronization	7-13
Sample policy with synchronization	7-14
Student exercises	7-15
Review questions	7-16
Review answers	7-17
Summary	7-18
Unit 8 Policy wizards.....	8-1
Objectives	8-2
The purpose of wizards	8-3
Types of wizards	8-4
Wizard descriptions	8-5
Wizard example: Event enrichment	8-7
Connect to an event reader	8-8
Connect to a data source	8-9
Identity information	8-10
Data source and event mappings	8-11
Policy and Data Summary	8-12
Viewing the policy	8-13
Starting the OMNIbusEventReader (TESTENRICH)	8-14
Wizard considerations	8-15
Student exercises	8-16
Review questions	8-17
Review answers	8-18
Summary	8-19
Unit 9 Notification policies	9-1
Objectives	9-2
Lesson 1 Notification policies	9-3
Types of notifications	9-4
Email service: EmailSender service	9-5
Configuring the EmailSender service	9-6
Using the SendEmail action function	9-7
Receiving email	9-9
DefaultEmailReader service configuration	9-10
Example policy to process email	9-11
Sending instant messages	9-12
Configuring JabberService	9-13
Using the SendInstantMessage action function	9-14
Receiving instant messages	9-15
DefaultJabberReader configuration	9-16
Example of a policy to process instant messaging	9-17

Student exercises	9-18
Review questions	9-19
Review answers	9-20
Summary	9-21
Unit 10 Reports	10-1
Objectives	10-2
Lesson 1 Introduction to reports	10-3
Types of reports	10-4
Available reports	10-5
Creating a report	10-6
Report example: The Action Efficiency report	10-7
Lesson 2 Data type reporting	10-8
Overview of data type reporting	10-9
Cache settings	10-10
Configuring cache settings	10-11
View performance Report	10-12
Lesson 3 Service Level Objective reporting	10-13
Service Level Objective reporting	10-14
Lesson 4 The Configuration Documenter	10-16
Introduction to the Configuration Documenter	10-17
Opening the Configuration Documenter	10-18
Configuration Documenter links	10-19
The Configuration Documenter application	10-20
Student exercises	10-21
Review questions	10-22
Review answers	10-23
Summary	10-24
Unit 11 Operator views	11-1
Objectives	11-2
Lesson 1 Introduction to operator views	11-3
Operator view benefits	11-4
Types of operator views	11-5
Configuring basic operator views	11-6
Action Panel settings	11-7
Information group settings	11-8
The basic operator view policy	11-9
The basic operator view HTML	11-10
Running the operator view	11-11
Sample operator view	11-12
Display event context	11-13
Display action buttons	11-14
Display data groups	11-15
Modifying operator views beyond the GUI	11-16
Accessing operator view source	11-17
Smart tags	11-18
The OrgNodes tag	11-19

Student exercises	11-20
Review questions	11-21
Review answers	11-22
Summary	11-23
Unit 12 Working with web services	12-1
Objectives	12-2
Lesson 1 Web services	12-3
The Web Services DSA	12-4
Using the Web Services DSA: Prerequisite tasks	12-5
The Web Services wizard	12-6
Web service example	12-7
Web services description language (WSDL)	12-8
WSDL service information	12-9
Creating a web service policy with the wizard	12-10
Introduction window	12-11
WSDL File and Client Stub window	12-12
Redeploying the impact application EAR	12-13
Web service method parameters window	12-14
Web Service Method Parameters window (continued)	12-15
Web Service End Point window	12-16
Summary and Finish window	12-17
Web service policy generated	12-18
Instructor demonstration	12-19
Student exercises	12-20
Review questions	12-21
Review answers	12-22
Summary	12-23
Unit 13 Hibernation, X in Y, and synthetic events	13-1
Objectives	13-2
Hibernations	13-3
Hibernation data types	13-4
Action keys and the hibernation timeout value	13-5
Hibernation functions (continued)	13-7
Retrieving hibernations	13-8
Waking a hibernation	13-9
The HibernatingPolicyActivator	13-10
Setting a hibernation	13-11
Activating the hibernation	13-12
Viewing hibernations	13-13
X in Y policies: X events in Y seconds	13-14
The X in Y wizard	13-15
X in Y wizard introduction window	13-16
X in Y wizard Connect to Event Reader window	13-17
X in Y wizard Set Thresholds window	13-18
X in Y wizard Summary and Finish window	13-19
X in Y program flow	13-20

Synthetic events	13-21
The NewEvent function	13-22
The NewObject function	13-23
Student exercises	13-24
Review questions	13-25
Review answers	13-26
Summary	13-27
Unit 14 Maintenance window management	14-1
Objectives	14-2
Maintenance Window Management	14-3
Maintenance Window Management (MWM) configuration	14-4
MWM properties	14-5
The MWMProperties policy getProperties function	14-6
MWM policy	14-7
The MWMActivator Service	14-8
Creating a maintenance window	14-9
Viewing the maintenance window	14-10
Starting The Netcool/OMNibus event viewer	14-11
Netcool/OMNibus view builder	14-12
Viewing maintenance events in Netcool/OMNibus	14-13
Student exercises	14-14
Review questions	14-15
Review answers	14-16
Summary	14-17
Unit 15 Command-line tools and self-monitoring	15-1
Objectives	15-2
Lesson 1 Command-line tools	15-3
String and policy tools	15-4
Export and import tools	15-5
Cluster status tool	15-6
Command-line manager	15-7
Starting the command-line interface	15-8
Command-line manager supported services	15-9
Command-line manager examples	15-10
JavaScript in the command-line manager	15-11
Lesson 2 Self-monitoring	15-12
Self-monitoring overview	15-13
Self-monitoring configuration using the GUI	15-14
Self-monitoring configuration using the command-line manager	15-15
Self-monitoring events	15-16
Student exercises	15-17
Review questions	15-18
Review answers	15-19
Summary	15-20

Unit 16 The Netcool/Impact UI data provider	16-1
Objectives	16-2
Lesson 1 The UI data provider	16-3
The UI Data Model components	16-4
Jazz for Service Management	16-5
Installing Jazz for Service Management	16-6
The Jazz for Service Management installer	16-7
Starting the Jazz for Service Management console	16-8
General steps to create a UI provider	16-9
Example: UI data provider data type	16-10
Example: UI data provider policy user parameters	16-11
Example: Configuring communication in WebSphere	16-12
Example: Connection in the Jazz console	16-13
Example: Console settings	16-14
Example: Creating a new page	16-15
Example: Adding and configuring the List widget	16-16
Student exercises	16-17
Review questions	16-18
Review answers	16-19
Summary	16-20
Unit 17 Server utilities	17-1
Objectives	17-2
Clustering architecture	17-3
Name server initial registration	17-4
Primary failure scenario	17-5
High-availability solution	17-6
nci_export command	17-7
nci_import command	17-8
Creating a second server on a separate host	17-9
Installing only the Impact server	17-10
Creating a new server on a single host	17-11
Starting the primary and secondary servers	17-12
Determining the primary Netcool/Impact server	17-13
Student exercises	17-14
Review questions	17-15
Review answers	17-16
Summary	17-17

About this course

IBM Training

IBM



IBM Tivoli Netcool/Impact 7.1 Administration and Implementation

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Netcool/Impact is a highly scalable analytics engine that adds event and service enrichment as well as business impact analysis for event data. Netcool/Impact 7.1 delivers features that enhance the visualization, usability, and integration functions. The visualization enhancements are facilitated by the integration of the IBM Dashboard Application Services Hub, which is part of Jazz™ for Service Management, and new Netcool/Impact features such as the UI data provider. This class teaches students how to deploy and administer Netcool/Impact through practice exercises.

The lab environment for this course uses the Linux platform.

Details	
Delivery method	Classroom or instructor-led on-line (ILO)
Course level	ERC 2.0 This course is an update of TN045:IBM Tivoli Netcool/Impact 7.1 Administration and Implementation ERC 1.0 ERC 2.1 Unit 14 was replaced
Product and version	IBM Tivoli Netcool/Impact 7.1.0.5
Recommended duration	4days
Skill level	Basic / Intermediate

Course objectives

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

- Install and configure Netcool/Impact
- Write policies using Netcool/Impact policy language (IPL)
- Create, update, and edit policies in the console
- Define Netcool/Impact data sources, data items, and services
- Use logs to verify policy function
- Use and deploy policies using the wizards
- Create a second server instance and import Netcool/Impact server components
- Create a Jazz for Service Management dashboard using Netcool/Impact data

Audience

This course is designed for anyone interested in gaining hands-on experience working with Netcool/Impact.

Prerequisites

Before taking this course, make sure that you have basic Linux skills, and some familiarity with Netcool/OMNIbus.

Agenda

- Unit 1 Introduction to IBM Tivoli Netcool/Impact
- Unit 2 The Netcool/Impact user interface
- Unit 3 The Netcool/Impact data model
- Unit 4 Policies
- Unit 5 Services
- Unit 6 The Enrichment policy
- Unit 7 Controlling policy execution sequence
- Unit 8 Policy wizards
- Unit 9 Notification policies
- Unit 10 Reports
- Unit 11 Operator views
- Unit 12 Working with web services
- Unit 13 Hibernation, X in Y, and synthetic events
- Unit 14 Maintenance window management
- Unit 15 Command-line tools and self-monitoring
- Unit 16 The Netcool/Impact UI data provider
- Unit 17 Server utilities

IBM Tivoli Netcool/Impact 7.1 Administration and Implementation

© Copyright IBM Corporation 2017

Agenda

Course agenda

The course contains the following units:

1. Introduction to IBM Tivoli Netcool/Impact

This unit introduces the IBM Tivoli Netcool/Impact product and describes the basics of the application architecture.

This unit's exercises focus on Netcool/Impact installation and the initialization of the database to be used for class exercises.

2. The Netcool/Impact user interface

This unit describes how to manage and create Netcool/Impact objects within the user interface.

This unit focuses on starting components, stopping components, and using projects.

3. The Netcool/Impact data model

The first step in developing a Netcool/Impact solution is to determine the data that will be used. This unit covers how to access data inside the Netcool/Impact structure, and how to build data sources and data types.

The focus of this unit is the Netcool/Impact Data Model.

4. Policies

This unit describes the purpose of a policy and how policies work. It also covers the basics of creating a policy in Netcool/Impact.

This unit's exercises are an introduction to writing Netcool/Impact policies.

5. Services

This unit focuses on controlling the operation of policies within Netcool/Impact. These topics are important in larger, more complex environments.

6. The Enrichment policy

One of the primary functions of Netcool/Impact is to provide event enrichment. This unit examines the event enrichment policy.

7. Controlling policy execution sequence

This unit focuses on controlling the operation of policies within Netcool/Impact. These topics are important in larger, more complex environments.

This unit focuses on event locking and policy chaining.

8. Policy wizards

This unit introduces Netcool/Impact wizards.

In this unit, you work with the policy wizard.

9. Notification policies

This unit describes how Netcool/Impact uses notifications, including sending and receiving email and instant messaging.

In this unit, you configure a notification policy.

10. Reports

This unit introduces the Netcool/Impact reporting features.

In this unit, you set up report collection, run several reports, and display the configuration documenter.

11. Operator views

This unit describes how to customize and use the operator view.

This exercise is an elementary introduction to the operator view. An operator view is created with an information panel and an action button, which runs a policy that clears all events in the event list.

12. Working with web services

This unit describes how to configure web services using Netcool/Impact.

In these exercises, you work with the Web Services wizard.

13. Hibernation, X in Y, and synthetic events

This unit describes how to customize and use Netcool/Impact advanced features.

This exercise is an introduction to advanced topics such as Hibernation, X in Y processing, and Synthetic Events.

14. Maintenance window management

This unit describes how Netcool/Impact uses Maintenance Window Management (MWM) to manage events.

In this exercise, you experiment with the Maintenance Window Management service.

15. Command-line tools and self-monitoring

This unit describes how Netcool/Impact uses command-line tools and self-monitoring to manage the Netcool/Impact server.

In this unit you experiment with policy encryption, and self-monitoring.

16. The Netcool/Impact UI data provider

This unit describes how to configure Netcool/Impact to use Jazz for Service Management and the UI Data Provider.

In this unit, a Netcool/Impact policy passes data to the IBM Service Management Hub to populate dashboard widgets.

17. Server utilities

In these exercises, you work with server utilities.

Unit 1 Introduction to IBM Tivoli Netcool/Impact

IBM Training



Introduction to IBM Tivoli Netcool/Impact

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit introduces the IBM Tivoli Netcool/Impact product and describes the basics of the application architecture.

Objectives

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

- Describe the Tivoli Netcool/Impact product and its components
- Define the role of Netcool/Impact in event management
- Describe the Netcool/Impact enrichment life cycle

IBM Tivoli Netcool/Impact overview

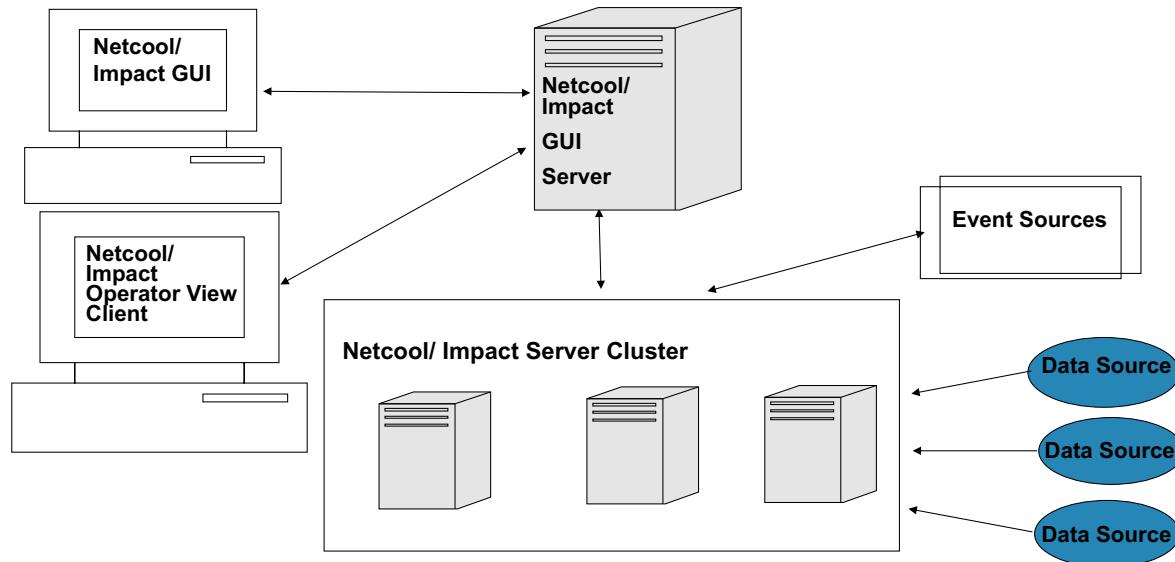
Netcool/Impact has the following characteristics:

- One of a suite of products that Netcool provides for network management
- A set of tools designed to capture and react to events
- Installed as a primary component of Network Operations Insight (NOI)

Network Operations Insight

- Network Operations Insight includes new releases of IBM Tivoli Netcool/OMNibus and IBM Tivoli Netcool/Impact that are extended with the network management capabilities of IBM Tivoli Netcool Network Manager and IBM Tivoli Configuration Manager.
- Key features:
 - Provides deeper operational insight across the IT and network
 - Brings operational visibility to the business providing dynamic incident and service information to key stakeholders through IBM Connections
 - Extends event reduction and correlation capabilities with network discovery and topology based root-cause analysis
 - Enables faster problem identification, reduced time to resolution, and enhanced service levels, resulting in improved productivity and lower operational costs

IBM Tivoli Netcool/Impact architecture



Introduction to IBM Tivoli Netcool/Impact

© Copyright IBM Corporation 2017

IBM Tivoli Netcool/Impact architecture

The Tivoli Netcool/Impact software product consists of two major components and a few other minor components. Central to the Netcool/Impact product is the Netcool/Impact Server. The Netcool GUI Server is just as important because it provides the web-based user interface for the Netcool/Impact Server.

Netcool/Impact supports two types of installations: single-system installations and distributed installations.

- A *single-system* installation consists of Netcool/Impact and related components that are installed on a single system in your environment.
- A *distributed* installation consists of Netcool/Impact and related components that are installed across multiple systems.

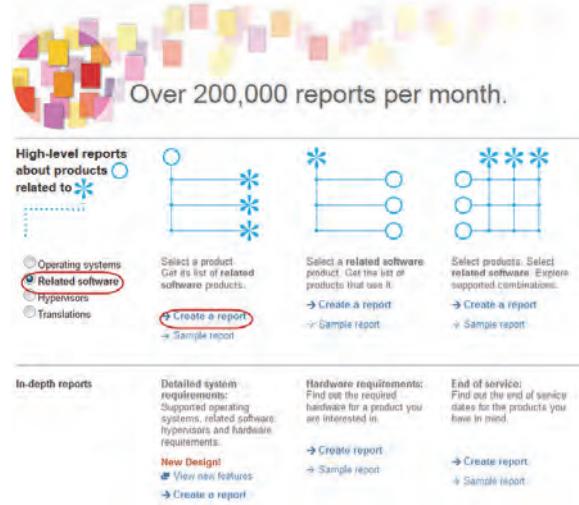
For the purposes of discussion and exercises in this class, the focus is on a single-system installation. A distributed installation might be more appropriate for a production implementation.

With *server clustering*, two or more instances of the Netcool/Impact server are installed on separate systems and configured as part of a server cluster. Clustering provides failover and load-balancing for the server instances.

Software compatibility

For IBM Software Product Compatibility Reports, see the following website:

<http://www-969.ibm.com/software/reports/compatibility/clarity/index.html>



Introduction to IBM Tivoli Netcool/Impact

© Copyright IBM Corporation 2017

Software compatibility

The link opens a portal that allows you to create product compatibility reports based on search criteria.

Select a specific product

The screenshot shows a search interface for related software products. The search term 'Netcool/Impact' has been entered into the 'Full or partial product name:' field. The search results list 'Netcool/Impact' and 'Tivoli Netcool/Impact'. Below the search results, there is a dropdown menu for 'Version:' set to '7.1' and a checked checkbox for 'Show fix packs'. A note at the bottom states: 'Note: The support shown in the generated report may require a particular maintenance level for the products.'

Introduction to IBM Tivoli Netcool/Impact

© Copyright IBM Corporation 2017

Select a specific product

Select a software product at a specific release level.

Compatibility report for Netcool/Impact V7.1

Tivoli Netcool/Impact 7.1

Detailed System Requirements

Report filters

Available Reports

7.1.0.1 maintenance level

Utilities

Regenerate Anytime

Download PDF

Print

Provide feedback

Notes

Data as of 2015-03-16
02:35:43 EDT

Disclaimer

New Design!

View new features

Operating Systems

Hypervisors

Prerequisites

Supported Software

Hardware

Packaging List

AIX

Linux

Solaris

Windows

AIX

Operating System	Operating System Minimum	Hardware	Bitness	Product Minimum	Components	Notes	Details
AIX 6.1	TLS	POWER System - Big Endian	64-Bit	7.1.0.1			View
AIX 7.1	Base	POWER System - Big Endian	64-Bit	7.1.0.1			View

AIX

Linux

Solaris

Windows

[Back to top](#)

Linux

Operating System	Operating System Minimum	Hardware	Bitness	Product Minimum	Components	Notes	Details
------------------	--------------------------	----------	---------	-----------------	------------	-------	---------

Introduction to IBM Tivoli Netcool/Impact

© Copyright IBM Corporation 2017

Compatibility report for Netcool/Impact V7.1

Specifics for AIX and Linux compatibility are shown. Hardware and other requirements are also available.

IBM Tivoli Netcool/Impact events

- An **event** is any change occurring within an information technology infrastructure that can be captured electronically
 - Translates to a status change of a resource
- Customers use Netcool/Impact to process high-volume event streams and to do the following tasks:
 - Gather additional information about an event or series of events (*aggregation*)
 - Alert staff about a high-priority conditions (*escalation*)
 - Decide which events should be ignored (*suppression*)
 - Set markers in diverse data sources (*correlation*)
 - Take action on devices (*autocorrection*)

Netcool/Impact event management transforms IT infrastructure operational data into usable information. Using that information network administrators can more easily manage and extract value from complex IT environments.

IBM Netcool/Impact event sources

- An **event source** (or **data source**) is the place in the network where the event is tracked or stored
- There are many sources of events:
 - Netcool/Impact is most often used to capture Netcool/OMNIbus events
 - Other sources of events include these examples:
 - RDBMS
 - Web Services
 - Email
 - Instant messaging
 - IBM Tivoli Network Manager IP
 - Command Line Policy execution

IBM Netcool/Impact event sources

Event sources (or **data sources**) are elements of the data model that represent sources of data in your environment. Netcool/Impact supports SQL databases, LDAP servers, and various other external data sources.

You can also use the Netcool/Impact internal data repository as a data source.

IBM Tivoli Netcool/OMNIbus

- Netcool/OMNIbus is the core alarm management database for any Tivoli Network Management solution
- A central data repository for network alarms
 - *Alarm* is the generic term for network faults, system faults, security faults, and performance indicators. Sometimes the term *alarm* is used interchangeably with *event*.
- The manager of managers
 - Netcool/OMNIbus consolidates information from other management applications, system management systems, and element management systems
 - Other management applications include voice, data, Internet Protocol (IP), wireless transmission, firewalls, and servers

Tivoli Netcool/OMNIbus provides powerful, sophisticated management of network events. It tracks alert information in a high-performance, in-memory database, and presents information of interest to specific users through individually configurable filters and views. Netcool/OMNIbus automation functions can perform intelligent processing on managed alerts.

Netcool/OMNIbus probes

- **Probes** are lightweight software components that collect event information and send it to the ObjectServer
- Using probes, the ObjectServer can be independent of the system being monitored
 - Many different probes for many different systems are available
 - Generic and vendor-specific probes exist
- Probes can modify and enrich event information
- Probes are resilient
 - Have storing and forwarding functions
 - Can be configured for automatic failover with other ObjectServers

Netcool/OMNIbus probes

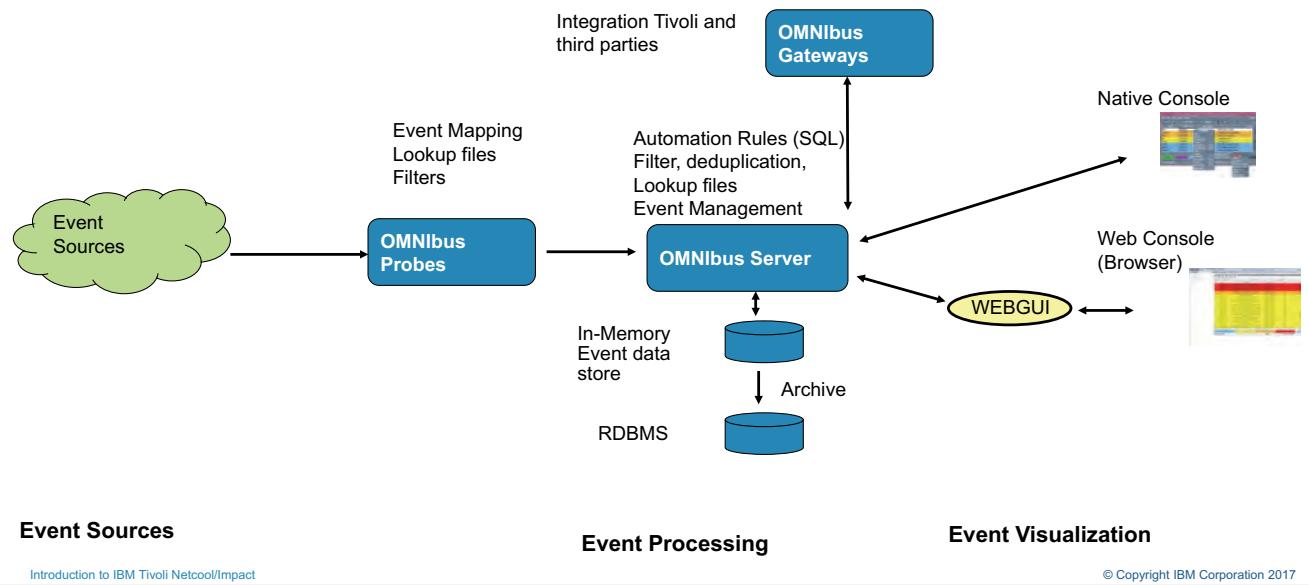
Netcool/OMNIbus probes use the logic that is specified in a rules file to manipulate the event elements. The probes then convert the event elements into fields of an event in the ObjectServer **alerts.status** table. Each probe is uniquely designed to acquire event data from a specific source.

Probes can acquire data from any stable data source, including devices, databases, and log files. Probes can also be configured to modify and add to the event data.

The ObjectServer **alerts.status** table is populated as a result of probe rules files. The files retrieve network events, automations, and user tools.

Often, business-level information does not exist in the original event but is available elsewhere in an organization's information technology infrastructure. Consider how it is possible to use information from other sources to add value to the Tivoli Netcool/OMNIbus events.

Tivoli Netcool/OMNIbus architecture overview



Event Sources

[Introduction to IBM Tivoli Netcool/Impact](#)

Event Processing

Tivoli Netcool/OMNIbus architecture overview

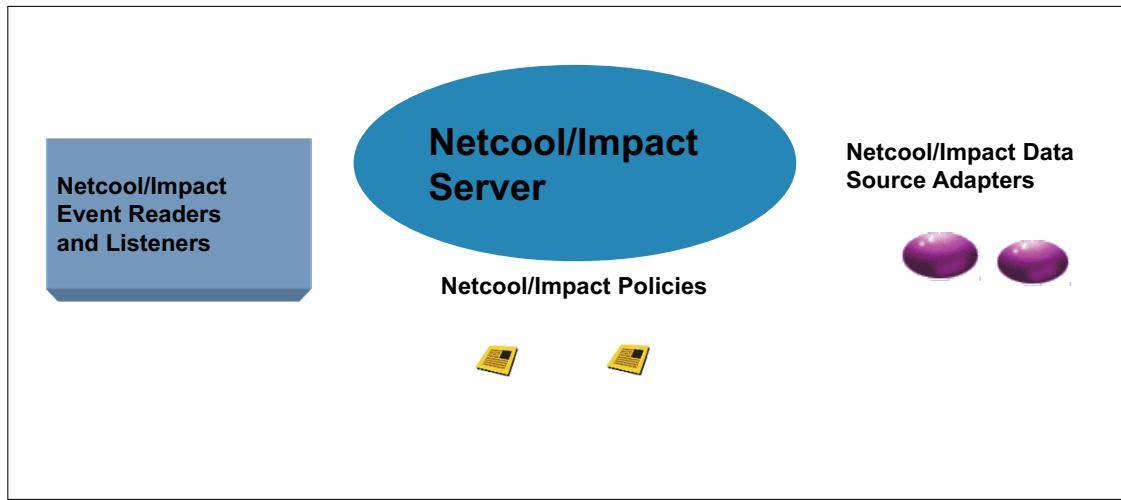
Event Visualization

© Copyright IBM Corporation 2017

This slide shows an overview of the Tivoli Netcool/OMNIbus component architecture. The Netcool/OMNIbus components work together to collect and manage network event information. Netcool/OMNIbus contains the following components:

- **ObjectServer:** The ObjectServer is the in-memory database server at the core of Tivoli Netcool/OMNIbus.
- **Probes:** Probes connect to an event source, detect, and acquire event data, and forward the data to the ObjectServer as events.
- **Gateways:** Netcool/OMNIbus gateways enable the exchange of events between ObjectServers and complementary non-IBM applications, such as databases and help desk or customer relationship management (CRM) systems.
- **Desktop tools:** The desktop is an integrated suite of graphical tools that you can use to view and manage events and to configure how event information is presented.
- **Administration tools:** Netcool/OMNIbus includes tools that administrators can use to configure and manage the system. These tools include the Netcool/OMNIbus Administrator, an SQL interactive interface, an import and export utility, and process control.

IBM Tivoli Netcool/Impact components



Netcool/Impact includes the following fundamental components:

- Netcool/Impact data source adapters (DSAs), or the Data Access Layer
- Netcool/Impact Event Readers and Listeners, or the Business Event Monitors
- Netcool/Impact Server: Policy Engine (Server/IDE) Impact policy editor
- Netcool/Impact policies

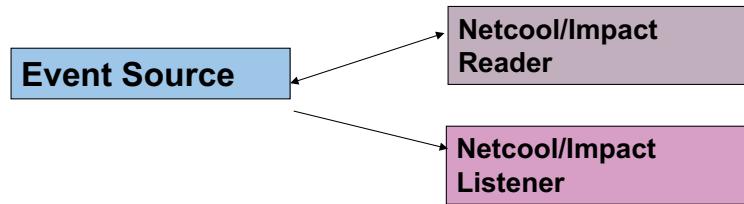


Note: The Federated Data Display (operator view) is not depicted.

Netcool/Impact event readers and listeners

Readers and listeners are responsible for capturing events in Netcool/Impact:

- **Readers** poll data sources for changes
- **Listeners** receive data sent from the data sources



Both readers and listeners support the full set of Netcool/Impact event operations. All policy functions can be used for either a reader or a listener policy. Readers and listeners are types of Netcool/Impact services.

Netcool/Impact policy engine

The policy engine is useful for these tasks:

- Evaluates events captured using readers and listeners
- Uses a preprogrammed set of rules called *policies* as it reacts to events
- Filters and directs events to one or more Netcool/Impact policies

Netcool/Impact policy engine

Netcool/Impact is an analytics application that runs behind the scenes after policies are configured. The Netcool/Impact results generally are seen in the event stores or through notifications. The only exception to this rule is the federated data or operator view, in which results can be seen immediately.

Netcool/Impact policies

Policies are preprogrammed sets of rules with the following capabilities:

- Interact with external applications
- Obtain events from Netcool/OMNibus and carry out field substitution
- Send emails and wait for replies
- Use a procedural scripting language
- Are used to build complex function combining multiple policies
- Can be simplified by useful action functions
- Provide functional capabilities for reusing code
- Can be assigned to projects

Netcool/Impact policies

Netcool/Impact policies are designed with a simple scripting language, that uses many action functions that provide predetermined common functions.

These functions are created to simplify the execution of commands such as these examples:

- Following links between data types
- Sending email
- Showing information to the user
- Adding new data to a data type
- Opening an external executable file
- Running a stored procedure in a database

Netcool/Impact data source adapters (DSAs)

DSAs are prewritten connectivity programs that provide these abilities:

- A variety of integration possibilities for cross-application use from an Netcool/Impact policy
- The ability to communicate externally using available DSAs, including these examples:
 - Relational Database DSA
 - Web Service DSA
 - Email and Instant Messaging DSA
 - XML DSA
 - JMS Messaging DSAs
 - SSH, Telnet, and TN3270 DSA
 - ITNM/IP DSA

Netcool/Impact data source adapters (DSAs)

Data source adapters (DSAs) are software components that Netcool/Impact uses to communicate with external data sources. DSAs broker information to and from SQL databases, LDAP servers, JMS topics and queues, and software systems that allow communication that uses web services APIs. Netcool/Impact also uses DSAs to parse XML strings and documents, communicate with web servers that use HTTP, and communicate with custom applications through generic socket transactions.

IBM Netcool/Impact operator views

- An **operator view** is a custom, web-based tool
- With operator views, users can view events and Netcool/Impact data in real time and can run Netcool/Impact policies based on that data
- Single pane of reference for multiple data views accessible through Netcool/Impact
- Predefined views for operators
- Template-based rendering (basic)
- Highly configurable smart tags (advanced)

IBM Netcool/Impact operator views

The operator view consists of the following components:

- **Display:** The single pane. It is an HTML file that is embedded with smart tags, causing data to be dynamically substituted at viewing.
- **Templates:** Display is generated from these files from a save operation in the Configuration GUI.
- **Assets:** Drawn from **netcool.war** and the assets repository area.
- **Policy:** Every display is associated with a Netcool/Impact policy to be executed before smart tags in the display are processed.
- **Configuration GUI:** In the Netcool/Impact GUI. Creates a basic operator view from a template.

From the Configuration GUI, the operator view does the following tasks:

- Generates and saves Netcool/Impact policies
- Reads and processes templates
- Generates and saves smart-tagged displays

Netcool/Impact V7.1 embedded products

- **Embedded WebSphere Liberty**
 - Netcool/Impact uses an embedded version of the IBM WebSphere Application Server
 - The Liberty server engine comes with a command-line tool: **wsadmin**
- **Apache Derby Database**
 - Derby is a relational database management system which supports Java and SQL.
 - Contains tables that store reporting information generated by the Netcool/Impact server

Netcool/Impact V7.1 embedded products

IBM Netcool/Impact uses WebSphere Application Service administration for component-specific management. WebSphere Liberty is described as a “Lightweight development and production runtime for web applications.”

IBM Tivoli Netcool/Impact concepts

- Netcool/Impact event management gives the Network Operations Center (NOC) and managers a business view of events by these actions:
 - Showing the impact of faults on customers
 - Reducing noise and highlights the main problems
 - Managing problems according to business effect
- Using the following event management concepts, managers can apply business rules to event processing:
 - Event Suppression
 - Event Correlation
 - Event Visualization
 - Event Aggregation
 - Event Enrichment

IBM Tivoli Netcool/Impact concepts

The role of Netcool/Impact in a Network Operations Center can be vital in delivering efficient and automated event management.

Automating some processes can reduce the amount of time that operators take to contact engineers to fix the problem. Prioritizing events according to business rules ensures that operators deal with the important problems first.

Providing managers with a way to view vital event, customer, and business information helps them make more informed decisions in prioritizing and managing relevant events.

Event suppression

- Not all events must be seen by an operator
- Netcool/Impact can suppress the following events:
 - Events from devices not yet provisioned
 - Events from devices undergoing maintenance
 - Events related to services that are critical only during defined time periods

Event suppression

The number of events within Netcool/OMNibus can be almost unusable by the operator. Even with Netcool/OMNibus, staff must handle large numbers of events and decide which events must be addressed.

With the correct data in external provisioning applications, Netcool/Impact can enable events to be suppressed from the operator event list, using a filter where necessary.

Event correlation

Includes event analysis, which determines whether events are *isolated* or *related*

Event correlation

Event correlation in Netcool/Impact simplifies and speeds the monitoring of network events by consolidating alerts and error logs.

For example, in a normal day a network administrator might need to monitor 20,000 events. Event correlation can reduce those 20,000 events to 50 alerts or log entries, which is more manageable on a daily basis.

Appropriate action would be taken based on that determination.

Event visualization

- ***Event visualization*** is a graphical view of the event management process
- IBM Tivoli Netcool/Webtop is a web-based application that processes data from one or more object servers and presents it in a graphical format
- You can use operator views to visualize the event management process

Event visualization

Event visualization provides a visual summary of event-related information.

Event aggregation

- Event aggregation is useful when a sequence of events requires analysis
- Netcool/Impact can create new synthetic events:
 - Netcool/Impact can generate new events that summarize the service status for a particular customer by examining all the events relating to that customer
 - Netcool/Impact can generate summary events giving the status of particular equipment types, such as the real-time total number of ports down
 - Netcool/Impact can analyze events over time; for example, it can suppress alarms from a device unless more than 10 alarms arrive in 5 minutes
 - Synthetic events can be used to drive the operator view

Event aggregation

A Netcool/OMNIbus event can give aggregate event information in the event list. This new event can be generated based on different criteria, such as these examples:

- The number of events that affect a particular customer
- The number of events that affect a defined service
- The number of events from a particular network element

Using the event summary, Network Operations Center (NOC) staff can see immediately which business area is affected, without having to sort through the events in the event list.

Event enrichment

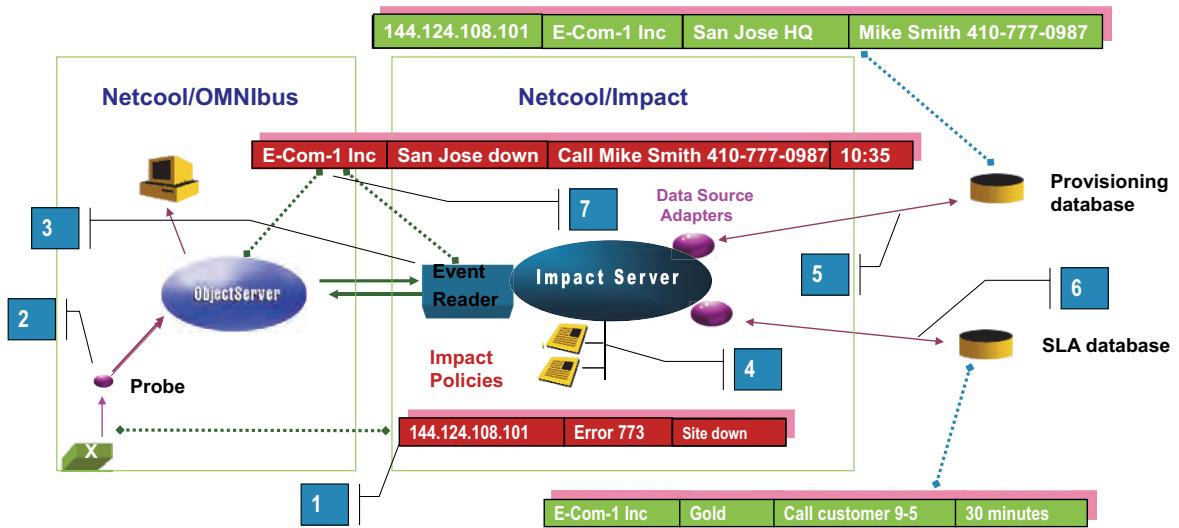
- Events from probes do not always reveal everything about a particular event
- Netcool/Impact can enrich the incoming events with additional information from relevant data sources:
 - Add customer details
 - Add service level agreement (SLA) information
 - Route the event to the responsible operator
 - Change the severity of the event according to the customer
 - Give the Network Operations Center (NOC) a view of faults from a customer's perspective

Event enrichment

Event enrichment is the most common use of the Netcool/Impact product. Using the data from the external databases and applications, the Netcool/OMNibus event can be enriched with this data where necessary.

Providing this enriched event to the NOC staff enables them to see the event from the customer perspective. They can then make more informed decisions on what action to take.

The event enrichment life cycle



The event enrichment life cycle

This example shows how event fields from Netcool/OMNibus can be enhanced (enriched) to add important customer information to the event.

The resulting event provides the NOC staff with live information, such as the customer name, location, and contact details. It then prioritizes the event according to the service level agreement (SLA) of that customer.

1. A new event occurs on the monitored equipment.
2. The probe for the ObjectServer reads and formats the event.
3. The Netcool/Impact event reader detects the event in the ObjectServer.
4. The event reader opens a policy to enrich the event.
5. The policy accesses external provisioning information using the configured data source adapters.
6. The policy accesses external customer SLA information using the configured data source adapters.
7. The policy updates the event information for the event in the ObjectServer.

Student exercises



Student exercises

Open your *Student Exercise* book and complete the exercises for this unit.

Review questions

1. What is an event?
2. What is Tivoli Netcool/OMNibus?
3. What is event enrichment?

Review answers

1. What is an event?

*An **event** is any change that occurs within an information technology infrastructure that can be captured electronically.*

2. What is Tivoli Netcool/OMNibus?

Tivoli Netcool/OMNibus tracks alert information in a high-performance, in-memory database. It presents information of interest to specific users through individually configurable filters and views.

3. What is event enrichment?

Event enrichment is the most common use of Netcool/Impact. Using the data from the external databases and applications, the Netcool/OMNibus event can be enriched with this data where necessary.

Summary

You now can perform the following tasks:

- Describe the Tivoli Netcool/Impact product and its components
- Define the role of Netcool/Impact in event management
- Describe the Netcool/Impact enrichment life cycle

Unit 2 The Netcool/Impact user interface

IBM Training



The Netcool/Impact user interface

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes how to manage and create Netcool/Impact objects within the user interface.

Objectives

- In this unit, you learn how to perform the following tasks:
- Describe the difference between a project and the global repository
- Create a project
- Describe version control

Overview of the Netcool/Impact user interface



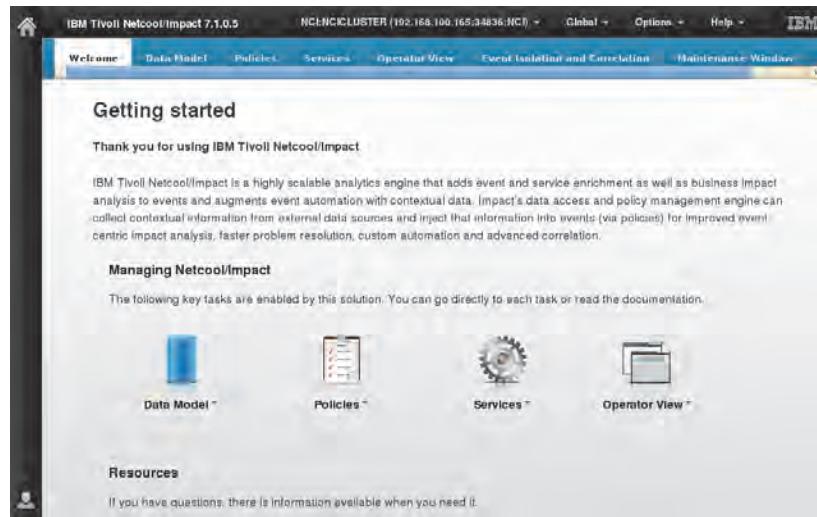
The Netcool/Impact user interface

© Copyright IBM Corporation 2017

Overview of the Netcool/Impact user interface

The user name and password are created during installation. The default user name is **impactadmin**.

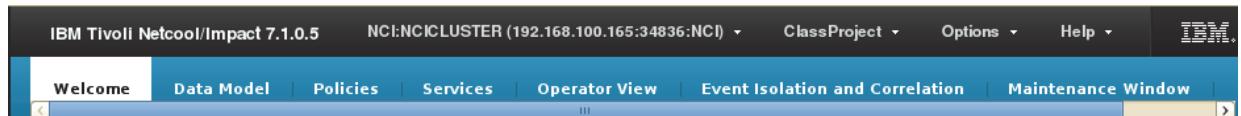
IBM Tivoli Netcool/Impact Welcome screen



IBM Tivoli Netcool/Impact Welcome screen

After authentication, the Netcool/Impact Welcome screen opens. The tabs across the top of the screen provide a quick visual summary of the available features.

The Tivoli Netcool/Impact view



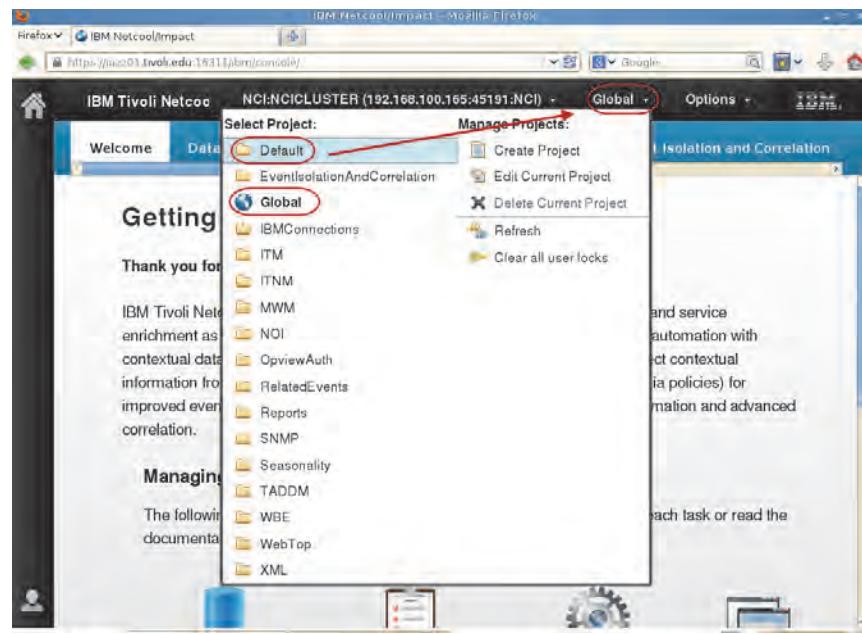
The Netcool/Impact user interface

© Copyright IBM Corporation 2017

The Tivoli Netcool/Impact view

By default the product opens the first time in the Global project.

The Project view versus the Global view



The Netcool/Impact user interface

© Copyright IBM Corporation 2017

The Project view versus the Global view

To change the project click the arrow next to Global to see the available projects.

Benefits of using projects

- You can manage policies and their associated elements easily and efficiently
- It is easier to determine which data types and services relate to each policy and how the policies relate to each other
- Projects assist when determining whether a policy, or its associated data types or services, should be deleted

Benefits of using projects

The creation of projects helps the management and the organization of Netcool/Impact components.

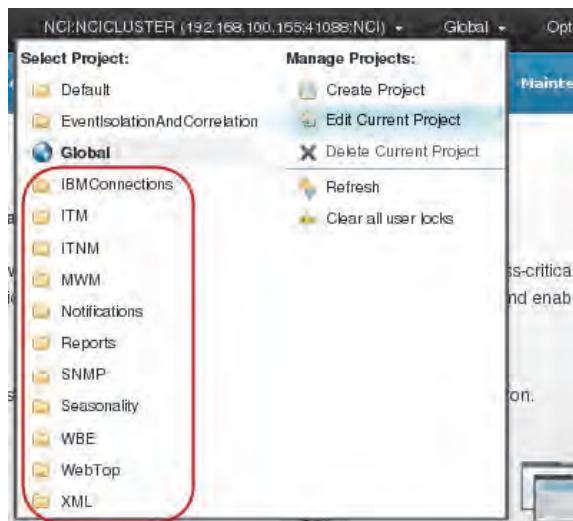
The project list

- Each project in the list of projects, when selected, shows a subset of the data that is stored in the Netcool/Impact global repository
- A project can consist of the following related elements:
 - Policies
 - Data sources set up for project data types
 - Data types associated with the policies
 - Operator views related to the policies
 - Reports
 - User-defined services

The project that is named **Default** is the Netcool/Impact default project.

Prebuilt projects

Product
Extensions



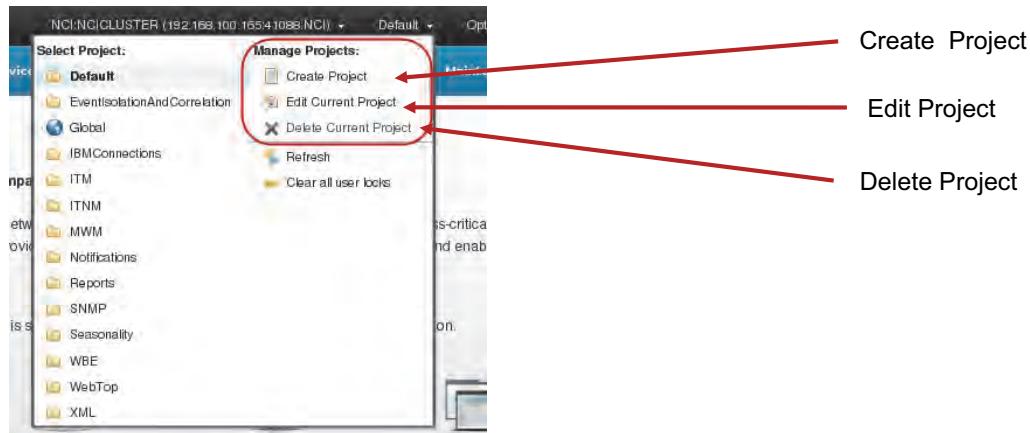
The Netcool/Impact user interface

© Copyright IBM Corporation 2017

Prebuilt projects

Netcool/Impact includes prebuilt projects, including all the components that you need to test function. You can make modifications to support specific implementation requirements. You can rename, edit, or delete all of the project elements.

Project elements



Create Project

Edit Project

Delete Project

Project elements

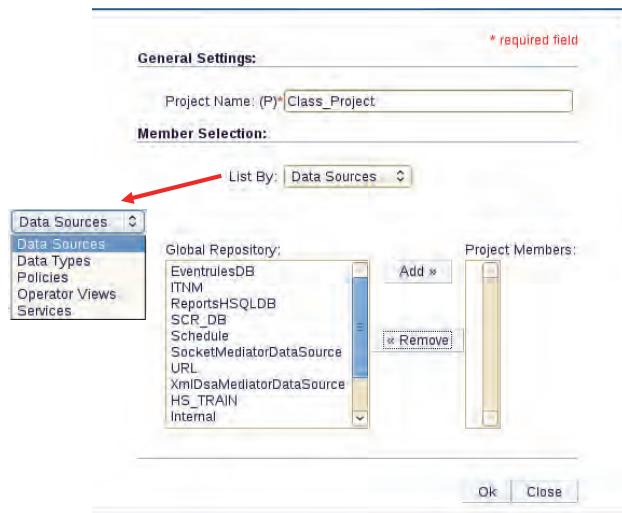
You can accomplish the following tasks while working with projects:

- View projects and project members
- Create projects
- Edit projects
- Add and delete project members
- Delete projects
- Copy the data sources, data types, policies, and services in a project between two running server clusters on a network

Creating a project

To create a new project, follow these steps:

1. Click any entry in the Netcool/Impact pane to open an edit window
2. Click the **New Project** icon
3. Select a **List By** option
4. Highlight Global Repository elements
5. Click **Add** to include the element in the new project as a Project Member
6. Click **Ok**



Creating a project

On the General Settings window, you can select objects from the global repository to add to a new or existing project. The **List By** drop-down list populates the **Global Repository** list, depending on the selection. Highlight an entry in the list, click **Add**, and the global repository entry is added to the **Project Members** list. The new entry is included as a part of the project that is listed in the **Project Name** field.

The global repository

- The global repository is the storage area for the following items for the connected server:
 - Policies
 - Data types
 - Data sources
 - Operator views
 - Reports
 - Services
- You can add the items you create to the global repository or to any other project at any time

The global repository

The **global repository** is the storage area for all Netcool/Impact policies, data types, data sources, operator views, reports, and services for the Netcool/Impact server.

Global repository elements

When editing and deleting items in projects and the global repository, be aware of these items:

- Editing a repository element changes it everywhere
- Deleting a repository element deletes it permanently from the server, the global repository, and every project it is attached to

Global repository elements

The editing function available from the global repository is similar to that of the other projects. Deleting an item from a list in the global repository removes it permanently from the Netcool/Impact database. It also removes the item from all projects of which it is a member.

Be careful to delete only items that you want to delete globally. The only safe way to remove an item from a project is from the **Project** window.

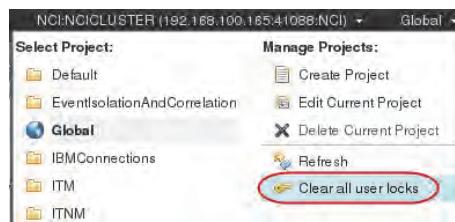
Version control

- Netcool/Impact uses version control to ensure that no changes are made to its elements, data sources, or data types by more than one person simultaneously
- Impact Subversion is installed by default with Netcool/Impact version 6.1
- Version control is enabled for Netcool/Impact by default and cannot be disabled
- Netcool/Impact supports the following external version control systems:
 - Impact Subversion (default)
 - Concurrent Versions System (CVS)
 - RCS
 - ClearCase
 - Subversion (SVN)

The default version control system that the version control manager uses is Impact Subversion (SVN). Impact Subversion is customized for use with Netcool/Impact.

Version control interface

- Netcool/Impact is installed with a version control interface with which you can save data as revisions in a version control archive
- When a new policy, data source, data type, or service is created, a corresponding element in the version control system is also created
- When an item is opened for viewing or editing, it is checked out of the version control system and other users are prevented from editing it
- When the item is saved, it is checked back in and available to other users
- The **Unlock ALL** icon is available from the global repository checks in all locked elements



The Netcool/Impact user interface

© Copyright IBM Corporation 2017

Version control interface

Opening objects in the Netcool/Impact GUI checks them out of the version control system and locks them to prevent other users from editing them. When objects are saved, they are checked back in and are available to other users.

On occasion, someone might check out a policy, but never check it back in. If a policy is locked, use **Unlock All** to release the checkout locking mechanism, so that policies can be edited again.

Typically, this situation is caused by a lockup or forced shut down of the Netcool/Impact GUI while policies are being edited.

Version control script

- The version control script is located in **\$NCHOME/bin/**: **nci_version_control**
- The script is used to perform housekeeping and maintenance tasks on data stored in the version control system that you cannot otherwise access using the GUI or the command-line interface
- The script requires one of the following commands:
 - **add**: adds a file to version control
 - **checkpoint**: labels and tags repositories
 - **ci**: check in
 - **co**: check out
 - **unco**: uncheck and unlock
 - **update**: update with specified revision
 - **server**: checks out the Tivoli Integrated Portal revision of all the files in version control
- Example of the **add** command:

```
nci_version_control NCI add /opt/IBM/tivoli/impact/etc/NCI_test.props admin
```

Version control script

Use the **nci_version_control** command to modify version control. The syntax for the **add** command is as follows:

```
add FILE_NAME USERNAME
```

To add the **NCI_test.props** file to the underlying version control system, use the following example:

```
nci_version_control NCI add /opt/IBM/tivoli/impact/etc/NCI_test.props admin
```

Student exercises



The Netcool/Impact user interface

© Copyright IBM Corporation 2017

Student exercises

Open your Student Exercise book and perform the exercises for this unit.

Review questions

1. What is the **Global** tab used for?
2. What is SVN?
3. How do you release the checkout locking mechanism?

Review answers

1. What is the **Global** tab used for?

*Use the **Global** tab to access the storage area for all the Netcool/Impact policies, data types, data sources, operator views, reports, and services for the Netcool/Impact server.*

2. What is SVN?

Subversion is used for version control.

3. How do you release the checkout locking mechanism?

*Click **Unlock All** to release the checkout locking mechanism.*

Summary

You now can perform the following tasks:

- Describe the difference between a project and the global repository
- Create a project
- Describe version control

Unit 3 The Netcool/Impact data model

IBM Training

IBM

The Netcool/Impact data model

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

The first step in developing a Netcool/Impact solution is to determine the data that will be used. This unit covers how to access data inside the Netcool/Impact structure, and how to build data sources and data types.

Reference: *IBM Tivoli Netcool/Impact Administration Guide*

Objectives

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

- Describe a Netcool/Impact solution
- Create data sources, data types, and data items
- Describe the purpose of links
- Describe event sources

Netcool/Impact solutions

Netcool/Impact solutions include the following components:

- Data models
- Services
- Policies

Netcool/Impact solutions

This data does not necessarily exist in a database or in a form that is easily accessible. The data can be stored in spreadsheets, or it can be paper-based. Gather all of the details about the status of the data, including accessibility.

You use data models, services, and policies to describe the data and then act upon that data to provide meaningful information. Subsequent units in this course describe policies and services in greater detail.

Data models

- A data model is a schematic representation of a business and any associated metadata
- A Netcool/Impact data model consists of the following components:
 - Data sources
 - Data types
 - Data items
 - Links
 - Event sources

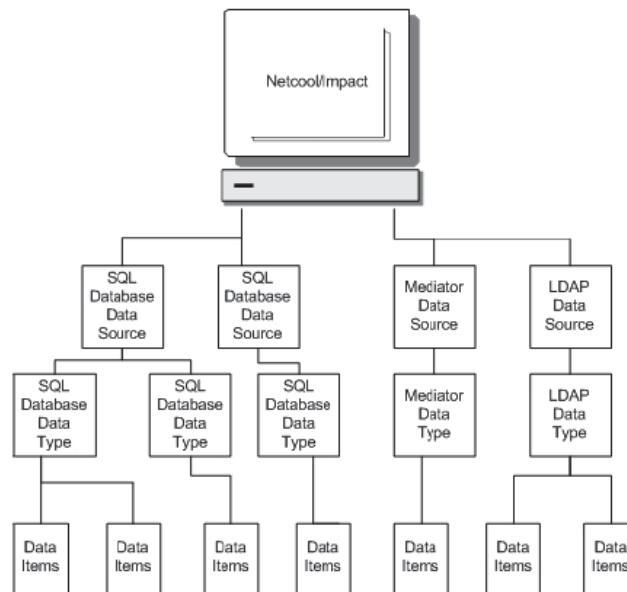
Data models

A data schematic is a useful tool, but do not rely on its existence. If the data does not exist in a readily accessible form, for example, if it must be converted from a spreadsheet to a flat file or database table, you might have to create it. You can save significant time and effort by using a data schematic.

A Netcool/Impact data model consists of the following components:

- **Data sources** provide the connection to the physical database instance.
- **Data types** use the data source to connect to tables in the relevant data source.
- **Data items** are the data itself or the contents of the table that the data type is discovering.
- **Links** can be created between fields in the tables by exposing relationships between disparate data types. By creating links, you can define a schema in Netcool/Impact between databases and applications of various types.
- **Event sources** are the originators of information technology data.

Data model architecture



The Netcool/Impact data model

© Copyright IBM Corporation 2016

Data model architecture

This diagram depicts the current Netcool/Impact architecture, showing how the data modeling components connect.

Generally, you set up a data model once, when you first design your Netcool/Impact solution. After that, you do not have to actively manage the data model unless you change the solution design.

To set up a data model, you must first determine what data to use in your solution and where that data is stored. Then you define a Netcool/Impact data source for each source of data. A data type is created for each structural element that contains the data needed. Optionally, you can create dynamic links between data types or static links between data items, making it easier to traverse the data programmatically from within a policy.

You manage a data model using the Netcool/Impact GUI. Using the GUI, you can view, create, edit, and delete all of the components of a data model.

Data sources

- A data source is an element of the Netcool/Impact data model that represents a real-world source of data in your environment
- Netcool/Impact provides the following categories of data sources:
 - SQL database data sources
 - LDAP data sources
 - Mediator data sources
 - Internal data repository

Data sources

Data sources contain the host, port, and login information necessary to connect to external data. Netcool/Impact supports many different data sources with available data source adapters, including the ones in the following list.

Data source type	Data source adapter
Netcool/OMNIbus ObjectServer	DB2®
Microsoft SQL Server®	MySQL
ODBC	Oracle
Derby	Sybase
Flat file	LDAP
SNMP	JMS
XML	Web Services

SQL database data sources

- SQL database data sources represent relational databases
- Netcool/Impact supports most of the popular commercial relational databases and freely available databases
- The Netcool/OMNIBus ObjectServer is also supported as an SQL data source

SQL database data sources

When you configure an SQL database data source, the connection criteria are defined for a particular database.

Lightweight directory access protocol (LDAP) data sources

- LDAP is an Internet application protocol that email and other programs use to query and modify directory services running over TCP/IP
- LDAP data sources represent LDAP directory servers that Netcool/Impact supports:
 - Netscape
 - iPlanet
 - OpenLDAP
 - Microsoft Active Directory servers

Lightweight directory access protocol (LDAP) data sources

See the *IBM Tivoli Netcool/Impact Administration Guide* for more information about supported LDAP data sources.

Mediator data sources

- Mediator data sources represent third-party applications that are integrated with Netcool/Impact through the Mediator DSA
- These data sources include a wide variety of network inventory, network provisioning, and messaging system software
- In addition, you can use providers of XML and SNMP data as mediator data sources

Mediator data sources

Typically, the Mediator DSA data sources and their data types are installed when you install a Mediator DSA. The data sources are available for viewing and, if necessary, for creating or editing.

The CORBAMediatorDSA, DirectMediatorDSA, and SNMPPDirectMediatorDSA are examples of supported mediator DSAs.

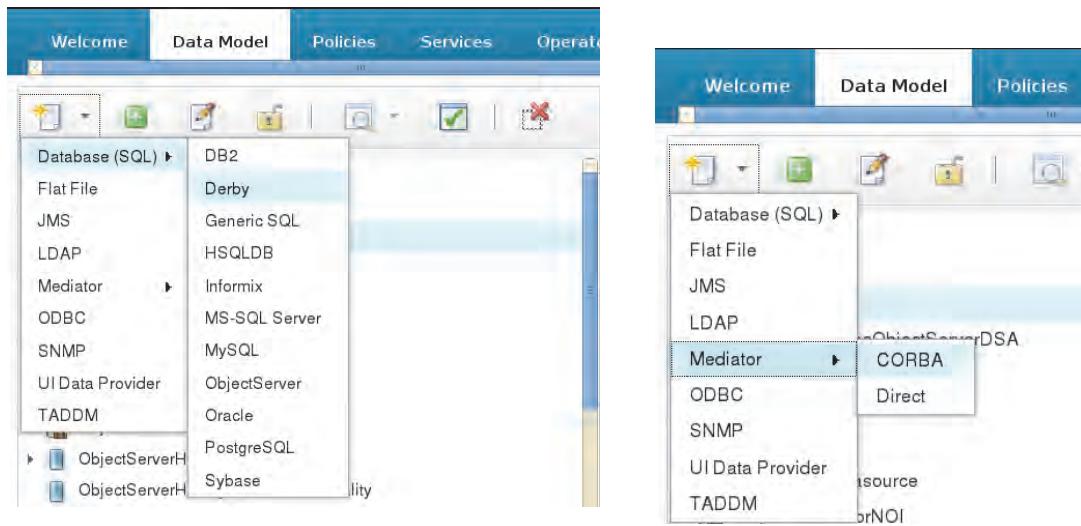
Internal data repository

- The Internal data repository is a built-in data source for Netcool/Impact
- The primary responsibility of the Internal data repository is to store system data
- You can also use it to store the data in predefined and user-defined internal data types

The internal data repository

Netcool/Impact creates an internal database during installation to use as the internal data repository.

Available data sources



Available data sources

Netcool/Impact automatically installs data source adapters during installation. You can open the full list of supported data sources by expanding the list in the **Data Model** window.

Creating a data source

The screenshot shows the Derby Data Source Editor interface. On the left, under 'General Settings', the 'Data Source Name' is set to 'HS_TRAIN', 'Username' to 'impact', and 'Password' to a masked value. The 'Maximum SQL Connection' is set to 5. Under 'Database Failure Policy', the 'Fail over' and 'Fail back' options are available, with 'Disable Backup' selected. On the right, the 'Primary Source' section is configured with 'Host Name' as 'jazz01.tivoli.edu', 'Port' as '1527', and 'Database' as 'ImpactDB'. A 'Test Connection' button is present. Below it, the 'Backup Source' section is configured with 'Host Name' as 'localhost', 'Port' as '1527', and 'Database' as 'database'. Another 'Test Connection' button is present.

The Netcool/Impact data model

© Copyright IBM Corporation 2016

Creating a data source

Follow these steps to create a data source:

1. Click **Data Model** to open the **Data Model** tab.
2. Select a project from the **Project** list.
3. Click the **New Data Source** icon.
4. Select a data source type from the list.
5. In the configuration window, supply the required configuration properties.
6. Click the **Save** icon.
7. Click **Test Connection**.

Data types

- Data types are elements of the Netcool/Impact data model that represent sets of data stored in a data source
- The structure of this data depends on the category of the data source where it is stored
- Netcool/Impact supports the following categories of data types:
 - SQL database data types
 - LDAP data types
 - Mediator data types
 - Internal data types

Data types

The data that Netcool/Impact works with is organized into types. A data type represents a collection of data in a data source. Netcool/Impact has both external and built-in, or internal, data types. All created data types must belong to a data source, either external or internal.

The purpose of data types

- Data types provide an abstract layer between Netcool/Impact and the associated set of data in a data source
- Netcool/Impact uses data types to locate the data to be used in a policy
- Create one data type for each table or other data structure in the data source that contains information to be used in a policy

The purpose of data types

Data types describe the content and structure of the data in the data source.

SQL and LDAP data types

- If the data source is an SQL database, each data type corresponds to a database table
- If the data source is an LDAP server, each data type corresponds to a type of node in the LDAP hierarchy
- A data type definition contains the following information:
 - The name of the underlying table or other structural element in the data source
 - A list of fields that represent columns in the underlying table or another structural element
 - Settings that define how Netcool/Impact stores data in the data type

Each data source type must have a corresponding data type.

Mediator data types

- Mediator data types represent data that is managed by third-party applications such as these examples:
 - A network inventory manager
 - A messaging service
- Typically, mediator data types do not represent data stored in database tables
- Instead, they represent collections of data that are stored and provided by the data source in various other formats, such as these examples:
 - Sets of data objects
 - Messages

Internal data types

Netcool/Impact provides the following categories of internal data types to correspond with internal data sources:

- System
- Predefined
- User-defined

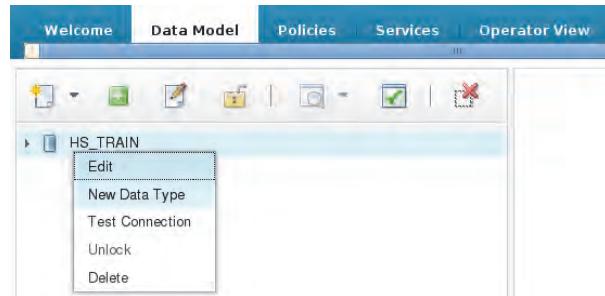
Internal data types

Data can be stored directly in a Netcool/Impact data repository and used as a data source. The internal data types can be defined to create and access this data.

Use internally stored data types to model data that does not exist or that cannot be easily created, in external databases. This data includes working data used by policies, which can contain copies of external data or intermediate values of data.

Internal data types are not designed for use outside of Netcool/Impact. They contain relatively small amounts of data.

Creating an SQL data type



Right-click the data source name and select

New Data Type

Creating an SQL data type

You can create, modify, or delete data types in the **Data Sources And Types** list.

Follow these steps to create a data type:

1. Select the **Default** project.
2. Click the **Data Model** tab.
3. Right-click a data source and select **New Data Type** from the list.

The data type configuration window opens in the main configuration window.

SQL data type configuration

The screenshot shows the 'New Data Type for HS_TRAIN' configuration page. It has two main sections: 'General Settings' and 'Table Description'.

General Settings:

- Data Type Name: HS_Node
- Data Source Name: HS_TRAIN
- Enabled:
- Access the data through UI data provider

Table Description:

- Configure the database table and fields. An * indicates required fields.
- Schemas: IMPACT (Completed fetching schemas)
- Tables: NODE (Completed table discovery)
- Refresh Fields** button
- Show New / Deleted Fields

Table Description Panel:

- Table Description: SQL Data Type Config Page
- Dynamic Links: Cache Settings
- Table Description: (empty)
- Refresh Fields** button
- Show New / Deleted Fields
- Schemas: IMPACT (Completed fetching schemas)
- Tables: NODE (Completed table discovery)

ID	Field Name	Format	Display Name	Description	Key Field
NODE	NODE	STRING	NODE	NODE	<input checked="" type="checkbox"/>
IP	IP	STRING	IP	IP	No
NODEID	NODEID	INTEGER	NODEID	NODEID	No

The Netcool/Impact data model

© Copyright IBM Corporation 2016

SQL data type configuration

To configure a data type configuration, provide the following information:

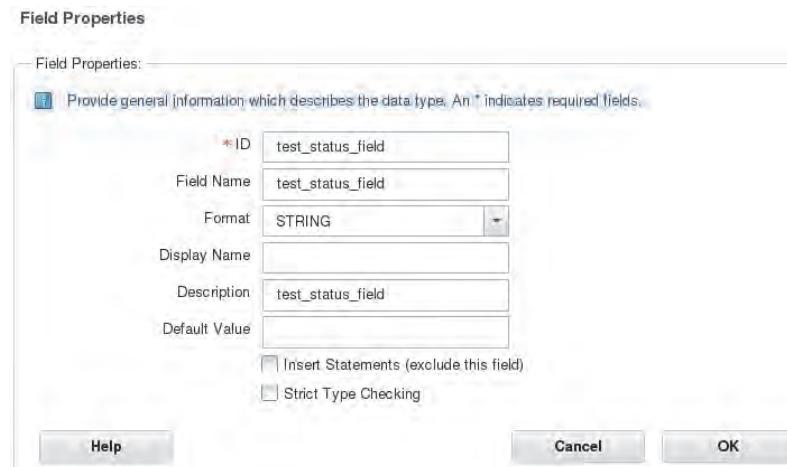
- Provide a unique name for the data type.
- Specify the name of the underlying data source for the data type.
- Specify the name of the database and the table where the underlying data is stored.
- Use **Refresh** to autopopulate the fields in the data type.
- Specify key fields for the data type.
- Save the definition.

Data items

- **Data items** are elements of the data model that represent actual units of data stored in a data source
- The structure of this unit of data depends on the category of the associated data source

When Netcool/Impact auto discovers the fields in the table, it continues to configure the data type. When Netcool/Impact identifies the fields in the table, you can change them and remove any unwanted fields. Removing fields from the data type configuration forces Netcool/Impact to retrieve only the fields that are deemed necessary, thus speeding data retrieval.

New Field window



The Netcool/Impact data model

© Copyright IBM Corporation 2016

New Field window

The **Field Properties** window opens after you click the **New Field** icon. After you enter the properties and click **OK** to accept them, a new field is added to the internal table.

Links

- **Links** are elements of the data model that define relationships between data types and data items
- **Dynamic links** define relationships between data types
- **Static links** define relationships between data items
- Links are an optional component of a Netcool/Impact data model

Links

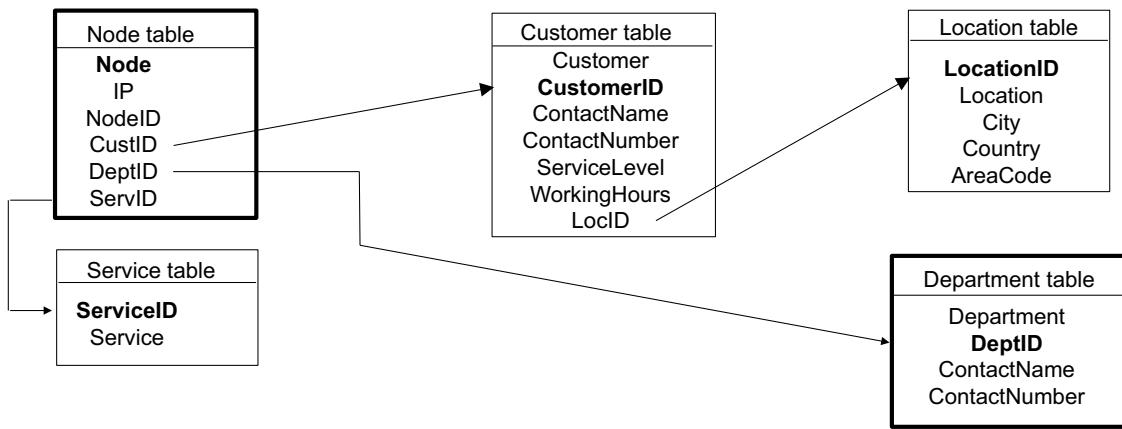
In Netcool/Impact data storage, links assist in creating or exposing relationships among the various data types or data items. You can create links between either external data or internal data types. Multiple links from each type are supported. This concept is similar to a JOIN function in an SQL database.

For example, assume that there are two tables that contain the following information:

- **Table 1** contains customer information, such as the name, phone number, and address. The table also has a unique customer ID key.
- **Table 2** contains a list of servers. In this table, the customer ID of the customer that owns the server is included.

Even if these data items are kept in different databases, you can create a link between Table 1 and Table 2 through the **customer ID** field. All of the servers owned by a particular customer are visible.

Database Schema example



The Netcool/Impact data model

© Copyright IBM Corporation 2016

Database Schema example

This diagram describes the relationships between the data, specifically, the links between the Node, Customer, Department, Service, and Location tables. It is important to understand how these relationships work and realize their flexibility.

In the example, all the secondary data retrieved from the training database is linked to events through the Node table.

Types of links

Netcool/Impact provides the following categories of links:

- Static
- Dynamic
- Composite

The three link options available in the Netcool/Impact product are static links, dynamic links, and composite links.

Static links

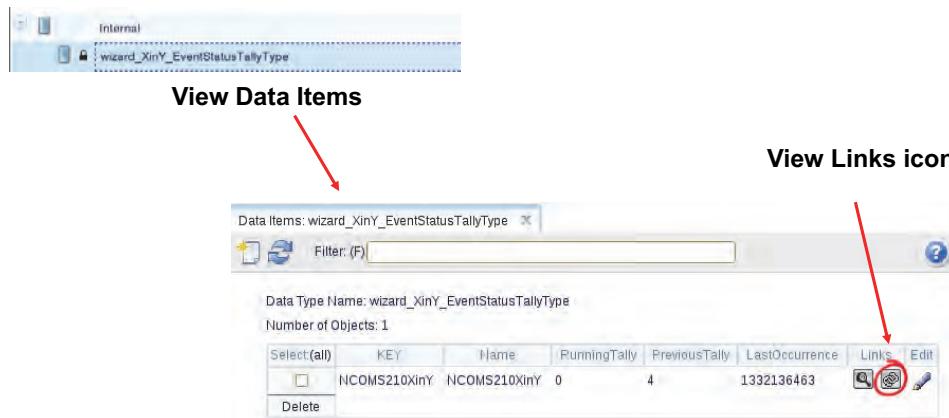
- Static links define a relationship between data items in internal data types
- Static links are not supported for other categories of data types, such as SQL database and LDAP types
- There is no way for Netcool/Impact to ensure the persistence of data items that are stored externally

Static links

As the term *static* suggests, the relationship between data items in static links never changes after you create them. Static links can be traversed in a policy or in the user interface when you browse the linked data items.

A static link is manually created between two data items when relationships do not exist at the database level. Static links are bidirectional and are supported for internal data types only.

Creating a static link



Creating a static link

To set up a static link, perform the following steps:

1. In the Netcool/Impact GUI, click **Data Model**.
2. Click the name of the internal data type that contains the source data items that you want to link.
3. Click the **Links** button for the data item you want to link.
4. In the Static Links window that opens, select the data type that contains the data items you want to link to.
5. Select the data item to link to from the list of data items that appears.

Dynamic links

- Dynamic links define a relationship between data types
- This relationship is specified when you create the link
 - It is evaluated in real time when Netcool/Impact encounters a call to the GetByLinks function in a policy
- Dynamic links are supported for internal, SQL database, and LDAP data types
- Netcool/Impact supports the following types of dynamic links:
 - Link by key
 - Link by filter
 - Link by policy

Dynamic links

Dynamic links are used only at the database level. Dynamic links are unidirectional links that are configured from the source to the target data type.

You can configure the following three types of dynamic links:

- **Link by key** evaluates an expression from one type and matches it to the **Key** field of another type. The **Key** field is only required in the target type. You do not have to specify the **Key** field, because Netcool/Impact recognizes this type of definition already. However, you must define the name of the **Source** field.
- **Link by filter** provides greater flexibility than the link by key, because a link by filter can be any expression between any two fields of any type. You must specify both the **Source** and **Target** fields. These can be key fields, but they always must follow this format:

Target Field Name = %Source Field Value%

The percent sign (%) indicates that it is a field in the source data type. If the field is a character, it must also be enclosed in single quotation marks.

- **Link by policy** uses a policy to expose the links between types. You can use this link type to define a many-to-many relationship between types.

Creating a Dynamic Link Example: Link by Filter

The screenshot shows the 'Data Type Name: HS_Department' page. On the right, there's a 'View Links' section with a 'View Details' link for 'Technical Services - US(HS_Department)'. Below it is a table with columns: Select, DEPARTMENT, DEPTID, CONTACTNAME, CONTACTNUMBER, View Links, and Edit. The table contains three rows: 'Technical Services - US' (DEPTID 1), 'Technical Services - EMEA' (DEPTID 2), and 'Finance' (DEPTID 3). The 'View Links' column for each row has a magnifying glass icon circled in red.

Below the table is the 'SQL Data Type Config Page' with tabs for Table Description, Dynamic Links, and Cache Settings. The 'Dynamic Links' tab is selected, showing a sub-section titled 'Link By Filter' with a red circle around it. A tooltip explains: 'Use the Dynamic Links By Filter table to manage dynamic links. A dynamic link is a relationship between two data types specified using the link filter syntax.' Below this is a 'Delete Selection' button and a 'New' button circled in red. A table below shows 'Target Data Type', 'Value', and 'Exposed Link Type', with a note 'No items to display'.

The Netcool/Impact data model

© Copyright IBM Corporation 2016

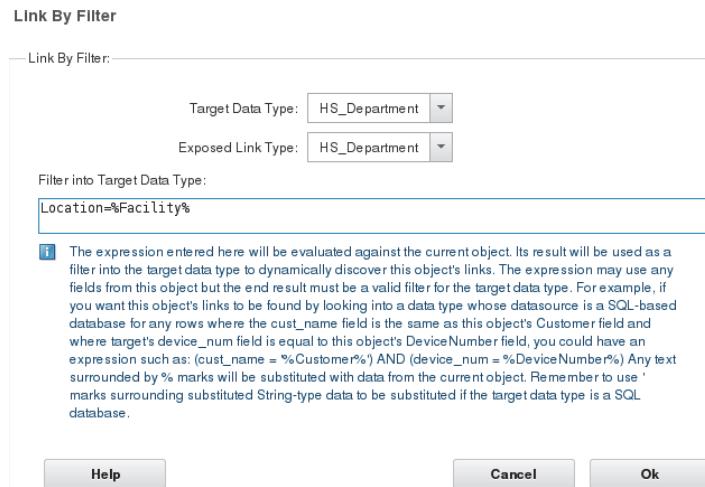
Creating a Dynamic Link Example: Link by Filter

Data types from one table can be linked to data types in other tables by filter, by key, or by policy.

In this example, a dynamic link by filter is created by performing these steps:

1. Open the Data Type editor.
2. Click a data type name.
3. Click **New Link By Filter**.

Link by Filter window



Link by Filter window

On the **Link By Filter** window, you can enter a filter expression. The expression limits the data that the target data type selects to dynamically discover the object links.

1. Select the target data type from the Target Data Types list.
2. Type the filter syntax for the link in the **Filter into Target Data Type** field.
3. Click **OK**.

Composite links: Virtual data types

- In some cases, it is convenient to create a virtual data type that contains data from multiple data types linked together
- All data types are still required
- To configure, add fields to current data types that reference other types

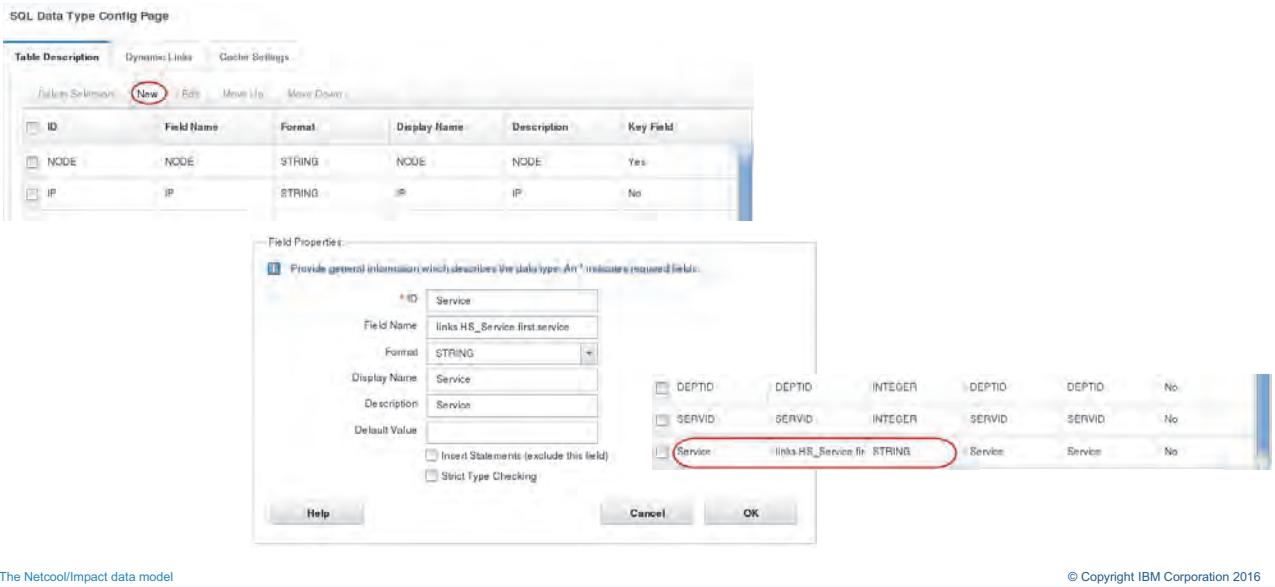
Composite links: Virtual data types

Often, for event enrichment, all that is needed from the specific data types is data to populate the ObjectServer fields or data to link to other data. The concept of *composite data type* makes lookups easier.

Composite data types create what are often known as *views*, which show multiple sets of linked data in one read-only table. This type of link only simplifies the data that a data policy uses. You can preconfigure commonly referenced links in the data types without having to reference the LinkBy attributes each time.

Adding the composite field

Add the field **Service** to the Node table



Adding the composite field

To add a composite field, click the **New Field** icon from the **Data Type** window. The **Field Properties** window opens.

Viewing a data type with composite fields

Data Type Name: PG_Node

Number of Objects: 43

Select: (all)	node (key)	ip	nodeid	custid	deptid	servid	Service	Links	Edit
<input type="checkbox"/>	gambit	10.10.10.10	0	1	1	1	Web Hosting	 	
<input type="checkbox"/>	wombat	10.10.10.1	1	2	2	2	Network Management	 	
<input type="checkbox"/>	orac	10.10.10.2	2	3	3	3	Home Banking	 	
<input type="checkbox"/>	muppet	10.10.10.3	3	4	4	0	NULL	 	
<input type="checkbox"/>	moose	10.10.10.4	4	5	5	5	Trading Floor	 	
<input type="checkbox"/>	hal	10.10.10.5	5	6	6	6	Micromuse University	 	
<input type="checkbox"/>	vixen	10.10.10.6	6	7	7	7	Email	 	
<input type="checkbox"/>	dewey	10.10.10.7	7	8	8	1	Web Hosting	 	
<input type="checkbox"/>	angel	10.10.10.8	8	0	9	2	Network Management	 	
<input type="checkbox"/>	link1	10.10.10.9	9	0	10	0	NULL	 	

Viewing a data type with composite fields

You can view the linked field as if it were another actual field in the table.

Event sources

- An **event source** is a special type of data source
- Each event source represents an application that stores and manages events
- The most common event source is the ObjectServer database
- With Netcool/Impact, other applications can also be used as event sources

Event sources

The primary event source that is used in this course is Netcool/OMNibus.

Types of event sources

Netcool/Impact provides the following types of event sources:

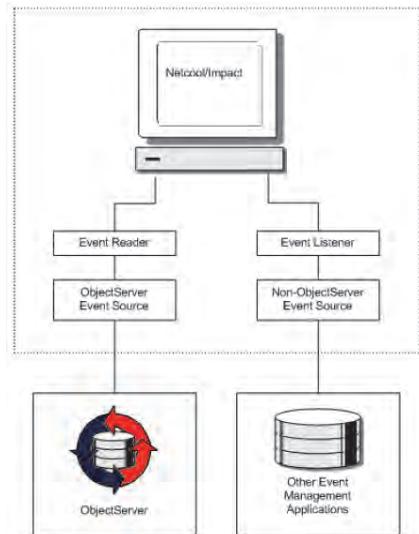
- ObjectServer event sources
- Non-ObjectServer event sources

Types of event sources

ObjectServer event sources represent instances of the Netcool/OMNIBus ObjectServer database. This type of event source is the most common. ObjectServer events are alerts that are stored in the **alerts.status** table of the database. These alerts have a predefined set of alert fields that you can supplement by adding fields that you define. Netcool/Impact monitors ObjectServer event sources using a Netcool/OMNIBus event reader service. The event reader service queries the ObjectServer at intervals and retrieves any new, updated, or deleted events that match its predefined filter conditions. The event reader then passes each event to the Netcool/Impact policy engine for processing.

Although ObjectServer event sources are the most common, there are also non-ObjectServer event sources.

Event source architecture



The Netcool/Impact data model

© Copyright IBM Corporation 2016

Event source architecture

The event source architecture depicts the relationship between Netcool/Impact and the event reader and the event listener. Non-ObjectServer event sources represent instances of other applications, such as external databases or messaging systems that provide events to Netcool/Impact. Non-ObjectServer events can take a wide variety of forms, depending on the nature of the event source. For SQL database event sources, an event might be the contents of a row in a table. For a messaging system event source, an event might be the contents of a message.

Netcool/Impact monitors non-ObjectServer event sources using an event listener service. The event listener service passively receives events from the event source and then passes them to the policy engine for processing.

Student exercises



The Netcool/Impact data model

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercise* book and perform the exercises for this unit.

Review questions

1. What is a data source?
2. Describe data items.
3. When are dynamic links used?

Review answers

1. What is a data source?

Data sources contain the necessary host, port, and login information that is needed to connect to external data.

2. Describe data items.

Data items are elements of the data model that represent actual units of data that is stored in a data source.

3. When are dynamic links used?

Dynamic links use a specified method to link data items of the source data type to the data items of a target data type. They are used at the database level.

Summary

You now can perform the following tasks:

- Describe a Netcool/Impact solution
- Create data sources, data types, and data items
- Describe the purpose of links
- Describe event sources

Unit 4 Policies

IBM Training

IBM

Policies

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes the purpose of a policy and how policies work. It also covers the basics of creating a policy in Netcool/Impact.

Reference: *IBM Tivoli Netcool/Impact Policy Reference Guide*

Objectives

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

- Create a policy
- Describe basic Netcool/Impact policy language (IPL) and JavaScript constructs
- Describe built-in variables
- Describe action functions
- Apply the length, Eval, extract, and parser functions in your policy
- Describe function libraries

Lesson 1 Netcool/Impact policy structure

IBM Training

IBM

Lesson 1 Netcool/Impact policy structure

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Policies overview

- Policies consist of sequential programming instructions that act upon event data
- Policies process events queried from the ObjectServer
- The policies that are used depend on the restriction filter defined in the event reader
- A policy instance acts on a single event from the ObjectServer through the event reader
- Policies instruct Netcool/Impact to perform a wide variety of actions, such as these examples:
 - Send and receive email
 - Send and receive instant messages
 - Communicate with external systems, devices, and applications

A policy consists of standard sequential programming steps. You can think of it as a series of conditional tests with subsequent actions.

Netcool/Impact policy language and JavaScript

- The Impact policy language (IPL) or JavaScript can be used to write policies in Netcool/Impact
- IPL is a scripting language similar in syntax to programming languages like C/C++ and Java
- As in other programming languages, users can define IPL variables and functions
- JavaScript is a scripting programming language that is commonly used to add interactivity to web pages
- JavaScript can also be used in browser environments
- JavaScript uses the same programming concepts that are used in IPL to write policies, though there may be syntax differences

Netcool/Impact policy language and JavaScript

A policy contains a set of statements that are written in the Netcool/Impact policy language, or IPL. Each statement is an instruction that describes a task for Netcool/Impact to perform.

Instructions can be for high-level tasks or low-level tasks:

- An example of a high-level task is sending an event to the Netcool/OMNIbus ObjectServer.
- An example of a low-level task is storing a value in the internal *MyDate* variable.

Other kinds of instructions cause Netcool/Impact to perform different tasks when different conditions are met, or to loop through a set of tasks.

For more information about JavaScript syntax, see the following website:

<http://www.w3schools.com/js/default.asp>

IPL basic constructs

- IPL provides the following set of constructs:
 - Data types
 - Built-in variables
 - Control structures
 - Functions
- IPL uses the operators shown in the following table:

+	-	*	/	%	
&&		^	&	==	!=
<	>	<=	>=	!	LIKE

IPL basic constructs

The IPL has a set of operators that you can use for the following purposes:

- Assigning and retrieving values from variables
- Performing mathematical operations
- Comparing values
- Concatenating strings

The IPL also defines a set of operators that you can use to perform bitwise operations.

IPL data types

- The policy-level data type is a different entity than the data types that are part of the data model
- Netcool/Impact policy language uses two categories of data types:
 - Simple data types
 - Integer
 - Float
 - String
 - Boolean
 - Date
 - Complex data types
 - Array
 - Context
 - Data item
 - Event container

IPL data types

The IPL has two sets of data types: simple and complex.

The simple data types are as follows:

- Integer
- Float
- String
- Boolean
- Date

The complex data types are as follows:

- Context
- Array
- Data item
- Event container

Simple data types

- Simple data types represent a single value
- IPL supports the following simple data types:
 - Integer
 - Whole numbers without a decimal point
 - Float
 - Numbers with a decimal point
 - Example: 1.234 and 12.345
 - String
 - Characters up to a length of 4 Kb
 - Enclosed in single or double quotation marks
 - Boolean
 - Represents a value of true or false
 - Use the True or False keywords
 - Date
 - A formatted string that represents a time or a calendar date
 - Required format YYYY-MM-DD HH:MM:SS.n, where *n* is a millisecond value
 - Enclosed in single or double quotation marks

Simple data types

Specify the ***n*** millisecond value in date fields when adding a data item to a data type or when updating a data item. If the date does not have a millisecond value, specify **0**. Specify the millisecond value when working with date fields in the Netcool/Impact GUI.

Simple data type example policy

The following policy example shows how to define simple data types:

```
Employee = NewObject();
//String
Employee.FirstName = "Joe";
Employee.LastName = "Example";
//Integer
Employee.Age = 34;
//Float
Employee.Height = 5.11
//Date
Employee.StartDate = "2006-07-31 11:12:13.4";
//Boolean
Employee.FullTime = True;
Log("Employee Data :", + Employee);
```

(StartDate=2006-07-31 11:12:13.4, LastName=Example, Height=5.11, FirstName=Joe, FullTime=true, Age=34)

Simple data type example policy

This example of a policy builds a new data type called **MyContext**, adds fields, and then assigns values to those fields.

Complex data types

- Complex data types represent sets of values
- IPL supports the following complex data types:
 - Context
 - Array
 - Data item
 - Event container

Complex data types

The Context data type has the following characteristics:

- It represents a heterogeneous set of data.
- It is stored in a set of variables called **member variables** contained inside the context.

The Array data type has the following characteristics:

- Like a context, an array also represents a heterogeneous set of data.
- Arrays are stored as unnamed elements rather than as member variables.

The Data item data type has the following characteristics:

- Data items can contain characters up to a length of 4 Kb.
- They are enclosed in single or double quotation marks.

The event container is a policy-level data type that represents an event.

Contexts

- A context represents a heterogeneous set of data
- A context is stored in a set of variables called member variables that are contained inside the context
- Member variables can be of any type, including other contexts
- Members variables are referenced using the dot notation
- Netcool/Impact provides a built-in context called the policy context
- The policy context is created automatically whenever a policy launches

Netcool/Impact provides a built-in context, called the **policy context**, that is created automatically whenever it opens the policy. The policy context contains all of the variables used in the policy, including built-in variables.

Context example

- The following example shows how to create a new context:

```
PrblmTktAssign = NewObject();
PrblmTktAssign.Name = "John Smith";
PrblmTktAssign.Phone = "18001234567";
PrblmTktAssign.Severity = "2";
PrblmTktAssign.Status = "open";
Log(PrblmTktAssign.Name + "/" + PrblmTktAssign.Phone + "/" + PrblmTktAssign.Severity
+ "/" + PrblmTktAssign.Status);
```

- This example prints the following message to the policy log:

John Smith/18001234567/2/open

Context example

To create a context in a policy, use the `NewObject` function. The example in the slide shows how to create a new context.

Arrays

- An array represents a heterogeneous set of data
- Data is stored as unnamed elements rather than as member variables
- Array elements are referenced using square bracket notation
- Use of the braces notation is also supported.
Array values can be assigned as a comma-separated series of numeric, string, or Boolean literals
- IPL arrays use a zero-based indexing system
In zero-based indexing, the first element of an array is identified by an index value of 0 instead of 1

Like contexts, arrays also represent a heterogeneous set of data. The data in arrays, however, is stored as unnamed elements rather than as member variables. As with contexts, the elements can be of any type, including contexts and other arrays.

Array example

- The following policy example shows how to create a new array and how to assign and reference its elements:

```
//An Array Containing Three Elements
OpenTickets = {"John Smith" + 36, "Mary Jones" + 23, "Mark Kay" + 36};
Log("All Tickets" + " " + OpenTickets);

//Log Elements Individually
Log ("First and Last Tickets" + " " + OpenTickets[0] + " " + OpenTickets[2]);

//Assign a Single Array Element to a Variable
AssignedTicket = OpenTickets[0];
Log("First Ticket Assigned" + " " + AssignedTicket);

• This example prints the following messages to the policy log:
  ▪ Parser log: All Tickets [John Smith36, Mary Jones23, Mark Kay36]
  ▪ Parser log: First and Last Tickets John Smith36 Mark Kay36
  ▪ Parser log: First Ticket Assigned John Smith36
```

Array example

This example shows how to create a new array and reference its elements.

Data items

- A data item is a policy-level data type that is used to represent data items in the Netcool/Impact data model
- A data item consists of a set of named member variables
- The member variables have a one-to-one correspondence with fields in the underlying data source
- Member variables are referenced using dot notation
- Many IPL built-in functions return an array of data items:
 - GetByFilter
 - GetByKey
 - GetByLinks
 - GetHibernatingPolicies
 - GetScheduleMembers

Data items are elements of a Netcool/Impact data model that represent a single unit of data as defined by a data type. Internal data items are created individually in the Data Items viewer. External data items are created automatically when a policy references the data type to which they belong, by a lookup in the external database.

Data item example

- Data items store data by reference rather than by value
- Stored by reference means that a change to the value of a member variable in a data item immediately changes the value in the underlying data source
Either of the following methods is used:
 - Directly accessing it in the internal data repository
 - Sending an SQL UPDATE
- The following example shows how to assign and reference values in a data item:

```
MyCustomers = GetByKey("Customer", 12345, 1);  
Log (MyCustomers[0].Name + ", " + MyCustomer[0].Location);  
MyCustomers[0].Location = "New Location";
```

Data item example

You can use the GUI to modify data items as follows:

- View data items
- Create data items
- Edit data items
- Delete data items

Event containers

- The event container item is a policy-level data type that represents an event
- An event container consists of a set of named member variables
- The event container member variables have a one-to-one correspondence with fields in the underlying event source
- Member variables are referenced using dot notation
- A new event container can be created with the NewEvent function

Event containers

The **event container** is a policy-level data type that represents an event. Like contexts and arrays, an event container consists of a set of named member variables. In an event container, the member variables have a one-to-one correspondence with fields in the associated event source. As with contexts and arrays, you use the dot notation to reference the member variables.

Event container example

- The following example shows how to modify a Event Container field and return the updated event:

```
EventContainer.Node = "DB2_Prod1";  
ReturnEvent(EventContainer);
```

- The following example shows how to create a new event container and how to assign and reference its member variables:

```
MyEvent = NewEvent("OMNIbusEventReader");  
MyEvent.Node = "ORACLE_01";  
MyEvent.Summary = "System not responding to ping request";
```

Event container example

A call to the NewEvent function is used to create new event containers in policies. The example shows how to create a new event container and how to assign and reference its member variables.

Variables

Netcool/Impact policy language supports the use of two types of variables:

- Built-in variables
- User-defined variables

The IPL has two types of variables: built-in variables and user-defined variables.

Built-in variables

There are built-in variables that Tivoli Netcool/Impact automatically populates in certain circumstances:

- EventContainer
- DataItems
- DataItem
- Num

The built-in variables are EventContainer, DataItems, Num, and DataItem. These variables handle data retrieved from external data sources and event data.

The EventContainer variable

- EventContainer is a built-in variable of the event container data type that represents an incoming event
- When a policy is triggered, the event processor service creates a new EventContainer and populates its member variables with the field values in the event
- The event processor creates one new member variable for each field in the event and assigns it the field value
- Event field variables can be referenced using the dot notation or the @ notation

The EventContainer variable

The event container is a policy-level data type that represents an event. Like contexts and arrays, an event container consists of a set of named member variables. In an event container, the member variables have a one-to-one correspondence with fields in the associated event source.

As with contexts and arrays, the member variables are referenced using the dot notation or @ notation.

Dot notation example

- The following example shows how to use the dot notation to reference event field variables:

```
Log(EventContainer.AlertKey);  
Log(EventContainer.AlertKey + ":" + EventContainer.Summary);  
EventContainer.Summary = EventContainer.Summary + ": Updated by  
Netcool/Impact";
```

- The following example shows the use of the @ notation to reference event field variables:

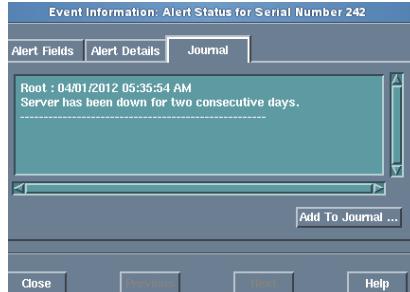
```
Log(@AlertKey);  
Log(@AlertKey + ":" + @Summary);  
@Summary = @Summary + ": Updated by Netcool/Impact";
```

Dot notation example

Event field variables can be referenced using the dot notation or the @ notation. The @ notation is a simpler way to refer to members of EventContainer without having to spell out the full name.

Event state variables

Netcool/OMNIbus Event Journal



Variable	Description
@JournalEntry	Set this field to add a new journal entry when you return an updated event to the event source. You specify the contents of the journal entry in single quotation marks. If you want to include newline or tab characters, you concatenate them separately with the string and enclose them in double quotation marks. You can only add journal entries to events when you update them from within a policy. You cannot add journal entries to new events.
@DeleteEvent	Set this field to True to delete the event when you return it to the event source.

Policies

© Copyright IBM Corporation 2016

Event state variables

EventContainer has a set of member variables that correspond to fields in the incoming event. It also contains two predefined member variables that you can use to specify the state of the event when you return it to the event source using the ReturnEvent function.

The following example shows how to add a new journal entry when you return an event:

```
// Set the @JournalEntry variable
@JournalEntry = 'LocalTime(GetDate()) \
+ "\r\n" + 'Server has been down for two consecutive days.';
// Return the event to the event source
ReturnEvent(EventContainer);
```

Netcool/Impact uses special rules for interpreting string literals assigned to JournalEntry. Text that is stored in JournalEntry must be assigned using single quotation marks, except for special characters such as \r, \n, and \t, which must be assigned using double quotation marks. If you want to use both kinds of text in a single entry, you must specify them separately and then concatenate the string using the plus (+) operator. To embed a line break in a journal entry, use an \r (backslash and letter r). The DeleteEvent variable is an event state variable that you use to specify that an event is to be deleted when it is sent back to the event source. Set the value of DeleteEvent to TRUE for an event to be deleted.

```
@DeleteEvent = True;
```

The DataItems variable

- **DataItems** is a built-in variable that stores an array of data items
- The DataItems variable is populated automatically by the functions that return data item arrays, such as these examples:
 - GetByFilter
 - GetByKey
 - GetByLinks
 - GetHibernatingPolicies
 - GetScheduleMember



Note: DataItems are also known as **OrgNodes**.

The DataItem variable

- DataItem is a built-in variable that references the first element in the DataItems array
- DataItem can be used as a shorthand for the following cases:
 - DataItems will contain only one element
 - You want to handle only the first element in the array
- **DataItem is equivalent to DataItems[0]**

The DataItem variable

DataItem is a built-in variable that references the first element in the DataItems array. You can use it to simplify cases in which you know that DataItems contains only one element. When you want to handle only the first element in the array, the DataItem is equivalent to **DataItems[0]**.

The Num variable

- **Num** is a built-in variable that stores the number of elements currently stored in the DataItems array
- The Num variable is supported in this version of Netcool/Impact for backward compatibility only
- The Num variable produces the same value as the Length() function
- Use of the Length function is recommended

The Num variable

Num is a built-in variable that stores the number of elements that are currently stored in the DataItems array. Use Num to count the number of data items returned by a function like GetByFilter, GetByKey, and GetByLinks. You can also use it to iterate through the DataItems array in a policy.

User-defined variables

- **User-defined variables** are variables that are defined when the policy is written
- Initialization is not required for variables used to store single values
- Call the NewObject function for context variables
- Call the NewEvent function for event container variables
- The member variables in contexts and event containers do not require initialization

User-defined variables

User-defined variables are variables that you define when you write a policy. You can use any combination of letters, numbers, and underscores as variable names, but the first character must be a letter. User-defined variables are used to store values during the lifetime of a policy in the same way that you use variables in other programming languages.

User-defined variables example

The following example shows how to create and reference user-defined variables:

```
//New User-defined Variables
MyInteger = 1;
MyFloat = 123.4;
MyBoolean = True;
MyString = "Event Summary";
//New Context
MyContext = NewObject();
MyContext.Status = MyBoolean;
//New Event
MyEvent = NewEvent();
MyEvent.Summary = MyString;
```

The first seven statements assign the value on the right side of the equal sign to the value on the left. The Log function writes the value of MyInteger and MyEventSummary to the log.

Control structures

- Control structures are Boolean expressions that execute until a specific condition is satisfied
- Usually a numeric counter is used to limit the number of times the expression and associated statements execute
- Netcool/Impact policy language supports the following control structures:
 - If
 - While

Control structures

The IPL has two control structures: If and While.

- Use the If structure to perform branching operations.
- Use the While structure to loop over a set of instructions until a certain condition is met.

If statement logic

The If statement has the following syntax:

```
If (x == 0) { Log ("x equals zero");  
}  
ElseIf (x == 1) { Log ("x equals one");  
} Else {  
    Log ("x equals any other value.");  
}
```

If statement logic

This If statement shows three conditions:

- x equal to 0
- x equal to 1
- x equal to any other value

While statement logic

The While statement has the following syntax:

```
C = 10;  
While (C > 0) {  
Log ("The value of C is: " + C);  
C = C - 1;  
}
```

This While loop sets a counter, C = 10. While C is greater than 0, a log value is created showing the value of C. The statement C = C - 1 is a decrementer, reducing the value of C. Each time this While loop iterates, the value of C is reduced by 1. The loop ends when C = 0.

Lesson 2 Policy functions

IBM Training

IBM

Lesson 2 Policy Functions

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Policy function overview

- Action functions
 - Perform common, predefined actions
 - Pass variables back to the policy, commonly referred to as arrays of data items called DataItems (also known as OrgNodes)
- Parser functions
 - Perform the logical processing within the policy
 - Provide statements such as IF, ELSE, WHILE looping, error handling
- User-defined functions
 - Used to organize code within a policy
- Web services functions
 - Used with the Netcool/Impact Web services DSA
- SNMP functions
 - Used with the SNMP DSA
- Java policy functions
 - Using Java policy functions, a policy can run a Java program or call any Java method if their Java classes are available from the Netcool/Impact runtime class path
- Local transactions
 - Use local transactions in a policy if you want to use more than one SQL operation to be treated as a single unit of work

Action functions

- Action functions accept information as input and return a response
- Responses are returned as an array called DataItems
- Responses can also be saved to another name
- Functions can also be used to execute other actions without returning responses

Action functions

Action functions are built-in functions that provide the core operation of Netcool/Impact policies. You use action functions to do the following tasks:

- Add, update, delete, and retrieve data items from data types
- Add, update, and delete events from event sources
- Send email
- Remotely run commands
- Run database functions and stored procedures

Action function example

This example shows a typical call to a function:

```
DataType="Admin";
Filter="Username LIKE 'super'";
CountOnly=False;
mySupers = GetByFilter(DataType,Filter,CountOnly);

Log("Returned " + Num + "rows.");
if (Num > 0) {
    Log ("The first was " + mySupers[0].Name);
}
```

Action function example

This Action function illustrates the use of the GetByFilter function. The GetByFilter function retrieves data items from a data type using a filter as the query condition.

Parser functions

- Parser functions are built-in functions that provide utility function for Netcool/Impact policies
- Utility function includes the following elements:
 - Basic type casting
 - String manipulation
 - Date and time conversions
- Parser functions are equivalent to the parser functions in Netcool/Impact Version 2.3 and earlier
- In most cases, the syntax for a parser function is the same as that for the equivalent function in the previous version

There is a long list of parser functions with a wide variety of capabilities. See the *IBM Tivoli Netcool/Impact Policy Reference Guide* for more details.

Parser function examples

- The Length function
- The Int function
- The Eval function
- The Extract function

Parser function examples

The next few slides examine these parser functions examples:

- The Length function
- The Int function
- The Eval function
- The Extract function

The Length function

- The Length function returns the number of elements or fields in an array or the number of characters in a string
- The Length function has the following syntax:

Integer = Length(Array | String)

The Length function

The Length function returns an **Integer** value, which you can assign to a variable.

Length function example

- The following example shows how to return the number of elements in an array:

```
MyNodes = GetByFilter("Node", "Location = 'New York'", False);  
NumNodes = Length(MyNodes);  
Log(NumNodes);
```

- This example prints the number of data items in the MyNodes array to the policy log

The Length function calculates the length of **MyNodes** in this example.

The Int function

- The Int function converts a float, string, or Boolean expression to an integer
- The function truncates any decimal fraction value associated with the number
- The Int function has the following syntax:

`Integer = Int(Expression)`

The Int function

The Int function converts float, string, or Boolean expressions to an integer. It truncates the decimal portion; for example, the value of **Int(1234.67)** is 1234.

Int function example

The following examples show how to convert floats, strings, and Booleans to an integer:

```
MyInt = Int(123.45);  
Log(MyInt);  
MyInt = Int(1234.5);  
Log(MyInt);  
MyInt = Int("456");  
Log(MyInt);  
MyInt = Int(False);  
Log(MyInt);
```

The policy log results are as follows:

```
Parser Log: 123  
Parser Log: 1234  
Parser Log: 456  
Parser Log: 0
```

The Int function converts floats, strings, and Booleans and stores them in **MyInt**.

The Eval function

- The Eval function evaluates an expression using the specified context
- The Eval function has the following syntax:

```
Integer | float | String | boolean = Eval(Expression, Context)
```

- Example:

```
MyContext = NewObject();  
MyContext.a = 5;  
MyContext.b = 10;  
MyResult = Eval ("a + b", MyContext);  
Log (MyResult);
```

- This example prints the following message to the policy log:

Parser log: 15

The Eval function evaluates a string and saves it to **MyResult**.

The Extract function

- The Extract function evaluates an expression using the specified context
- The Extract function has the following syntax:

```
String = Extract(Expression, Index, [Delimiter])
```

- Example:

```
MyString = "This is a test.";  
MyWord = Extract(MyString, 1, " ");  
Log (MyWord);  
MyString = "This|is|a|test.";  
MyWord = Extract(MyString, 3, "|");  
Log (MyWord);
```

- This example prints the following message to the policy log:

```
Parser log: is  
Parser log: test
```

The Extract function evaluates the **MyString** expression and stores the value in **MyWord**.

The UpdateEventQueue function

- The UpdateEventQueue function updates or deletes events in the event reader event queue
- The UpdateEventQueue function has the following syntax:

```
[Integer = ] UpdateEventQueue(EventReaderName, Filter, UpdateExpression, IsDelete)
```

The UpdateEventQueue function

One event queue per Netcool/Impact cluster is maintained for each event reader (generic or ObjectServer). Use the UpdateEventQueue action function when a policy might modify an event that has other related events in the event queue at the same time.

Updating and managing the event queue ensures that the following actions take place:

- Events that are processed have up-to-date data before they are processed
- Related events are deleted from the queue

UpdateEventQueue example

- The batch update example:

```
EventReaderName = "OMNIbusEventReader";
Filter          = "Node = 'Node Name'";
UpdateExpression = "Node = 'New Node Name'";
IsDelete        = false;
NumUpdatedEvents = UpdateEventQueue(EventReaderName, Filter, UpdateExpression,
IsDelete);
Log("Number of updated events: " + NumUpdatedEvents);
```

- The batch delete example:

```
EventReaderName = "OMNIbusEventReader";
Filter          = "Node = 'ORA_01'";
IsDelete        = true;
NumDeletedEvents = UpdateEventQueue(EventReaderName, Filter, null, IsDelete);
```

UpdateEventQueue example

Some of the reasons to use UpdateEventQueue include the following examples:

- Gaining the useful features of deduplication available for ObjectServer events
- Ensuring current data if the data in the events being processed has changed
- Deleting events in the queue because the first event to be processed has already made all necessary changes
- Keeping the event queue from growing over time because large event queues have performance implications

 **Syntax:** The syntax for UpdateEventQueue is as follows:

```
UpdateEventQueue (EventReaderName, Filter, UpdateExpression, IsDelete)
```

The UpdateEventQueue function returns an integer that specifies the number of events in the queue on which that action was taken.

Function libraries

- Function libraries are used to create a set of stored functions that can be called from any policy
- Function libraries have the following characteristics:
 - Encapsulate and reuse the custom code in your policies
 - Are similar in concept to libraries used by the C and C++ programming languages
 - Comprise a special type of policy that contains only user-defined functions

A function library is a good idea, especially if there are policy processes that are repeated.

Configuring function libraries

```
// NormalizeString trims whitespace, replaces the space character
// with an underscore and converts all characters to upper case

Function NormalizeString(StringToNormalize) {
    StringToNormalize = Trim(StringToNormalize);
    StringToNormalize = Replace(StringToNormalize, " ", "_");
    StringToNormalize = ToUpper(StringToNormalize);
}

// GetCustomersByNode returns an array of data items from
// a data source, where each data item represents a customer

Function GetCustomersByLocation(Location, Customers) {
    Type = "Customer";
    Filter = "Location = '" + Location + "'";
    CountOnly = false;
    Customers = GetByFilter(Type, Filter, CountOnly);
}
```

Parameters in the functions can be used as both input and output variables. The example shows a function library. This library is a policy named UTILS_LIBRARY.

Calling function libraries

- To call a function library, use following syntax:

```
function_library.function_name(param, [param ...])
```

function_library : name of the library policy

function_name : the name of the function

param : function arguments

- To call functions in the library named UTILS_LIBRARY, use this example:

```
// Normalize location string
UTILS_LIBRARY.NormalizeString(Location);

// Get customers at the specified location
UTILS_LIBRARY.GetCustomersByLocation(Location, Customers);

// Print customer info to the policy log
Log(Customers);
```

The library name does not have to be explicitly referenced before calling its functions, as is required in programming languages like C and C++. The example shows how to call functions in the library named UTILS_LIBRARY. The functions are the same as the functions defined in the previous slide.

Lesson 3 Error handling and other policy topics

IBM Training

IBM

Lesson 3 Error handling and other policies

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Exceptions

- Exceptions include anticipated as well as unanticipated policy results
- Exceptions occur while the policy is running
- Impact Policy Language and JavaScript can be used to raise and catch exceptions within a policy
- Impact Policy Language and JavaScript can be used to handle Java exceptions that are raised internally
- The Raise keyword is used to raise an exception
- The following example shows the syntax for Raise:

```
Function MyFunction(Param1, Param2) {  
    If (Param1 < 0) {  
        Raise IntOutOfRangeException("Value of Param1 must be greater than 0");  
    }  
}
```

In this example, an exception called **IntOutOfRangeException** is raised when Param1 is less than zero.

Handling exceptions

- The **handler** is a function with the following characteristics:
 - Is called each time a specific exception is raised at the policy level
 - Is a specific Java exception is raised by Netcool/Impact during the execution of a policy
- When an exception occurs, the exception handler runs, and the function no longer continues to run
- Declare exception handlers in advance of any position where they are triggered in a policy
- Insert error handlers at the beginning of a policy, before any other operations are specified
- The following example shows the syntax for exception handlers:

```
Handle ExceptionName { statements ... }
```

ExceptionName is the name of the exception raised using the **Raise** keyword. It can also be the name of the Java exception class that Netcool/Impact raises during the execution of the policy.

Sample exception handler IPL

- The following example shows how to handle policy-level exceptions using an exception handler

```
Handle IntOutOfRangeException {
    Log("Error: Value of parameter submitted to MyFunction is less than 0");
}
Function MyFunction(Param1, Param2) {
    If (Param1 < 0) {
        Raise IntOutOfRangeException("Value of Param1 must be greater than 0");
    }
}
```

- The following example shows how to handle Java exceptions using exception handlers

```
Handle java.lang.NullPointerException {
    Log("Null pointer exception in policy.");
}

Handle java.lang.Exception {
    log("ErrorMessage: " + ErrorMessage);
    MyException = javaCall("java.lang.Exception",ExceptionMessage, "getCause", null);
    log("MyException is " + MyException);
    MyException = javaCall("java.lang.Exception",MyException, "getCause", null);
    log("MyException again is " + MyException);
}
```

Sample exception handler IPL

The policy in the first example shows how to handle the `InOutOfRangeException` when it is raised in a policy. The **Handle** statement is declared before the **Raise** statement in the policy.

The second policy shows how to handle `java.lang` exceptions. Netcool/Impact policy processing generate Java language exceptions. Java language exceptions create an `ExceptionMessage`, which can be used for user-defined error processing.

Sample exception handler JavaScript

- The following example shows how to handle policy-level exceptions using an exception handler

```
try {
    MyFunction(Param1, Param2);
} catch(e) {
    if (e == "IntOutOfRangeException") {
        Log("Error: Value of parameter submitted to MyFunction is less than 0");
    }
}

function MyFunction(Param1, Param2) {
    If (Param1 < 0) {
        throw "IntOutOfRangeException";
    }
}
```

- The following example shows how to handle Java exceptions using exception handlers

```
try {
    ...code that is running...
} catch(e) {
    if (e.javaException instanceof java.lang.NullPointerException) {
        Log("Null pointer exception in policy.");
    }
    if (e.javaException instanceof java.lang.Exception) {
        log("ErrorMessage: " + ErrorMessage);
        MyException = javaCall("java.lang.Exception",ExceptionMessage, "getCause", null);
        log("MyException is " + MyException);
        MyException = javaCall("java.lang.Exception",MyException, "getCause", null);
        log("MyException again is " + MyException);
    }
}
```

Sample exception handler JavaScript

These two JavaScript policies accomplish the same tasks that the IPL policy accomplishes. The syntax is slightly different.

The DefaultExceptionHandler

The DefaultExceptionHandler policy has the following characteristics:

- Is a Netcool/Impact predefined policy
- Handles failed events if the policy failure is not handled in the policy
- Prints a log of the events that failed to execute

```
log("DefaultExceptionHandler got the event: " + EventContainer);
```
- Has the option of customized error handling policies that you can configure

The *Netcool/Impact Solution Guide* provides additional information about customizing the exception handler.

The failed event exception handler

- The FailedEventExceptionHandler policy:
- Is a Netcool/Impact predefined policy
- Uses the FailedEvent data type, together with the ReprocessedFailedEvents policy to provides a way to deal with failed events that are passed from the ObjectServer
- Prints a log of the events that failed to execute
- The FailedEvent data type can be used to re-create the EventContainer and send it back to the original policy that caused the error
- FailedEvent includes four fields:
 - Key
 - EventContainerString
 - Policy Name
 - EventReader name
- ReprocessFailedEvent is predefined policy that reprocesses failed events

The failed event exception handler

Run PolicyActivatorService at regular intervals to deal with failed events. The FailedEvent policy is as follows:

```
***** {COPYRIGHT-TOP-RM} ***
* Licensed Materials - Property of IBM
* "Restricted Materials of IBM"
* 5724-S43
*
* (C) Copyright IBM Corporation 2008, 2011. All Rights Reserved.
*
* US Government Users Restricted Rights - Use, duplication, or
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
***** {COPYRIGHT-END-RM} ***
param=NewObject();
param.KEY = EventContainer.KeyField;
param.EventContainerString=EventContainerString;
param.PolicyName= PolicyError.PolicyName;
param.EventReaderName = EventReaderName;
addDataItem("FailedEvent", param);
```

Local transactions

- Local transactions are used in a policy if you want to use more than one SQL operation to be treated as a single unit of work
- A typical template of a policy that uses local transactions:

```
Handle com.micromuse.response.action.TransactionException {
    Log(" Transaction Failed " + ErrorMessage);
    RollbackTransaction();
}

SQL_Operation_1();
...
BeginTransaction();
...
SQL_Operation_2();
SQL_Operation_3();
...
CommitTransaction();
...
SQL_Operation_4();
```

Some basic knowledge of SQL syntax and function is helpful when working with local transactions. The **Commit** statement completes a unit of work, essentially locking in all changes since the last **Commit**. When a **Rollback** occurs, all changes within the unit of work are undone, and a new **Commit** does not occur.

Local transactions considerations

- Local transactions should ideally combine SQL operations to a single database
- Avoid doing operations inside the transaction block that are not related to the actual SQL operations
- Do not rely solely on com.micromuse.response.action for every exception
For example: Use the general java.lang.Exception and call RollbackTransaction()

```
Handle com.micromuse.response.action.TransactionException {
    Log(" Transaction Failed " + ErrorMessage);
    RollbackTransaction();
}

Handle java.lang.Exception {
    Log("Policy Execution Failed" + ErrorMessage);
    // If there is no transaction, this won't do anything
    RollbackTransaction();
}
```

Although certain actions are possible, they can cause inefficient policy processing.

Clear cache

- Netcool/Impact stores information about data types in a system-level data type named Types
- Each data item in Types represents a data type that is defined in the system
- To clear a data type cache:
 1. Call the GetByFilter or the GetByKey function
 2. Retrieve the data item from Types that corresponds to the data type
 3. Set the **clearcache** member variable associated with the data item
- Clear cache example:

```
DataType = "Types";
Key = "User";
MyTypes = GetByKey(DataType, Key, 1);

MyTypes[0].clearcache = True;
```

Continuation character

- The line continuation character in IPL and JavaScript is the backslash (\)
 - This character is used to indicate that the code on a subsequent line is a continuation of the current statement
 - The line continuation character helps format policies so that they are easier to read and maintain
 - Line continuation characters cannot be used inside a string literal as specified with enclosing quotation marks
- This usage is not allowed and an error is reported during processing

Continuation character

The following example shows the use of the line continuation character:

```
Log("You can use the line continuation character" + \
    "to format very long statements in a policy.");
```

Student exercises



Policies

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What is IPL?
2. What is the difference between DataItem and DataItems?
3. What two types of functions does Netcool/Impact support?

Review answers

1. What is IPL?

Netcool/Impact policy language

2. What is the difference between DataItem and DataItems?

*DataItem is a built-in variable that references the first element in the DataItems array.
DataItems is a built-in variable that stores an array of data items.*

3. What two types of functions does Netcool/Impact support?

User-defined functions and built-in functions

Summary

You now can perform the following tasks:

- Create a policy
- Describe basic Netcool/Impact policy language (IPL) and JavaScript constructs
- Describe built-in variables
- Describe action functions
- Apply the length, Eval, extract, and parser functions in your policy
- Describe function libraries

Unit 5 Services

IBM Training

IBM

Services

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit focuses on controlling the operation of policies within Netcool/Impact. These topics are important in larger, more complex environments.

References: *IBM Tivoli Netcool/OMNIbus Administration Guide*

IBM Tivoli Netcool/Impact DSA Reference Guide

IBM Tivoli Netcool/Impact Solutions Guide

Objectives

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

- Describe the Netcool/Impact services
- Describe the OMNIbus event reader
- Describe the OMNIbus event listener
- Describe the policy logger

Services

- **Services** are subcomponents that you can run and control from within Netcool/Impact and from the command line
- Services perform many of the functions associated with the Impact server, including monitoring event sources, sending and receiving email, and triggering policies
- Event readers and event listeners are services
- To set up services, follow these steps:
 1. Determine what service function is necessary for the solution
 2. Create and configure the required services
 3. After you configure the services, you can start and stop them
 4. Set up messages to be logged to file, if you want

Services

Generally, you set up services once, when you first design your Netcool/Impact solution. After that, you do not need to actively manage the services unless you change the solution design.

Types of services

- Internal services are defined by Netcool/Impact to control the standard processes of the application
- User-defined services are defined per project for use in specific policies

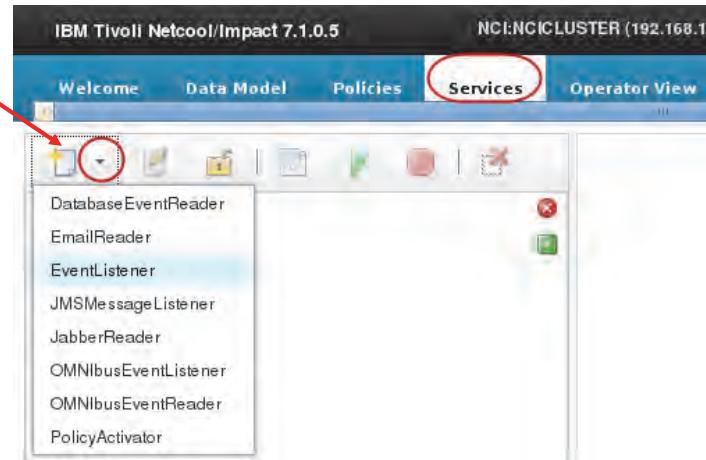
Types of services

There are two types of services in Netcool/Impact:

- **Internal** services coordinate the tasks that Netcool/Impact performs, such as these examples:
 - Receiving events from the ObjectServer and other external databases
 - Executing policies
 - Responding to and prioritizing alerts
 - Sending and receiving email and instant messages
 - Handling errors
- **User-defined** services are services that you can create with a unique name for use within a specific policy or a specific project.

Services: Default

Create New Service drop-down list



Services

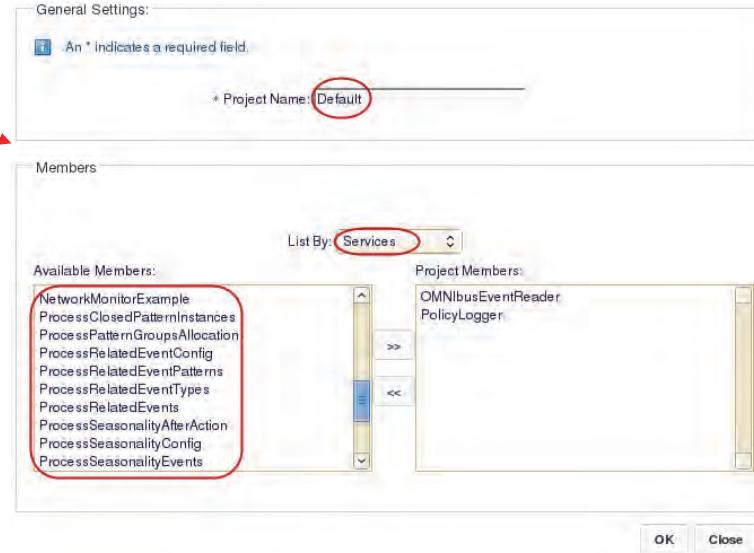
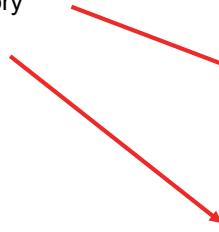
© Copyright IBM Corporation 2017

Services: Default

You can view the user-defined services for a project in one place. To create a user-defined service, in the navigation tree, expand **System Configuration > Event Automation**, and click **Services** to open the **Services** tab. Click the **Create a New Service** icon to view the drop-down list of available services.

Services: Global

Edit the project and add services from the Global Repository



Services

© Copyright IBM Corporation 2017

Services: Global

Click **Default** in the tool bar and select **Edit Current Project. List By <Services>** to see all the available services.

Netcool/Impact predefined services

Examples of predefined services:

- Event processor
- Email sender
- Hibernating policy activator
- Policy logger
- Command-line manager

Netcool/Impact predefined services

Predefined services are services that are created automatically when Netcool/Impact is installed. Predefined services can be configured, but new instances of the predefined services cannot be created and existing services cannot be deleted.

Netcool/Impact internal and user-defined services

Examples of internal and user-defined services:

- CommandExecutionManager
- CommandLineManager
- DefaultJabberReader
- DefaultPolicyActivator
- EmailSender
- EventProcessor
- HibernatingPolicyActivator
- ImpactDatabase
- **JMSMessageListener***
- JabberService
- MWMAactivator
- **OMNIbusEventReader***
- PolicyLogger
- SelfMonitoring

Netcool/Impact internal and user-defined services

You can create internal and user-defined services in the following ways:

- By using the defaults that are stored in the Global Repository
- By selecting them from a list in the **Services** tab

The services can be added to projects as project members.

Enable the default services rather than configuring new services, or in addition to creating new services.

The OMNIbus event reader

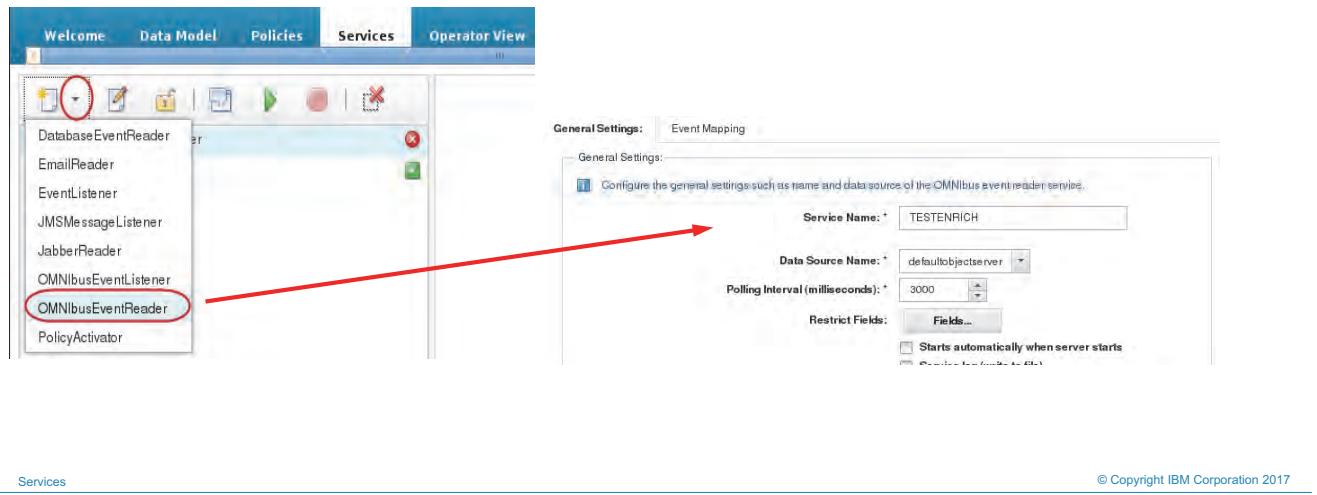
- **OMNIbus event readers** are services that monitor a Netcool/OMNIbus ObjectServer event source
- **ObjectServer events** are alerts stored in the ObjectServer database
- The ObjectServer stores alerts as rows in the **alerts.status** table
- Event readers are the most important Netcool/Impact services

The OMNIbus event reader

When an **OMNIbusEventReader** discovers new, updated, or deleted alerts in the ObjectServer, it retrieves the alerts and sends them to an event queue. The events are held in the event queue until the event processor handles them.

Configuring the OMNIbusEventReader

1. Select **OMNIbusEventReader** in the **Services** list
2. Click the **General Settings** tab and provide the settings information



Services © Copyright IBM Corporation 2017
Configuring the OMNIbusEventReader

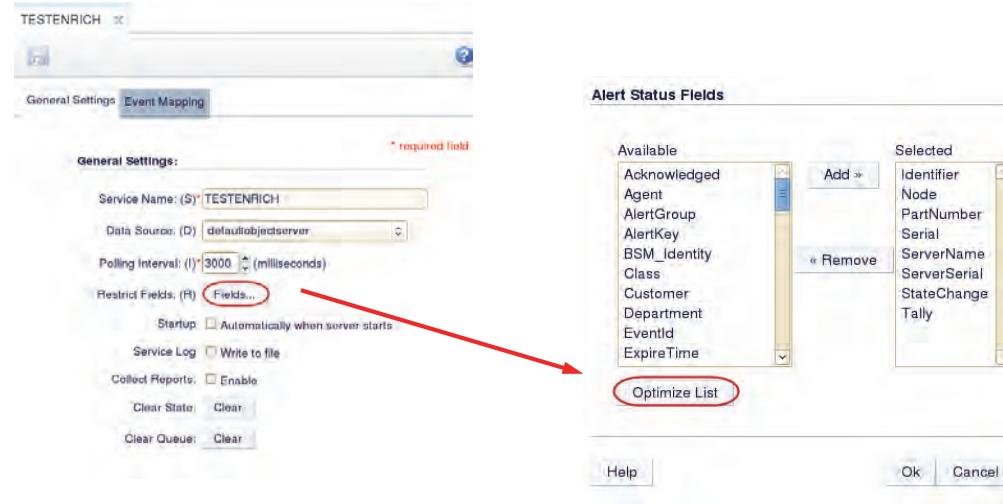
You can configure the following **OMNIbusEventReader** service fields:

- **Service Name:** A unique name, which identifies the service.
- **Data Source:** Select an OMNIbusObjectServer data source.
- **Polling Interval:** The polling interval is set to 3000 milliseconds (3 seconds) by default.
- **Restrict Fields:** Selects which fields from the ObjectServer to load into Netcool/Impact.
- **Startup:** Automatically starts the event reader when the Netcool/Impact server starts.
- **Service Log:** Starts logging to file.
- **Collect Reports:** Creates reports on the performance of the event reader.
- **Clear State:** The Serial and StateChange information that is stored for the event reader is reset to 0.
- **Clear Queue:** Clears unprocessed events.

The OMNIbusEventReader service is specially designed and optimized to communicate directly with Netcool/OMNIbus. For more detailed information on setting up the OMNIbus Object Server, see the *Netcool/OMNIbus Administration Guide*.

General Settings: Restrict Fields

You use **Restrict Fields** to limit retrieved fields from the data source

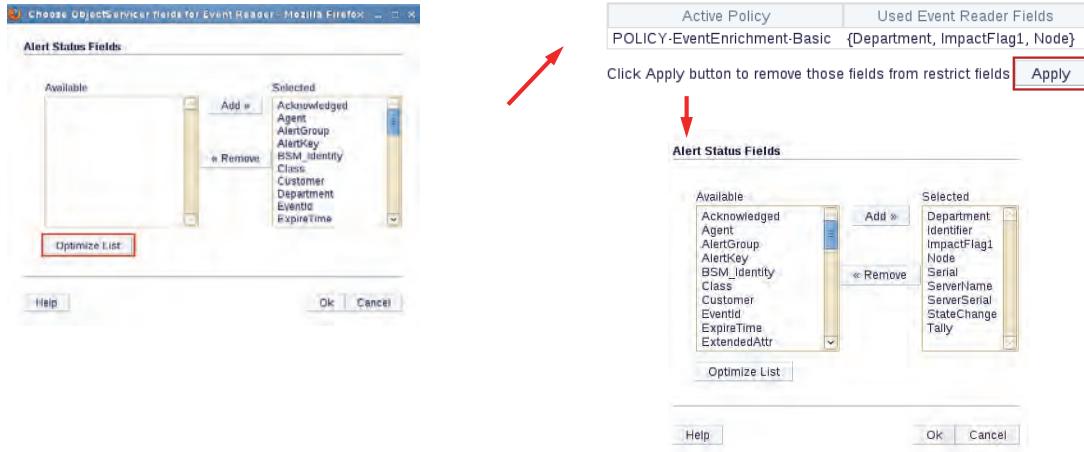


General Settings: Restrict Fields

Use the **Restrict Fields** option to specify which event fields to retrieve from the ObjectServer. By default, all fields are retrieved in the **alerts.status** table and placed in the **event.container**.

General Settings: Optimize List

Optimize List automatically selects fields used in the policy



Services

General Settings: Optimize List

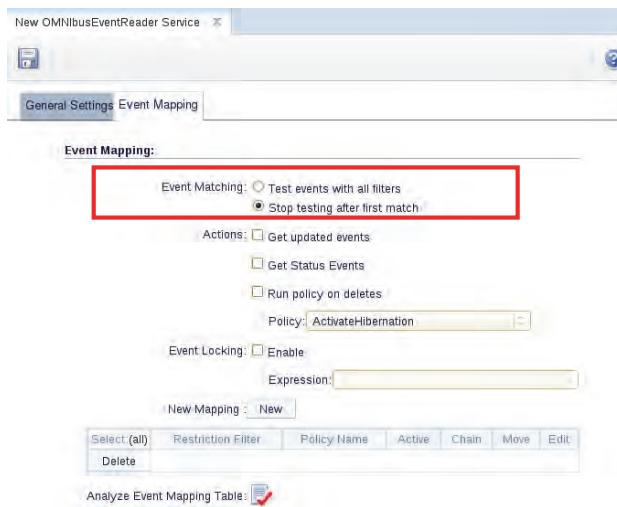
© Copyright IBM Corporation 2017

To improve OMNIbus event reader performance and reduce the performance impact on the ObjectServer, configure the event reader to retrieve only the fields that are used in the policies defined to this service. You can manually add or remove fields from the **Selected** list. **Optimize List** can make the selections automatically. You can view the list after you save the service.

Event Mapping: Event Matching

There are two options available for event matching:

- Test events with all filters
- Stop testing after first match



Event Mapping: Event Matching

This service can contain multiple restriction filters, each one triggering a different policy from the same event stream, or it can trigger a single policy. **Event Matching** determines whether a policy is run for an event if it matches more than one filter.

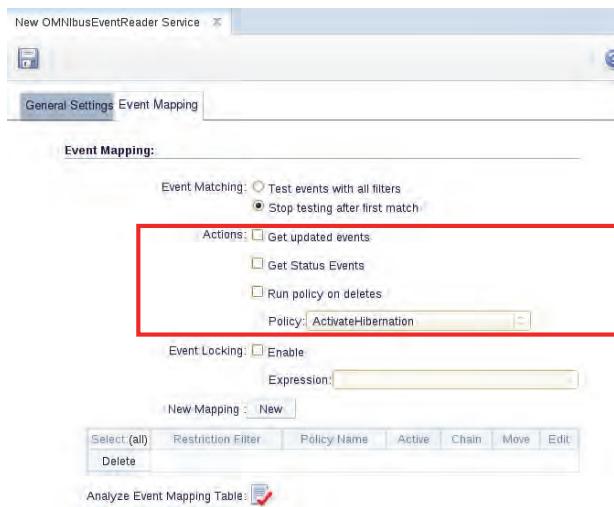
Test with all filters: Test events with all filters and run any matching policies. If an event matches more than one filter, all policies that match the filtering criteria are triggered.

Stop testing after first match: The event tests only with the first filter it matches.

Event Mapping: Actions

There are three **Action** options:

- Get updated events
- Get Status Events
- Run policy on deletes



Services

© Copyright IBM Corporation 2017

Event Mapping: Actions

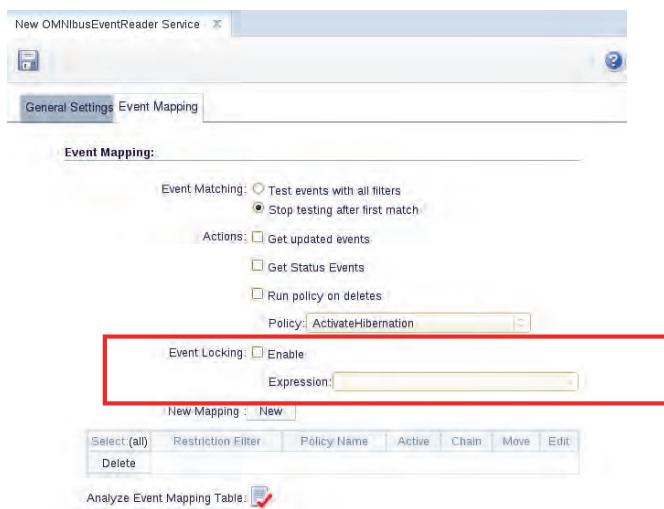
Three **Actions** options are available:

- **Get updated events:** Select to receive updated events and new events from the ObjectServer. The **Get updated events** option, when selected, requires the event reader to check against the @StateChange value to see whether anything changed since the last data source poll.
- **Get Status Events:** Select to receive the status events that the Self Monitoring service inserts into the ObjectServer. With the **Get Status Events** option, the event reader can process events with a Class of Impact (10500) value. Otherwise, these events are omitted from processing.
- **Run policy on deletes:** Select if you want the event reader to receive notification when alerts are deleted from the ObjectServer. Then select the policy that you want to run when notification occurs from the Policy list.

Normal operation of the event reader causes it to select events by matching events in which the @Serial value is greater than the last @Serial value found.

Event Mapping: Event Locking

Turn on **Event Locking** by selecting the **Enable** check box



Services

© Copyright IBM Corporation 2017

Event Mapping: Event Locking

Select **Event Locking** if you want to use event order locking and type the locking expression in the **Expression** field. With event locking enabled, if more than one event exists with a certain lock value, then these events are not processed at the same time. These events are processed in a specific order in the queue. Use event locking when you want to prevent a multithreaded event processor from attempting to access a single resource from more than one instance of a policy running simultaneously.

The Locking expression consists of one or more alert field names. To lock on a single field, specify the field name, for example, **Node**. To lock more than one field, concatenate them with the plus (+) sign, for example: **Node+Severity**.

If the value of that field is the same in both events, then one event is locked, and the second thread must wait until the first one is finished.

Event Mapping: New Mapping

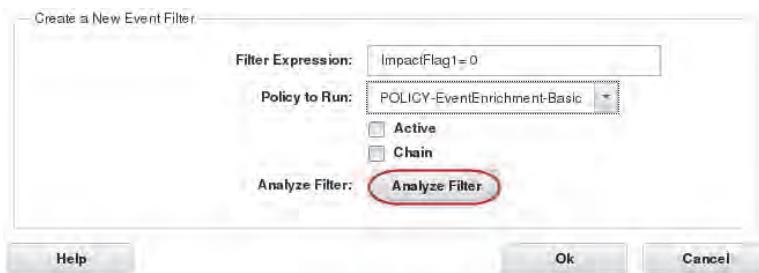
Click **New** to create a new event mapping



Event Mapping: New Mapping

Create a new event mapping by clicking **New**. Set a **Filter Expression** written as a SQL statement, and assign a **Policy to run** in the **Create New Mapping** window. The filter instructs the policy to execute whenever an incoming event matches the filter.

Analyze Filter



Services

© Copyright IBM Corporation 2017

Analyze Filter

The **Analyze Filter** option analyzes the **Filter Expression** to determine whether there are conflicts with any other filters that are defined. Using the **Event Matching** option, you can either run all the policies that match (more common) or run only the first policy that matches. With this option, you can perform the following tasks:

- Set up a type of case statement by selecting the first policy that can be processed.
- Create a default policy to be processed at the end with no conditions set.

Normal operation of the event reader causes it to select events by matching events in which the @Serial value is greater than the last @Serial value found. The **Get updated events** option, when selected, requires the event reader to check against the @StateChange value to see whether anything changed since the last data source poll. With the **Get Status Events** option, the event reader can process events with a Class of Impact (10500) value. Otherwise, these events are omitted from processing.



Note: Remember to make the event filter active by selecting the **Active** check box.

Filter analysis results

Filters to analyze:
Active filters Refresh

Filter Syntax Analysis Result:
No filter syntax error was found.

Filter Range Overlap Analysis Result:
NOTE: Your current event matching choice is 'Test events with all filters'.

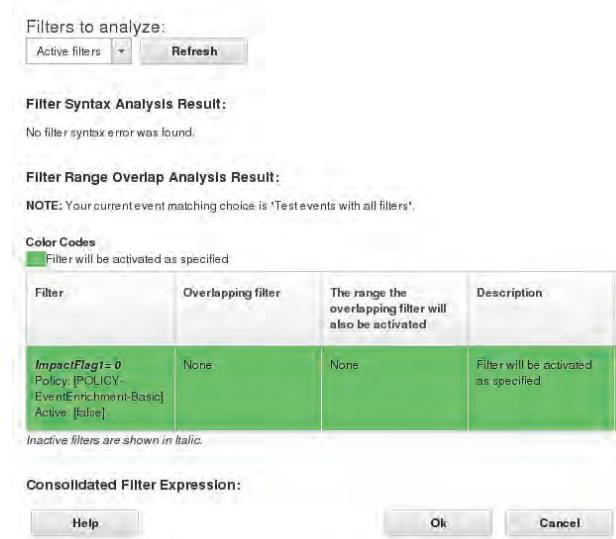
Color Codes
Filter will be activated as specified

Filter	Overlapping filter	The range the overlapping filter will also be activated	Description
<i>ImpactFlag1=0</i> Policy: [POLICY-EventEnrichment-Basic] Active: [false]	None	None	Filter will be activated as specified

Inactive filters are shown in *italic*.

Consolidated Filter Expression:

Help Ok Cancel



Filter analysis results

Netcool/Impact analyzes the restriction filters listed in the mapping table. The goal is to determine whether there are data contention issues or filter conflicts that will affect policy processing. The table is a color-coded map of the filters that are defined to the OMNIbusEventReader service. The map provides a clear visual indication of where any possible conflicts might exist.

Clicking the **Analyze Filter** icon in the **Create a New Event Filter** window displays the results of filter analysis. The table shows that only one filter is defined; therefore, there are no overlaps with other defined filters from other policies.

Running the OMNIbusEventReader

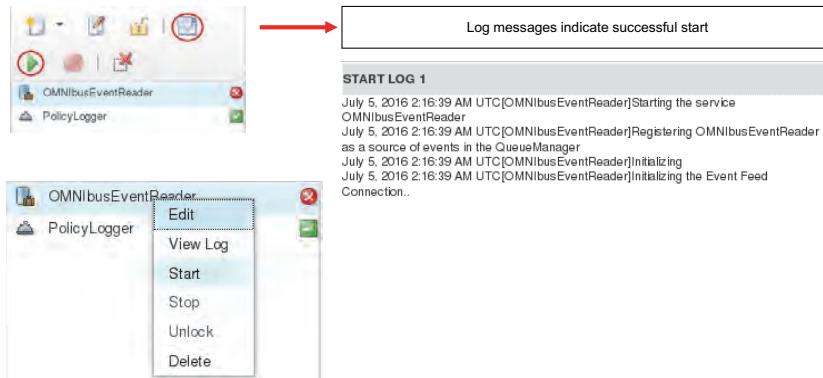
Click the **Start** icon

or

Locate
OMNIbusEventReader
in the **Services** tab

Right-click and select
Start

Running service



Running the OMNIbusEventReader

Use the default OMNIbusEventReader (TESTENRICH) to launch the enrichment policy.

Start the OMNIbusEventReader by one of the following methods:

- Clicking the **Start** icon in the **Services** tab
- Clicking the icon in the main display area of the OMNIbusEventReader

The log shows the event reader queries to the ObjectServer. Typically, as the event reader is processing, the log is refreshed continuously. If you want to stop scrolling to better view items in the log, click **Pause**.

Default logging

- Tivoli Netcool/Impact is configured to do basic logging upon installation
You can find basic logs for the Netcool/Impact server in this location:
\$NCHOME/impact/logs/
- You can set the location of the additional logs in the file:
\$NCHOME/impact/etc/NCI_server.props
- All other logging is based on the services associated with logging including policy execution service logging
- You set service logging from the service configuration screen for each service

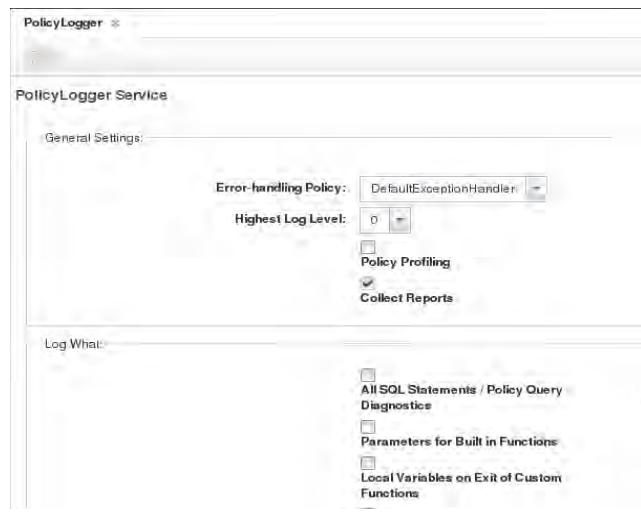
Default logging

This lesson covers the defaults of the logging function. Netcool/Impact automatically writes to the files found in the **\$NCHOME/logs** directory.



Note: **\$NCHOME** equates to **/opt/IBM/tivoli/impact** for this installation.

The policy logger



The policy logger

The PolicyLogger Service specifies an error-handling policy to activate when an error occurs during the execution of a policy. When configuring this service, select a policy that can handle the errors as they occur. This policy is normally **DefaultExceptionHandler**, unless you develop a special exception-handling policy.

Select a number value in the **Highest Log Level** field. When Tivoli Netcool/Impact processes a Log function in a policy, it evaluates the specified log level against the number that is selected for this field. If the level specified in this field is greater than or equal to the number specified in the policy Log statement, the message is recorded in the policy log.

Select or clear the check boxes in the **Log What** section. These options determine the amount of information that the service log shows.

- **All SQL statements:** Logs all SQL statements made by the server.

Select or clear the check boxes for the following options:

- **Pre-execution Action Module Parameters:** The values of all the parameters passed to a built-in action function before the function is called are added to the log.
- **Post-execution Action Module Parameters:** This option includes post-execution parameters in the log.
- **Policy Profiling:** If you want to view performance data about policies, you can use this feature to see how long it takes Netcool/Impact to process variable assignments and functions. It can also show how long it takes Netcool/Impact to process an entire function and the entire policy.
- **Service Log: Write to file:** This option writes the service log to a file.
- **Collect Reports:** This option starts report collection. You can view reports on the **Reports** tab.
- **Append Thread Name to Log File Name:** This option separates the log statements into different files, depending on the thread that is used in the server.
- **Append Policy Name to Log File Name:** This option separates the log statements into different files, depending on what policy is being run.

Log files

- ImpactServerName_EventReaderName.log
 - Shows the event reader fetching information (same as user interface)
- ImpactServerName_PolicyLogger.log
 - Shows policy debugging information (same as user interface)
 - Checks pre-execution and post-execution for objects passed or returned
 - Checks SQL statements for validity
- Service log options

The screenshot shows the 'Policy Logger Settings' interface. On the left, under 'Service Log:', there are three checkboxes: 'Service log (write to file)' (which is checked and circled in red), 'Append Thread Name to Log File Name', and 'Append Policy Name to Log File Name'. On the right, a button labeled 'Resulting Log Files ➔' leads to a list of log files: NCI_policylogger_POLICY-EventEnrichment-Basic.log, NCI_policylogger_TestPolicyEval.log, NCI_policylogger_TestPolicyExtract.log, and NCI_policylogger_TestPolicyLength.log.

Log files

These additional logs provide information specific to a particular service. Multiple log files can be created as follows:

- 1 log file for each policy
- 1 log file for each thread in the event processor
- 1 log file for each policy for each thread

By default, a log file for a single policy is created.

Expanding logging capabilities

- You can configure logging in Tivoli Netcool/Impact
- Logging is based on the publicly available log4j package available from the Apache Software Foundation
- You can set some logging configuration in the user interface
You set more fine-grained logging options in configuration files
- There are five basic levels of logging in the log4j package in the Netcool/Impact server:
 - DEBUG
 - INFO
 - WARN
 - ERROR
 - FATAL

Expanding logging capabilities

Configure Netcool/Impact logs to catch specific types of errors. Different logging levels are available and are used in conjunction with the PolicyLogger highest log level:

```
log(3, "Message");  
log(2, currentcontext());
```

Other services and optional service selections

- Database event readers
- OMNIbus event listeners
- Automatically start services
- Service log files

Database event reader

- Functions similarly to the OMNIbus event reader
- Queries any SQL database
- Replaces the generic event reader from previous releases

Database event reader

The database event reader was called the *generic event reader* in Netcool/Impact versions 4.0.2 and earlier. This service polls an SQL data source at intervals and retrieves rows from a table. It then converts the rows to event format and passes them to Netcool/Impact for processing. The data source can be any of the SQL data sources supported by Netcool/Impact. Conceptually, it is similar to the OMNIbus event reader, which polls the ObjectServer to get network fault events.

Database event reader General Settings

DatabaseEventReader Service

General Settings Event Mapping

General Settings:

Configure the general settings such as data source and data type of the Database event reader service.

Service Name: * DBService

Data Source Name: * HS_TRAIN

Data Type Name: * HS_Customer

Polling Interval (milliseconds): * 3000

Restrict Fields:

Starts automatically when server starts

Service log (write to file)

Clear State:

Clear Queue:

Services

© Copyright IBM Corporation 2017

Database event reader General Settings

Use the **General Settings** tab to configure the event reader service. Descriptions of the fields are as the same as for the OMNIbusEventReader with the following exceptions:

- **Service Name:** A unique name for the service
- **Data Source:** An external data source selected from the list
- **Data Type:** Populates when the data source is selected

Database event reader Event Mapping

The screenshot shows the 'Event Mapping' tab of the DatabaseEventReader Service configuration interface. It includes sections for 'Event Matching' (with options for 'Test events with all filters' or 'Stop testing after first match'), 'Action' (with checkboxes for 'Get updated events', 'Time Stamp Field: CUSTOMER', and 'Key Field: CUSTOMER'), and an 'Event Mapping Table' (with instructions to double-click or press ENTER to edit, and buttons for Delete Selection, New, Edit, Move Up, Move Down, and Quick Add).

Services

© Copyright IBM Corporation 2017

Database event reader Event Mapping

Use the event reader **Event Mapping** tab to configure mapping filters. You create filters by entering an SQL filter statement and assigning a policy to it. The filter instructs the policy to run whenever an event matches the filter. On this tab, the following tasks are available:

- Create a new filter and assign a policy to it
- Edit an existing filter
- Delete an existing filter
- Change the order of the filter list

Select policies to chain so that they run sequentially when triggered by an event reader service.

OMNIbus event listeners

- Netcool/Impact uses the OMNIbus event listener service to integrate with Netcool/OMNIbus and receive immediate notifications of fast-track events
- With the OMNIbus event listener, Netcool/Impact can analyze, correlate, and enrich data stored in OMNIbus in real time
- Netcool/OMNIbus 7.2.x or higher is required to configure this service

OMNIbus event listeners

The OMNIbus event listener service works specifically with Netcool/OMNIbus 7.2. or 7.3. It receives notifications through the Insert, Delete, Update, or Control (IDUC) channel.

You must create triggers in Netcool/OMNIbus before Netcool/Impact can receive accelerated events. For more information on OMNIbus triggers and accelerated event notification, see the *Netcool/OMNIbus Administration Guide*.

For more detailed information on setting up DSAs and event listener services, see the *Netcool/Impact DSA Reference Guide* and the *Netcool/Impact Solutions Guide*.

OMNIbusEventListener configuration

OMNIbusEventListener Service

General Settings:

Service Name: *

Data Source: ObjectServerForNOI

One or more Channels: * default

Event Matching:

Test events with all filters
 Stop testing after first match

Event Mapping Table:

Double click or press ENTER on a cell to edit; press SHIFT + ENTER to finish editing. Press New or Quick Add to add new mapping.

Restriction Filter	Policy Name	Active	Chain
--------------------	-------------	--------	-------

Services © Copyright IBM Corporation 2017

OMNIbusEventListener configuration

Use the Netcool/Impact GUI to create a new **OMNIbus Event Listener**. On the configuration screen, specify one or more policies to run when the OMNIbus event listener receives incoming events from Netcool/OMNIbus.

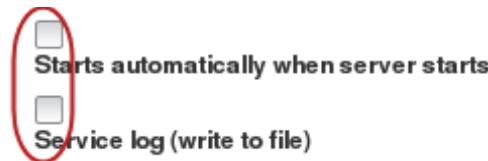
- **Service Name:** Type a unique name for the service.
- **Data Source:** Select a Netcool/OMNIbus version 7.2 or 7.3 ObjectServer data source.
- **Event Matching:**
 - **Test events with all filters:** Select this check box if you want to trigger all policies that match the filtering criteria when an event matches more than one filter
 - **Stop testing after first match:** Select this check box if you want to trigger only the first matching policy.
- **New Mapping:** Click to create a new event filter.
- **Analyze Event Mapping Table:** Click this icon to check the validity of event filters.
- **Startup:** Select to start the service automatically when the Netcool/Impact server starts.
- **Service Log:** Select this check box to write the service log to a file.

Start the service to establish a connection to the ObjectServer and subscribe to the IDUC channel to get notifications for inserts, updates, and deletes from OMNIbus.

Service startup and service logging

Optional service selections:

- Services can automatically start when the Netcool/Impact server starts
- Services can write output to a file



Service startup and service logging

You can set up services to start **Automatically when the server starts**. Services can also be enabled to send output to a file from within the user interface. Most have a **Service Log** check box, which sends the output from the service to a log file. The content is the same as displayed in the GUI. Log files are named based on the type of service.

Review questions

1. What types of services does Netcool/Impact provide?
2. Describe the OMNIbus event listener.

Review answers

1. What types of services does Netcool/Impact provide?

There are two types of services: internal and user-defined.

2. Describe the OMNIbus event listener.

The OMNIbus event listener service is a new feature in Netcool/Impact version 5.1.

Netcool/Impact uses the OMNIbus event listener service to integrate with Netcool/OMNIbus.

Summary

You now can perform the following tasks:

- Describe the Netcool/Impact services
- Describe the OMNIbus event reader
- Describe the OMNIbus event listener
- Describe the policy logger

Unit 6 The Enrichment policy

The Enrichment policy

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

One of the primary functions of Netcool/Impact is to provide event enrichment. This unit examines the event enrichment policy.

Objectives

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

- Describe Netcool/Impact Solution types
- Describe event enrichment policies
- Create a policy with the policy editor
- Launch a policy with the OMNIbusEventReader

Netcool/Impact solution types

Netcool/Impact solution types include these examples:

- Event gateway solutions

Event gateway is an implementation of Netcool/Impact where event information is sent from the ObjectServer to a third-party application for processing

- Event notification solutions

Event notification is the process by which Netcool/Impact monitors an event source for new events and then notifies an administrator or users when a certain event or combination of events occur

- X events in Y time solutions

X events in Y time is the process in which Netcool/Impact monitors an event source for groups of events that occur together and takes the appropriate action based on the event information

- Event enrichment solutions

Event enrichment is the process by which Netcool/Impact monitors an event source for new events, looks up information related to them in an external data source and then adds the information to them

Event enrichment solutions

An event enrichment solution consists of the following components:

- A data model that represents the data that you want to add to events
- An OMNIBus event reader service that monitors the event source
- Event enrichment policies look up information related to events and add information to them

Policy creation

You can create new policies in Netcool/Impact in the following ways:

- From scratch within the policy editor
- From saving an existing policy under a new name within the policy editor
- By using a policy wizard
- By using policy templates*
- By using the policy uploader

*Policy templates are documented in earlier releases

Policy creation

You can create new policies in the Policy Editor from an existing policy, by using a policy wizard, policy templates, and by using a policy uploader. Policy templates provide examples of commonly used policies that you can adapt to your own needs. Policy wizards present a series of windows that guide you through the policy creation process. Policy wizards are described in greater detail in later units. The policy uploader imports policies that are built outside the Netcool/Impact tool.

After you create a policy, it is stored in the **Policies** tab.

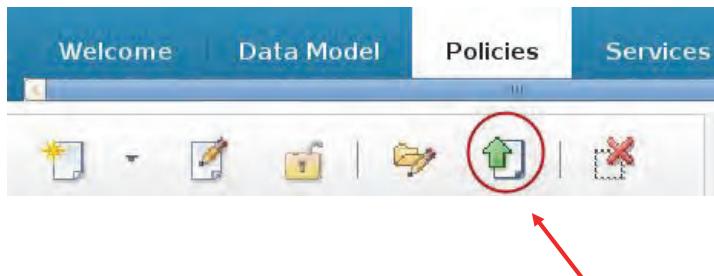
The Policies tab



The Policies tab

Policies consist of a series of function calls that manipulate events and data from supported data sources. A *policy* is a script that contains a set of instructions to automate alert management tasks. For example, policies can define the conditions for sending an email to an administrator or can send instructions to the ObjectServer to clear an event.

Importing policies



Click the **Upload a Policy File** icon

The import function is in the **Policies** tab. Click the **Upload a New Policy** icon to open the utility.

Policy uploader

Upload Policy

Select a policy or parameters file to upload.

Policy file (.ipl or .js)
Browse ... No file selected

Parameters file (.params)
Browse ... No file selected

Encoding

Upload **Close**

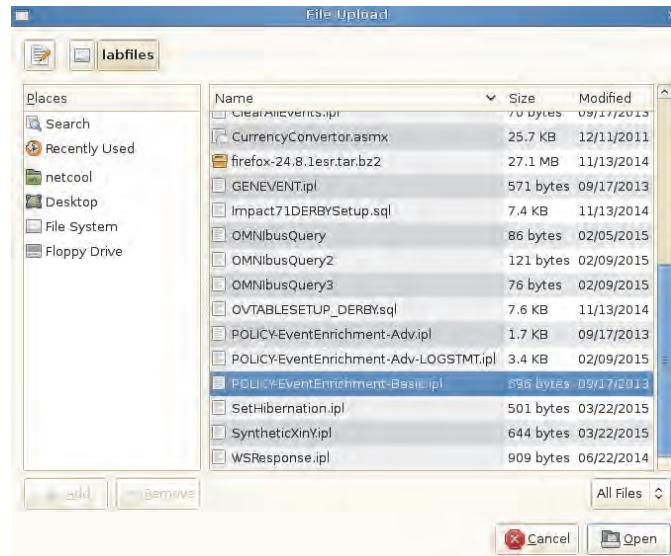
The Enrichment policy

© Copyright IBM Corporation 2017

Policy uploader

Provide the fully qualified path to a policy (.ipl or .js) file. Optionally, upload a parameters file if necessary. You can use the **Browse** button to locate the policy in the file system.

The File Upload window



The Enrichment policy

© Copyright IBM Corporation 2017

The File Upload window

When the selected policy shows in the uploader window, click the **Upload** button to import the policy. Although you can browse all file types, you can upload only files that are in the .ipl (or .js) file format. To upload a separate file that contains user parameters, select the **Upload policy user parameters file** check box. Then either type the file name or click **Browse** to find the file.

Updated policy upload window

Upload Policy

Select a policy or parameters file to upload.

Policy file (.ipl or .js)
 File to upload: POLICY-EventEnrichment-Basic.ipl

Parameters file (.params)
 No file selected

Encoding:

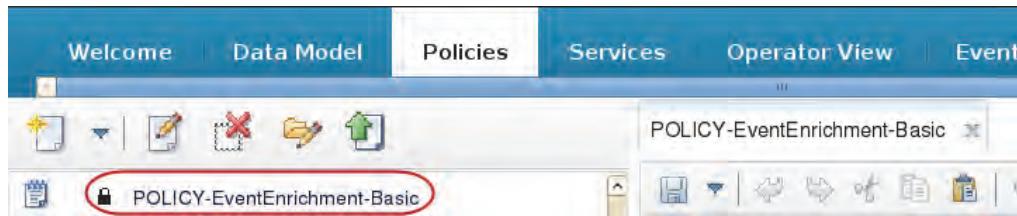
[The Enrichment policy](#)

© Copyright IBM Corporation 2017

Updated policy upload window

If a policy is successfully uploaded, it is shown in the **Policies** list. To view the policies in the policy list, you might have to refresh the browser.

Policies tab after the upload



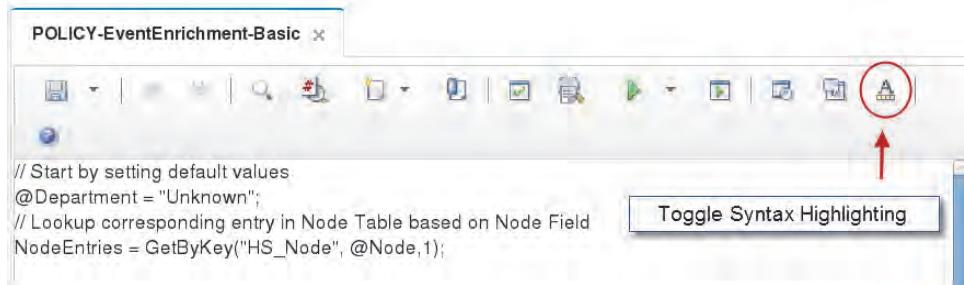
The Enrichment policy

© Copyright IBM Corporation 2017

Policies tab after the upload

You show and modify a policy that is uploaded into the Netcool/Impact product in the policy editor the same way as for any other policy. You can turn off policy highlighting by clicking the highlighting toggle icon.

The policy editor



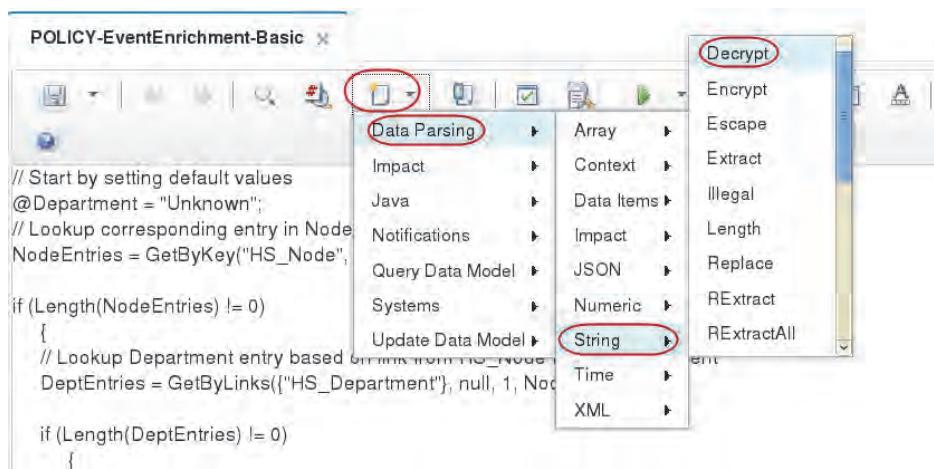
The policy editor

One way to add functions to a policy is by using the **Insert Function** icon. Netcool/Impact provides a library of predefined action functions so that a policy can perform the following tasks:

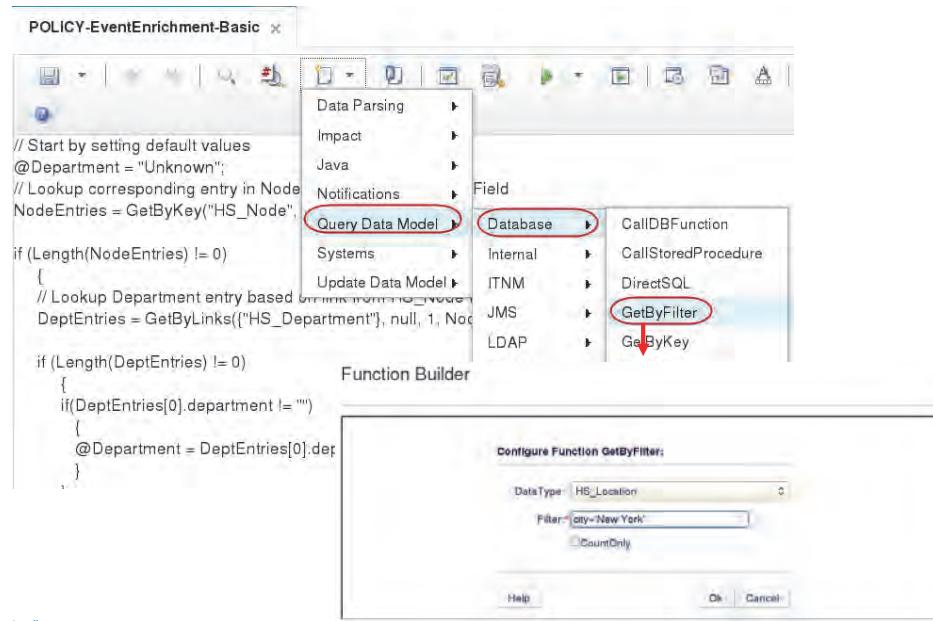
- Insert, update, and delete data in internal and external data sources
- Send email
- Activate other policies
- Manage hibernating policies
- Run external commands, scripts, and applications

Using the tool is useful if you do not know the syntax for a particular function. Also, you can add functions to the policy manually.

The Insert Function list



Configuring functions



The Enrichment policy

© Copyright IBM Corporation 2017

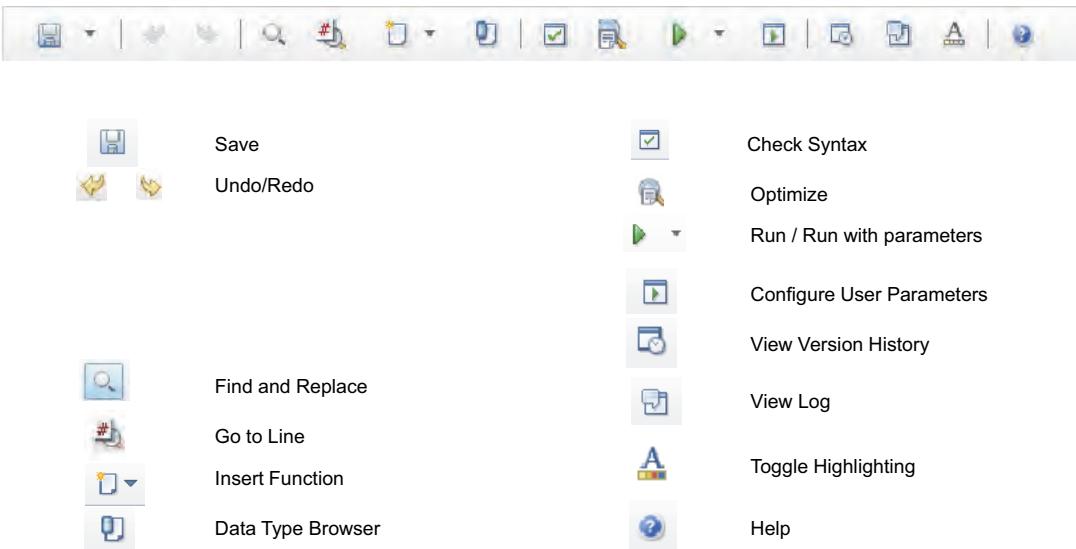
Configuring functions

Use the **Function Builder** tools to help define the parameters for predefined functions. You are automatically prompted for function parameters, and the function with parameters is inserted into the policy code at the end of the existing policy.

1. Select the action type from the list.
2. Click **Insert Function**.
3. Enter the necessary function parameters on the configuration screen.

Save the policy regularly when you make changes.

Policy editor toolbar functions



Policy editor toolbar functions

This is the current list of icons available in the policy editor. This page can be used as a reference.

- The **Save** icon saves the policy.
- The **Undo** icon restores your work to its state before your last action, such as add text, move, or delete. This function will undo only one level.
- The **Redo** icon restores your work to its state before you selected the **Undo** action.
- The **Find and Replace** icon searches for text and replaces it with text you specify.
- The **Go to Line** icon places the cursor at a line number that you select.
- The **Insert Function** icon starts the Function Builder tool.
- The **Data Type Browser** icon accesses a list of data types.
- The **Check Syntax** icon checks the policy for syntax errors. If there are errors, the error message locates the error by the line number. If there are no errors, message to that effect is displayed.
- The **Optimize** icon checks the syntax of the policy that you specified in the **Policy Name** field.
- After removing all syntax errors, click the **Run Policy** icon to ensure that the policy produces the intended result.
- The **Set Runtime Parameters** icon sets the runtime parameters for a policy.
- Click the **View Version History** icon to view the history of changes made to policies, and compare different versions of policies. This icon is disabled for new and drafted policies, and it

becomes active after the policy is committed to server. It is available with embedded Subversion only.

- The **View Log** icon opens the policy logger service log in the log viewer.

The event enrichment policy

```
// Start by setting default values
@Department = "Unknown";
// Lookup corresponding entry in Node Table based on Node Field
NodeEntries = GetByKey("HS_Node", @Node,1);

if (Length(NodeEntries) != 0)
{
    // Lookup Department entry based on link from HS_Node to HS_Department
    DeptEntries = GetByLinks(["HS_Department"], null, 1, NodeEntries);

    if (Length(DeptEntries) != 0)
    {
        if(DeptEntries[0].department != "")
        {
            @Department = DeptEntries[0].department;
        }
    }
}

// Return information back to ObjectServer
// Set Flag as to not re-process
@ImpactFlag=1;
ReturnEvent(EventContainer);
```

```
// Start by setting default values
log("Policy Start***");
log("Serial from the Event Container----->"+@Serial);
@Department = "Unknown";
log("Department set in the policy----->"+@Department);
log("Node from the Event Container-----> "+@Node);
// Lookup corresponding entry in Node Table based on Node Field
NodeEntries = GetByKey("HS_Node", @Node,1);
log("NodeEntries: Result of Lookup in HS_Node-----> "+NodeEntries);
if (Length(NodeEntries) != 0)
{
    // Lookup Department entry based on link from HS_Node to HS_Department
    DeptEntries = GetByLinks(["HS_Department"], null, 1, NodeEntries);
    log("Dept Entries: Result of lookup in HS_Department -----> "+DeptEntries);
```

The Enrichment policy

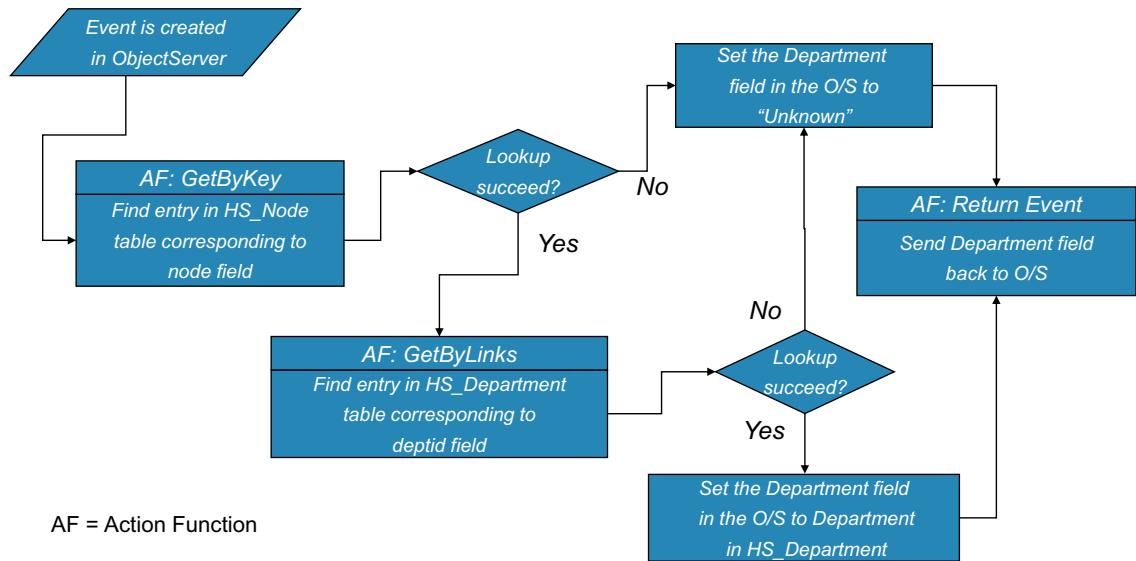
© Copyright IBM Corporation 2017

The event enrichment policy

The policy flow is as follows:

1. The event shows up in the Object Server.
2. The **GetByKey** function is used to find the event container node value in the HS_Node table.
3. The following actions occur, depending on whether there was a GetByKey match:
 - If there was a **GetByKey** match, the following actions occur:
 - ◆ The information in the HS_Node table is used to access the HS_Department table using the **GetByLinks** function.
 - ◆ If the **GetByLinks** lookup succeeded, the Department field from the HS_Department table is copied to the Department field in the Object Server, and the event container is returned
 - ◆ If the **GetByLinks** lookup failed, the **Department** field is set to **Unknown**, and the event container is returned to the Object Server.
 - If there was no **GetByKey** match, the **Department** field is set to **Unknown**, and the event container is returned to the Object Server.

The event enrichment policy logical flow



The event enrichment policy logical flow

Another way to better understand policy data flow is to use a logical flow diagram. The logical flow diagram focuses on the Action Functions (AF) used in the policy.

Launching policies for testing

There are several ways to launch a policy for testing:

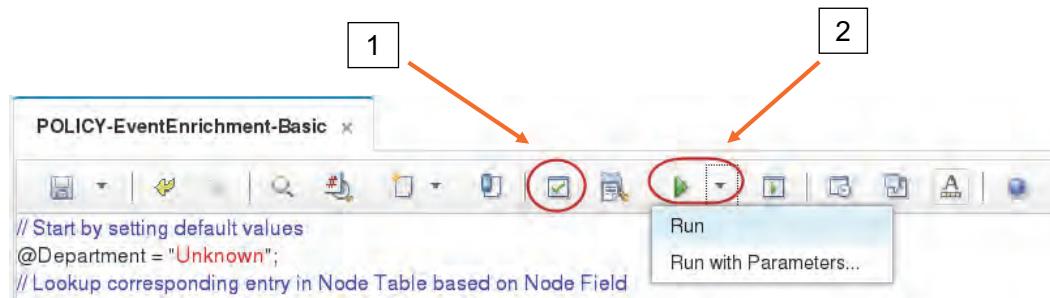
- Click **Run** from the **Policies** tab
- Associate a policy with a service on the **Mapping** tab
- Use the command-line nci_trigger tool

Launching policies for testing

Use the methods shown in the slide to launch policies for testing or automation purposes. Also, you can launch policies from within other policies.

Testing from the Policies tab

- Click to check syntax
- Click to launch the policy



The Enrichment policy

© Copyright IBM Corporation 2017

Testing from the Policies tab

Follow these steps to test from the **Policies** tab:

1. Click **Check Syntax**.
2. When all syntax errors are cleared, click **Execute Policy**.

The user interface tools offer one-time configuration of passed parameters for testing. Select **Run with Parameters** to test with this feature.



Note: The syntax checker must be free of errors before running a policy. The syntax checker cannot resolve logic errors.

Policy triggering

- The nci_trigger tool can be used by other tools and scripts to launch policies within Netcool/Impact
- These policy triggers, among others, are available:
 - OMNIbusEventReader
 - Event listeners
 - Policy activator
- Use this command:

```
$NCHOME/impact/bin/nci_trigger servername username/password policymame  
passed_param_name1 parameter passed_param_name1 parameter for multiple parameters
```

Policy triggering

With the **nci_trigger** command, you can launch a policy from the command line or from a script. Any parameter that is passed is treated in the same way as it would be treated in a typically executed policy. To test policies that perform ReturnEvent actions, make sure that **Identifier** and **EventReaderName** are set.

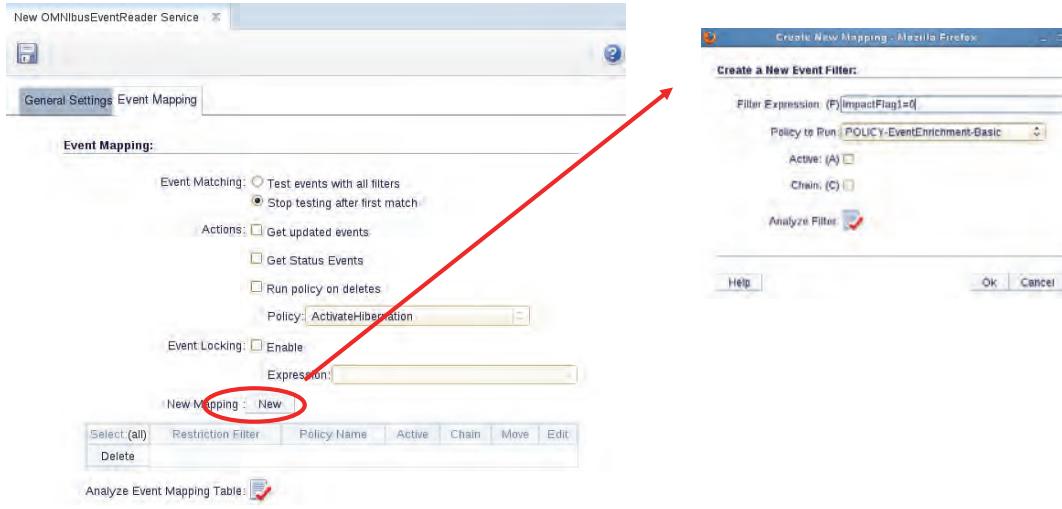
The following command is an example of the **nci_trigger** syntax:

```
nci_trigger NCI admin/netcool UpdateSelectedEvent Identifier node-up-23445  
Serial 2345
```

 **Note:** **EventReaderName** is not a field in OMNIbus. Netcool/Impact uses it to route the event to the appropriate event reader.

The OMNIbusEventReaderService

Click **New** to create a new event mapping



The Enrichment policy

© Copyright IBM Corporation 2017

The OMNIbusEventReaderService

Use the default OMNIbusEventReader (TESTENRICH) to launch the enrichment policy. The service associates the policy when the policy mapping information for the enrichment policy has been added. After the service is started, the log shows the event reader queries to the ObjectServer.

Testing the policy

- Testing with partial or full data
 - Do not use the probe data initially, because it is difficult to follow
 - Create test events in Netcool/OMNibus tools test events
 - Create events that match the specified condition
 - Create other events that do not match the condition
- Add log statements to the policy to verify field contents
- Automations, probes, visual objects, gateways, product integrations, and other elements affect how policies run
- Test in a controlled environment, and later test in a production analogous environment
- Remember that timing is critical

Testing the policy

Testing policies can be easier in a controlled environment. Events that the policy processes can be restricted by sending only one or two events through the policy.

The slide lists general testing considerations. However, each testing environment presents unique circumstances that you must always consider when testing policies.

Expected policy results

When executed, the enrichment policy in the example sets these fields:

- The **Department** field in the event, which displays looked-up department information or Unknown
- The **ImpactFlag1** field is set to a value of 1 after the event has been processed

Expected policy results

The policy enriches selected events with the department data. The **ImpactFlag1** field ensures that these events are not processed a second time by the same policy.

This policy requires that you manually add the **ImpactFlag1** to the data source, or that it was previously added using a script. While writing the policy, be sure to verify the existence of all data types that are accessed within policy logic.

Policy debugging

Suggestions for debugging policies include these examples:

- Limit the events that pass through the OMNIbusEventReader service
- Check data types for accuracy and verification of links
- Check spelling of variables
- Use the log function to verify policy flow and policy field contents
- Use service logging features

Policy debugging

When debugging policies, check the relevant log files for errors.



Note: \$NCHOME equals /opt/IBM/tivoli/impact for this installation.

Error handling

- Systems are dynamic and interrelated, which makes them inherently prone to errors over time
- You can write common error handlers to deal with errors without the user having to code many exceptions for each policy
- The error handler can contain many cases for exceptions that you can anticipate
- Users can raise custom exceptions and create handlers for the custom and the java.lang exceptions
- Page or notify the administrator when a specific type of failure occurs
- Deactivate a specific policy when multiple failures occur in the policy
- Turn off the event reader if a catastrophic exception condition occurs
- Create a Netcool/OMNibus event when an unknown exception occurs that the error handler does not know how to handle
- Flag an event not to be reprocessed if it causes an error a certain number of times

Error handling varies, depending upon the environment in which Netcool/Impact was deployed.

Review questions

1. What is the purpose of encryption?
2. Where are the default logs found for Netcool/Impact?
3. What is the function of the OMNIbusEventReader?

Review answers

1. What is the purpose of encryption?

The creator or administrator of a particular policy can use encrypting policies to hide the code from the Netcool/Impact user.

2. Where are the default logs found for Netcool/Impact?

*The logs are in **\$NCHOME/log/netcool.log**. To find any additional logs, check **\$NCHOME/impact/etc/NCI_server.props**.*

3. What is the function of the OMNIbusEventReader?

The OMNIbusEventReader polls the Netcool/OMNIbus ObjectServer (data source) for events.

Student exercises



[The Enrichment policy](#)

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Summary

You now can perform the following tasks:

- Describe Netcool/Impact Solution types
- Describe event enrichment policies
- Create a policy with the policy editor
- Launch a policy with the OMNIbusEventReader

Unit 7 Controlling policy execution sequence

IBM Training



Controlling policy execution sequence

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit focuses on controlling the operation of policies within Netcool/Impact. These topics are important in larger, more complex environments.

Objectives

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

- Describe Netcool/Impact threading and policy execution
- Configure policy chaining
- Configure event locking
- Control threading in policy blocks
- Control synchronized blocks

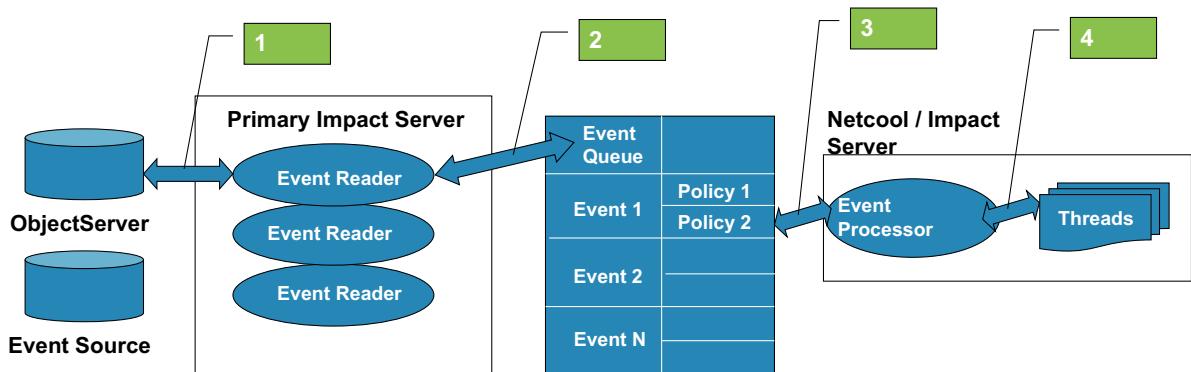
Threading

- Each instance of a policy running for each event in the queue uses one thread in execution
- With multiple threads, Netcool/Impact can process more efficiently
 - These threads can run on servers in the following ways:
 - A single server with a single processor (limited help)
 - A single server with multiple processors (takes advantage of multiprocessors)
 - Multiple servers running multiple Netcool/Impact servers
- Threading issues:
 - Execution order
 - Data contention

Threading

Because of the nature of independent event processors and threads, by default there is no guarantee that any policy or event is processed in a specific order.

Policy execution



Policy execution

The following steps are performed in policy execution:

1. Each event reader running on the primary Netcool/Impact server gathers events based on filter criteria
2. The event reader populates the event queue with events from Step 1. Based on each policy mapping entry, Netcool/Impact also must verify which policies will run on the event.
3. The event processor requests events to work on and receives the event.
4. The event processor executes one specific policy on one specific event. This action is performed on one particular thread. If multiple threads are enabled for a particular event processor, it can execute this step for each thread simultaneously.

Policy chaining

- Policy chaining is a Netcool/Impact policy language feature in which multiple policies can be configured to run in sequence
- An event reader service triggers chained policies

Policy chaining

Occasionally, it is useful to ensure that multiple policies execute in a certain sequence without having the processed event reprocessed by the event reader.

For example, consider the event enrichment policy. From a development perspective, it would be beneficial to exclude the code to retrieve the bits of data all in the same policy. Otherwise, all of the information gathered could require more work from various sources.

Consider breaking the event enrichment policy into separate policies for obtaining the following types of information:

- Department information
- Customer information
- Location information
- Part IDs

Policy chaining eliminates inefficiencies

Policy chaining performs in the following ways for greater efficiency:

- Defines a sequence in which policies are executed
- Limits execution of policies on queued events to be run in series, rather than simultaneously
- Can pass variables and context from each of the preceding policies to the following policy in the sequence
- Eliminates reprocessing inefficiencies

Policy chaining eliminates inefficiencies

Policy chaining can eliminate inefficiencies that occur from reprocessing the same events more than once.

Configuring policy chaining

OMNIbusEventReader Service

General Settings: Event Mapping

Event Mapping Table:

Double click or press ENTER on a cell to edit, press SHIFT + ENTER to finish editing.
Press New or Quick Add to add new mapping.

Restriction Filter	Policy Name	Active	Chain
	BasicChain1	Yes	Yes
	BasicChain2	Yes	Yes

Configuring policy chaining

Policy chaining is configured through the appropriate event reader. Each entry in the mapping table is executed in the order listed, if there are multiple entries with the **Chain** option selected. If you do not select the **Chain** option, Netcool/Impact does not guarantee the order in which those policies are executed.

Configuring threading

- Threading can be configured on each individual server within the cluster
All servers might not function in the same way
- Threading is controlled in the EventProcessor service



Configuring threading

The event processor operates by requesting events from the event reader queue to process. You can fine tune this operation by specifying the number of events that it pulls from the queue at one time and how often this fetch is performed.

To configure the event processor, click the **Event Processor** link to open the configuration window. The value in the **Minimum Number of Threads** field defines the minimum number of processing threads that can run policies at one time.

The value in the **Maximum Number of Threads** field defines the maximum number of threads that can run policies at one time.

When **Maximize** is checked the event processor tries to get the maximum performance out of the threads. This can result in high CPU usage. When you leave this field cleared, it runs conservatively at around 80% of peak performance.

If you set the option to maintain tuning on restart, each time the event processor is started it uses the same number of threads it had adjusted to in the earlier run. This feature is useful in cases where the environment where Netcool/Impact runs has not changed much from the previous run.

Controlling threading processing

- Two mechanisms control threading in Netcool/Impact:
 - Event-locking limits which events can be processed together system-wide, based on values contained within the event
 - Synchronized blocks to control which events are forbidden to be processed together, based on a much more granular level within each policy
- Event locking or synchronized blocks are used in the following situations:
 - To preserve the order in which incoming alerts are processed
 - To prevent a multithreaded event processor from attempting to access a single resource from more than one instance of a policy running simultaneously

Controlling threading processing

Event locking is a feature that allows a multithreaded event reader to categorize incoming alerts based on the values of specified alert fields and then to process them within a category one at a time in the order they were sent to the ObjectServer.

Configuring event locking

You configure event locking in the event reader



Locking Expressions categorize incoming events.
One or more fields from alerts.status

Example:
Node+AlertKey

Configuring event locking

Click the **OMNIbusEventReader** to edit the settings for the service. Event locking is configured on the **Event Mapping** tab. To enable this feature, select the **Enable** check box in the **Event Locking** section. After you enable it, you must enter a locking expression in the **Expression** field. This field specifies the way that the event broker categorizes incoming alerts by using a locking expression. The locking expression consists of one or more **alerts.status** field names concatenated with a plus sign (+).

For example, to prevent alerts with the same **Node** and **AlertKey** from being processed at the same time, set the following expression:

Node + AlertKey

Synchronized blocks

- With synchronization, an enclosed block of thread-critical code can run in a single-threaded mode
- Synchronization is used to implement thread-safe policies in Netcool/Impact
- Only one thread at a time can run the synchronized block of code; all others wait for their turn
- This feature is useful when multiple policies (or multiple threads of a single policy) are using a shared resource
- Usually, only write operations to a shared resource need synchronization
- If synchronization is not required, do not use it, because it slows the processing speed of your policies

Synchronized blocks

With the synchronized statement blocks feature of the Netcool/Impact Policy Language (IPL), you can write thread-safe policies for use with a multithreaded event processor.

Use synchronized statement blocks in situations where you anticipate that more than one instance of a policy (or different policies that access the same resource) will be running simultaneously on different event processor threads.

Synchronized statement blocks consist of any set of IPL statements that are enclosed in braces (`{ }`).

Synchronized blocks syntax

```
Synchronized (object) {  
    // section of IPL code  
}
```

- The object is the clause to be compared; it excludes any blocks from executing when the object is the same
- It must be passed as a variable (no literal values)
- If contention is found in the same Netcool/Impact server, the second or subsequent policies wait until the first synchronized block completes processing

Synchronized blocks syntax

When Netcool/Impact processes a synchronized statement block, it registers the synchronization identifier. No other synchronized statement block with that identifier can be executed until the registered block finishes running. This feature essentially *locks* the statement block. It permits only other statement blocks with the same identifier to run sequentially, rather than simultaneously, with the first block. As a result, resources accessed in the synchronized portion of the policy are protected from simultaneous access by multiple threads.

Sample policies without synchronization

```
//Netcool Impact 6.1 Policy 1
Log("==> Starting Policy 1 for event Serial : " + @Serial);

Log("==> Ending   Policy 1 for event Serial : " + @Serial);

//Netcool Impact 6.1 Policy 2
Log("==> Starting Policy 2 for event Serial : " + @Serial);

Log("==> Ending   Policy 2 for event Serial : " + @Serial);
```

Sample policies without synchronization

The two policy scripts log that the policies have started and ended. The sample policies do not include synchronization.

Sample policy with synchronization

```
//Policy 3
Log("==> Starting Policy 3 for event Serial : " + @Serial);
SynchObject="dummy";
synchronized(syncObject)
{
    log("== POLICY3 - Includes Synchronization : " + @Serial);
}

Log("==> Ending Policy 3 for event Serial : " + @Serial);
```

Sample policy with synchronization

This policy has synchronization. Because of synchronization, the policy runs only once for an event in the queue. No other instances of the policy start until the first one is complete.

Student exercises



Controlling policy execution sequence

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What is policy chaining?
2. How is synchronization used?

Review answers

1. What is policy chaining?

Policy chaining is a Netcool/Impact policy language feature with which multiple policies can be configured to run in sequence.

2. How is synchronization used?

You use synchronized statement blocks in situations where you anticipate that more than one instance of a policy, or different policies that access the same resource, will run simultaneously on different event processor threads.

Summary

You now can perform the following tasks:

- Describe Netcool/Impact threading and policy execution
- Configure policy chaining
- Configure event locking
- Control threading in policy blocks
- Control synchronized blocks

Unit 8 Policy wizards

IBM Training

IBM

Policy wizards

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit introduces Netcool/Impact wizards.

Objectives

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

- Describe the types of wizards
- Create a policy using a wizard

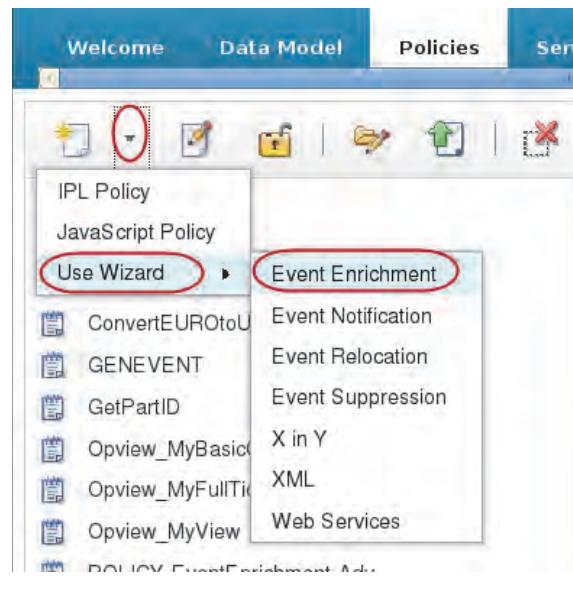
The purpose of wizards

- Use policy wizards to create simple policies without having to manually create data types and add functions
- Wizards provide step-by-step assistance when creating policies
- Using a wizard is the simplest way to create a policy in Netcool/Impact

The purpose of wizards

The wizards consist of a series of windows that provide guided policy creation. At the end of the process, you can run the policy immediately, with no further modification. However, modification is possible at any time using the policy editor.

Types of wizards



Policy wizards

© Copyright IBM Corporation 2017

Types of wizards

You can access these wizards by expanding the Wizards pane from the Netcool/Impact GUI. To use a wizard, open the Wizards task pane at the bottom of the **Navigation** panel and click the wizard type.

Wizard descriptions

- Event enrichment
 - Event enrichment policies add information about events from external data sources
- Event notification
 - Event notification policies notify someone that an event has occurred
- Event relocation
 - Use event relocation policies to send events from one central ObjectServer to another ObjectServer
- Event suppression
 - Event suppression policies set a flag in a Netcool/Impact event in response to a database query
 - A filter can use the flag to prevent the event from showing in the Event List

Wizard descriptions

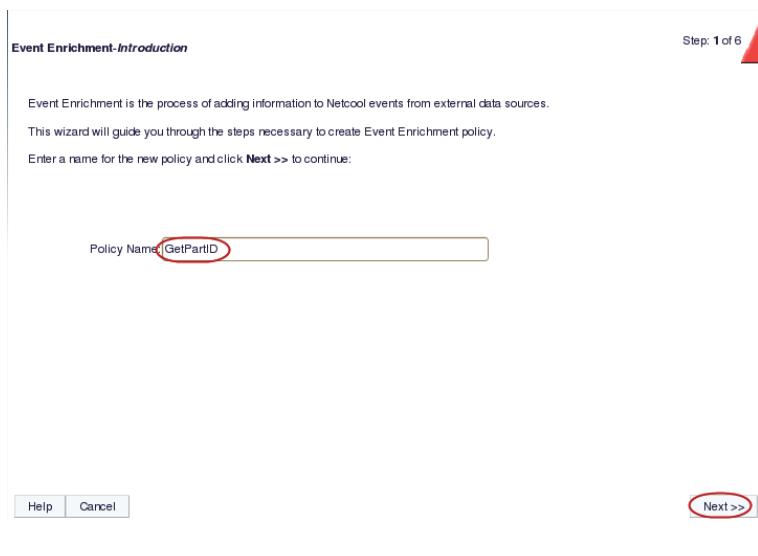
This slide describes four of the seven wizards available. Event enrichment is the most commonly used type of policy.

Wizard descriptions (continued)

- Web services
 - Using web-service DSA policies, Netcool/Impact can exchange data with external systems, devices, and applications using web-service interfaces
- XML DSA
 - With XML DSA policies, Netcool/Impact can read and extract data from any well-formed XML document
 - The XML DSA can read XML data from files, from strings, and from HTTP servers using the network (XML over HTTP)
- X in Y
 - X Events in Y policies suppress Netcool/Impact events until a certain number of identifiable events occurs within a specified period of time

Web Services, XML DSA, and X in Y are three additional available wizards.

Wizard example: Event enrichment



Policy wizards

© Copyright IBM Corporation 2017

Wizard example: Event enrichment

On the Event Enrichment wizard introduction window, provide a meaningful policy name.

Connect to an event reader

Event Enrichment-Connect to Event Reader

Step: 2 of 6

Event Source:

Select a previously configured Event Reader to supply events to this new policy:

Event Reader: TESTENRICH

Restriction Filter:

You can reduce the number of events returned by entering an SQL WHERE clause (e.g. Severity > 4 AND Acknowledged = 0)

PartNumber = 1

Help | Cancel | << Back | Next >>

Policy wizards

© Copyright IBM Corporation 2017

Connect to an event reader

Specify the event reader, and supply a restriction filter. The restriction clause for the selected event reader is the *where* clause for a selected statement. The restriction clause is being set against fields in the ObjectServer.

The clause is being used to ensure that when the field meets the criteria, the same event is no longer processed by this policy.

Use the integer fields for restriction clauses when possible. By using integers, the SQL query is more efficient against the ObjectServer.

Connect to a data source

Event Enrichment-*Connect to Data Source*

Step: 3 of 6

Data Source:

Choose from either a new or existing Data Type to supply information to this policy. If the Data Source you require is not listed in the selection list, please contact your Database Administrator for connection details and create a new Data Source before continuing.

Select a previously configured Data Type:

Data Type: ADDITIONAL_INFORMATION_CONFIGUR

Provide a name for a new Data Type and select a Data Source:

Data Type: HS_Inventory

Data Source: HS_TRAIN

Help Cancel

<< Back Next>>

Policy wizards

© Copyright IBM Corporation 2017

Connect to a data source

Using the wizard, you can select a data type or create a new one. The data source must already exist.

Identity information

Event Enrichment-Identity Information Step: 4 of 6

Data Source Base Table:

Select a Schema and Table from the data source to identify the event enrichment information. After the selections are made, press the Refresh button to update the mappings for the filter builder.

Base Table: IMPACT INVENTORY Refresh

Database Filter Builder:

Build a filter based on the mapping between data source and event source. The filter will be used to establish whether the event enriched. Choosing a value from the Field dropdown will cause the Comparator and Value dropdowns to dynamically update based on choice.

Choose match criteria:

Match all Match any

Field	Comparator	Value
NODE	=	Node

Add Conditions View SQL Clear All

Database Query:

Build a database query to return the event enrichment information:

Match all Match any

Field	Comparator	Value
Select Field	Select Field	
PARTID		
NODE		
INTERFACE		
STATUS		
TYPE		
VERSION		

Add Conditions Clear All

<< Back Next >>

Policy wizards

© Copyright IBM Corporation 2017

Identity information

The identity information consists of the base table. When populated, the base table shows the fields after you click **Refresh**. You create a database query by using the table fields, selecting a comparator, and associating a value.

Data source and event mappings

Event Enrichment-Enrich Event

Step: 5 of 6

Data Source to Event Field Mappings:

Select the desired event field then click Refresh to update the list of Available Data Source fields.

Event Fields: PartNumber

Refresh

Data Source Fields:

Select a data source from the list of Available Data Source Fields then click the Add button to add it to the list of Selected Data Source Fields. Repeat this for each data source field you want to map to the event field, then click Apply to save your selections.

Available Data Source Fields: NODE, INTERFACE, STATUS, TYPE, VERSION

Selected Data Source Fields: PARTID

Add > Remove

Apply

<< Back Next >>

Help Cancel

Policy wizards

© Copyright IBM Corporation 2017

Data source and event mappings

The Event Enrichment wizard provides windows for data mapping.

Policy and Data Summary

Event Enrichment-Summary and Finish

Step: 6 of 6

Policy and Data Summary:

Click Finish to complete the wizard. Once the wizard has completed creating the requested items, the wizard will close itself and return you to the policy editor with the new policy. The wizard will create the following items:

Policy.GetPartID
Data Type:HS_Inventory

Help Cancel

<< Back Finish

Policy wizards

© Copyright IBM Corporation 2017

Policy and Data Summary

The Policy and Data Summary window displays the selected information. Click **Finish** at the bottom of the window to create the policy.

Viewing the policy

The screenshot shows the IBM Policy Editor interface. At the top, there's a navigation bar with tabs: Welcome, Data Model, Policies, and a workspace showing a project named 'ACMEBP_Te'. Below the navigation bar is a toolbar with various icons. The main area is titled 'GetPartID' and contains a code editor with the following Java-like pseudocode:

```
//This policy generated by Impact Wizard.  
  
log("Start policy: GetPartID.");  
EnrichmentOrgNodes = GetByFilter('HS_Inventory', "(NODE='"+EventContainer.Node+"')", false);  
Num = length(EnrichmentOrgNodes);  
log("GetByFilter successful. Found "+Num+" dataItem(s).");  
if (Num > 0) {  
    EventContainer.PartNumber = ''+ EnrichmentOrgNodes[0].PARTID;  
    ReturnEvent(EventContainer);  
    log("ReturnEvent successful");  
}
```

The **Policies** pane displays the policies that are created using a wizard. You can make modifications from the policy editor and save them.

Starting the OMNIbusEventReader (TESTENRICH)

The screenshot shows the IBM Policy Studio interface for the service 'TESTENRICH'. The 'Event Mapping' tab is selected. An 'Event Mapping Table' is displayed with one row:

Restriction Filter	Policy Name	Active	Chain
PartNumber=''	GetPartID	Yes	No

Below the table, the 'Services' panel shows the service 'TESTENRICH' with a green play button icon circled in red, indicating it is running.

Policy wizards

© Copyright IBM Corporation 2017

Starting the OMNIbusEventReader (TESTENRICH)

To start the OMNIbusEventReader, locate the service TESTENRICH in the **Services** panel. Click the green arrow to start event reader. Open the logs to verify processing. The green checked box indicates that the service is running.

Wizard considerations

- All wizards consist of a series of windows that guide you through the policy creation process
- At the end of the process, you can run the generated policy immediately with no further modification
- You can modify policies in the policy editor at any time
- The OMNIbusEventReader service must be running before you can use a wizard, with the following exceptions:
 - Web Services wizard
 - XML DSA wizards
- You must configure the EmailSender service before you can use the EventNotification policy wizard

Wizard considerations

This basic information is true for all policies created using the wizards. To start the OMNIbusEventReader, locate the service TESTENRICH in the **Services** panel. Click the green arrow to start event reader. Open the logs to verify processing.

Student exercises



Policy wizards

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What is the purpose of wizards?
2. Can you edit policies created with wizards?

Review answers

1. What is the purpose of wizards?

You can use policy wizards to create simple policies without having to manually create data types and add functions.

2. Can you edit policies created with wizards?

Yes, you can use the policy editor to edit policies created with wizards.

Summary

You now can perform the following tasks:

- Describe the types of wizards
- Create a policy using a wizard

Unit 9 Notification policies

IBM Training

IBM

Notification policies

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes how Netcool/Impact uses notifications, including sending and receiving email and instant messaging.

Objectives

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

- Set up Netcool/Impact notifications
- Describe email capabilities
- Configure the EmailSender service
- Describe instant messaging capabilities
- Create a notification policy and verify its results

Lesson 1 Notification policies

IBM Training



Lesson 1 Notification policies

- Provide real-time responses to an event or a series of events
- Supports advanced enrichment by email and instant message notification and response
- Provide advanced notification based on schedules
 - Route event information to specific operators, based on the on-call schedule for the organization

Netcool/Impact has the added function of providing email notification for information reasons or for escalation procedures, further enhancing the event management process. Netcool/Impact can also receive email to activate policies where necessary. Notification policies provide immediate response to notification events. This feature is especially important in environments where all outages are considered critical and require immediate response.

Types of notifications

- Email service
 - With Netcool/Impact you can send email from within a policy
 - This feature is used to send email notification to administrators and users when a certain event or combination of events occurs
- Instant messaging
 - Instant messaging (IM) is a network service that two participants can use to communicate using text in real time
 - You can use Netcool/Impact IM to send and receive instant messages from within a policy

Types of notifications

Netcool/Impact supports two kinds of notifications:

- Email service
- Instant messaging

Email service: EmailSender service

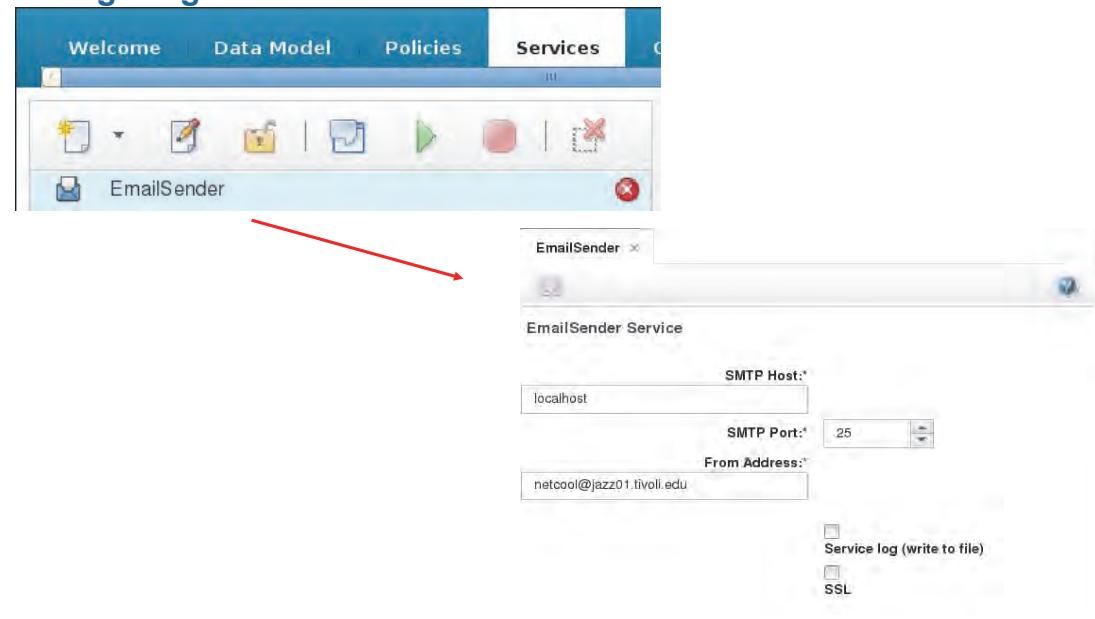
- This service uses the SendEmail() action function
- To use this function, you must configure and start the SendEmail service

Email service: EmailSender service

With Netcool/Impact, you can send email from within a policy. You can use this feature to send email notification to administrators and users when a certain event or combination of events occurs.

Netcool/Impact does not provide a built-in mail server. Make sure that an SMTP server is available in your environment before attempting to send an email. The Netcool/Impact email sender service must also be running before a policy can successfully send email.

Configuring the EmailSender service



Notification policies

© Copyright IBM Corporation 2017

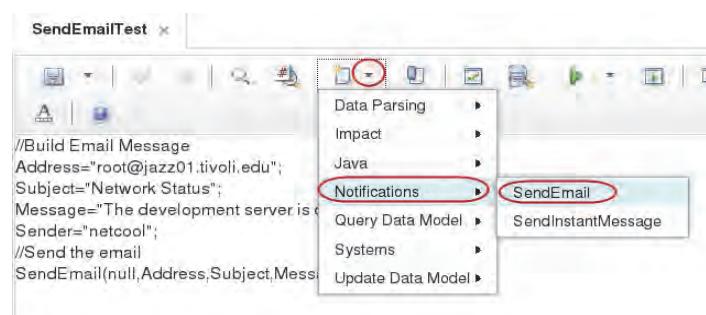
Configuring the EmailSender service

The EmailSender service is in the **Services** tab. Click the service name to open the EmailSender Configuration window. If the fields are not automatically populated, check with the system administrator at your location for the required values.

Using the SendEmail action function

SendEmail has eight parameters:

- User (optional)
- Address
- Subject (optional)
- Message
- Sender (optional)
- ExecuteOnQueue
- Attachment
- CallProperties



```
SendEmail([User], [Address], [Subject], Message, [Sender], ExecuteOnQueue,  
[Attachment],[CallProperties])
```

Using the SendEmail action function

The **SendEmail** function uses the following parameters:

- **User**: The OrgNode of any data type named **User** whose **Email** field contains the email address of the recipient for the email. If this parameter is used, the address variable is ignored.
- **Address**: The target email address.
- **Subject**: The subject for the message.
- **Message**: Text for the message.
- **Sender**: The user who sent the message.
- **ExecuteOnQueue** Set to FALSE.

ExecuteOnQueue specifies whether to place the outgoing message in the queue that the command execution manager service controls. If you specify TRUE, the message is placed in the queue and sent asynchronously by this service. If you specify FALSE, the message is sent directly by the Netcool/Impact. In this case, the policy engine waits for the message to be sent successfully before processing any subsequent instructions.

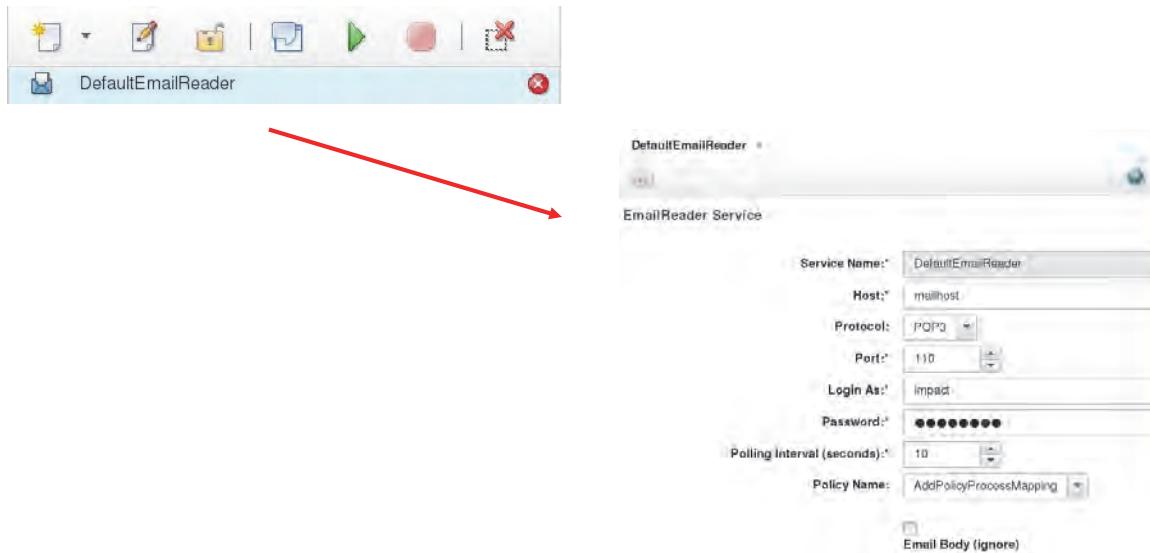
The following example shows how to send an email. In this example, the email is sent to the address **admin@example.com**.

```
// Call SendEmail and pass the addressee, subject, body content and //sender as
// input parameters.
Address = "admin@example.com";
Subject = "URGENT: Node down";
Message = "URGENT MESSAGE: Node 1237ASDS is not responding to network ping.";
Sender = "Netcool/Impact";
ExecuteOnQueue = False;
SendEmail(null,Address, Subject, Message, Sender, ExecuteOnQueue);
```

Receiving email

- To receive email, the DefaultEmailReader must be configured properly
- When the DefaultEmailReader pulls email from the server, it launches a specified policy to process the email

DefaultEmailReader service configuration



Notification policies

© Copyright IBM Corporation 2017

DefaultEmailReader service configuration

The email reader service polls a specified POP host for email messages. It reads email from a mailbox at intervals that are defined when the email reader service is created. The DefaultEmailReader service requires the following parameters:

- **POP Host:** This parameter is the mail host name.
- **Login As:** The default value is the one used to log in to Netcool/Impact.
- **Password:** Typed characters are displayed as asterisks (*).
- **Polling Interval:** This parameter records how often, in seconds, the service polls the POP host for new email messages.
- **Policy:** This policy is the one that executes for this event.
- **Startup:** This parameter specifies to automatically start when the server starts.
- **Service Log:** This parameter specifies to write to the log file.

Example of a policy to process email

Parse the email, and acknowledge an event based on a subject containing ACK:serial number

```
// Log the email retrieved
log("Email Received from " + @Sender);
log("Subject: " + @Subject);
log("Message was :\n" + @Message);

// Check that the Subject is formatted properly
if (@Subject like "ACK:[0-9]+$")
{
    Serial = reextract (@Subject, "ACK:([0-9]+)$");
    BatchUpdate("Alerts","Serial = " + Serial,"Acknowledged = 1");
}
```

Example policy to process email

This example shows a policy that processes an email. The policy logs that the email was retrieved, checks that the subject has the correct format, and sends an alert.

Sending instant messages

- Jabber is a set of protocols and technologies that provides the means for two software entities to exchange streaming data over a network
- Netcool/Impact uses the JabberService service to communicate with instant messaging systems to notify administrators, operators, and other users when certain events occur
- Messages are sent during the execution of a policy when a call to the SendInstantMessage function is encountered
- You can configure the JabberService to connect to one of these services:
 - Jabber server
 - AOL Instant Messenger
 - MSN Messenger
 - Yahoo! Messenger
 - ICQ messaging service

Sending instant messages

With Netcool/Impact instant messaging (IM), you can send and receive instant messages from within a policy. Netcool/Impact can monitor an instant messaging account on any of the most popular services for incoming messages. It can also perform operations when specific messages are received.

Netcool/Impact can send instant messages to any other IM account. You can use IM to notify administrators, operators, and other users when certain events occur in your environment.

Netcool/Impact IM uses Jabber to send and receive instant messages. Jabber is a set of protocols and technologies through which two software entities can exchange streaming data over a network. For more information, see the Jabber website at the following location:

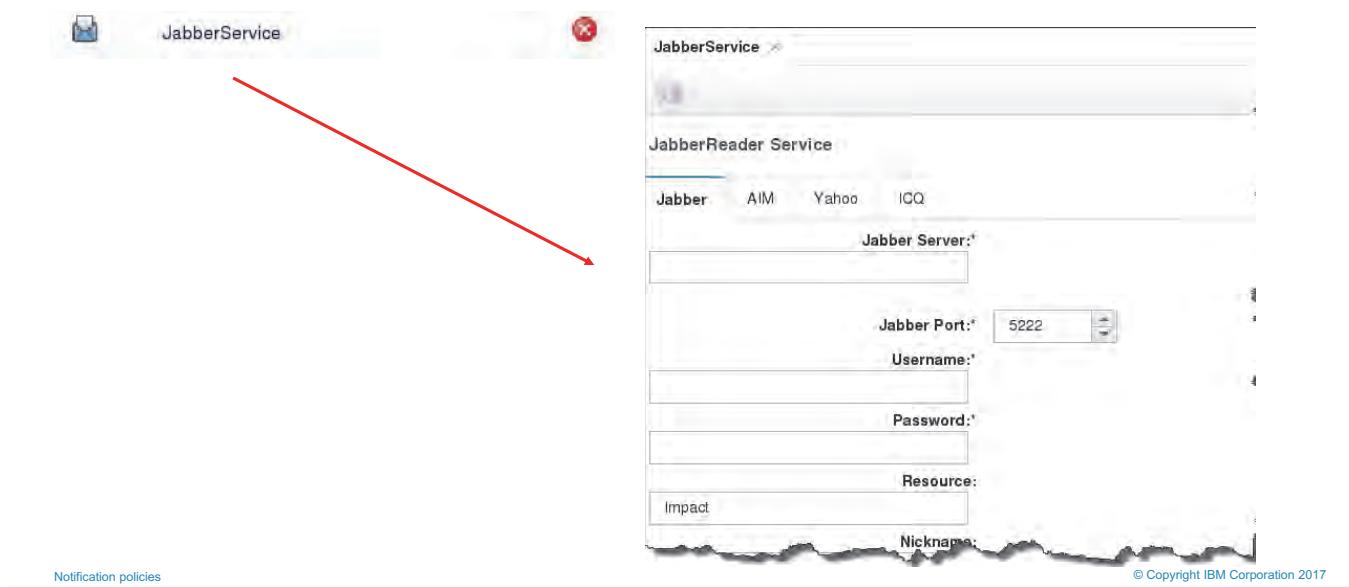
www.jabber.org

Netcool/Impact has two types of services that work together with your policies to provide the IM feature. The Jabber reader service listens for incoming instant messages and then starts a specified policy when a new message is received. The Jabber service sends messages to other IM accounts.

The Netcool/Impact IM process has the following phases:

- Message listening
- Message sending

Configuring JabberService



Configuring JabberService

During the message listening phase, the Jabber reader service listens for new messages from one or more IM accounts. When a new message is received, the Jabber reader creates an EventContainer and populates it with the contents of the incoming message. The Jabber reader starts the policy that is specified in its configuration settings and passes the EventContainer to the policy. Netcool/Impact then processes the policy.

The JabberService is on the **Services** tab. Click the service name to open the JabberService Configuration window.

Using the SendInstantMessage action function

SendInstantMessage has five parameters:

- To
- Group (optional)
- Subject (optional)
- TextMessage
- ExecuteOnQueue

Using the SendInstantMessage action function

SendInstantMessage requires the following parameters:

- **To:** Contains the instant message name to send the message to (format varies by messaging service).
- **Group:** Can be used to indicate the group for some instant message types.
- **Subject:** The subject for the message, which is used for some messaging services.
- **TextMessage:** The content of the instant message.
- **ExecuteOnQueue:** A value of TRUE places the message on queue and waits for an appropriate time to send it.

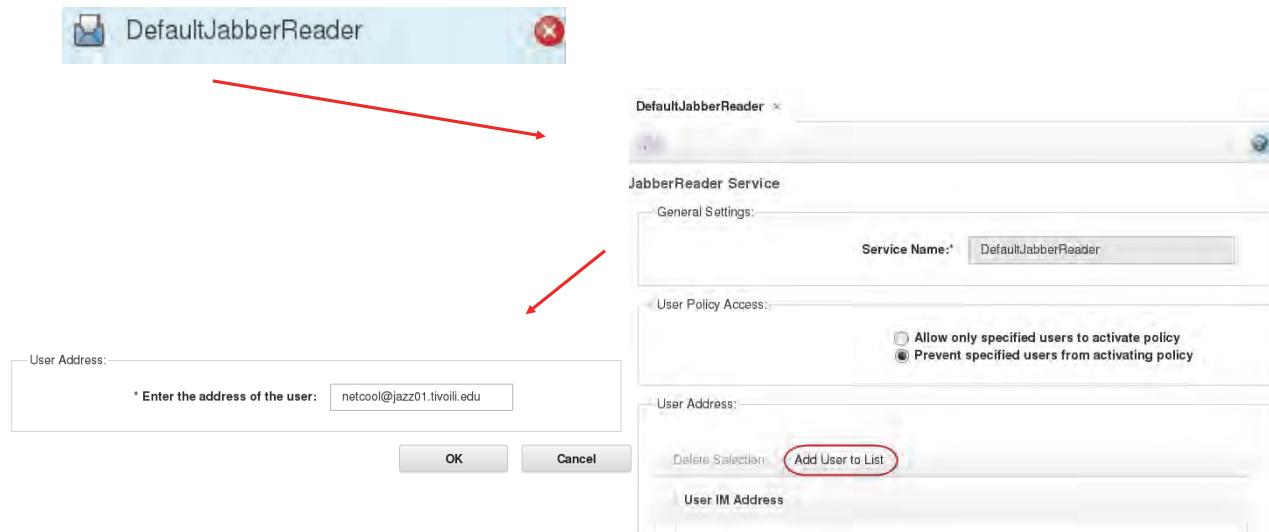
Receiving instant messages

- To receive an instant message, you must properly configure the DefaultJabberReader
- When the DefaultJabberReader detects reception of a message, it launches a specified policy to process the message
- The DefaultJabberReader connects to the messaging service specified in the JabberService
- The policy receives context parameters:
 - from
 - body

Receiving instant messages

When the JabberReader receives an incoming message, it populates the **From** and **Body** fields in the EventContainer variable. The **From** field contains the user name of the account from which the message was sent. The **Body** field contains the contents of the message. You can access the contents of the fields by using either the dot notation or the @ notation.

DefaultJabberReader configuration



Example of a policy to process instant messaging

Responding to any instant message sent to this location:

```
// Save the IM retrieved
Msg      = EventContainer.body;
Sender = EventContainer.from;

log("IM from " + Sender);
log(Msg);

// Send IM Response
SendInstantMessage(Sender, null, "Server Status", "Up and running ", false);
```

Example of a policy to process instant messaging

The policy to process the IM sets two variables: *Msg* and *Sender*. It also logs to keep track of the sender. **SendInstantMessage** is used to send a short response.

Student exercises



Notification policies

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What is the purpose of notification policies?
2. What are the types of notifications?

Review answers

1. What is the purpose of notification policies?

Notifications provide immediate response to events.

2. What are the types of notifications?

Emails and instant messaging

Summary

You now can perform the following tasks:

- Set up Netcool/Impact notifications
- Describe email capabilities
- Configure the EmailSender service
- Describe instant messaging capabilities
- Create a notification policy and verify its results

Unit 10 Reports

IBM Training



Reports

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit introduces the Netcool/Impact reporting features.

Objectives

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

- Describe the types of reports available
- Create a report
- Describe Service Level Objective reporting
- Use the Configuration Documenter

Lesson 1 Introduction to reports

IBM Training

IBM

Introduction to reports

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

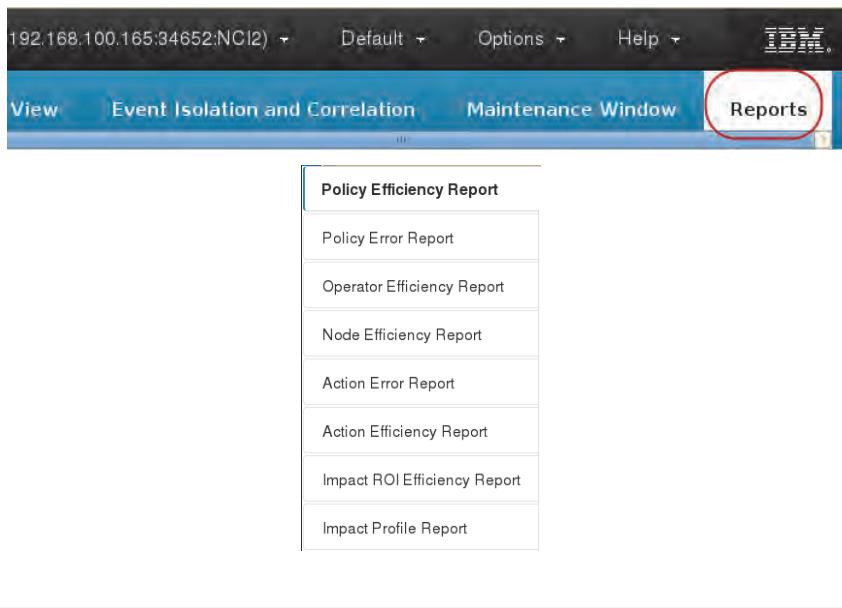
Types of reports

- Netcool/Impact provides eight reporting tools
- Two reports provide information about the network and network operators:
 - Node Efficiency report
 - Operator Efficiency report
- Six reports help to assess the efficiency of the Netcool/Impact configuration:
 - Action Efficiency report
 - Action Error report
 - Impact Profile report
 - Impact Return on Investment (ROI) report
 - Policy Efficiency report
 - Policy Error report

Types of reports

Eight reports are available in the Netcool/Impact product. Reports provide analytical information about running policies, which helps highlight potential problems. The information might indicate that a particular policy requires modification to function more efficiently.

Available reports



Reports

© Copyright IBM Corporation 2017

Available reports

Access the Netcool/Impact reports by expanding the Netcool/Impact **Reports** panel. The reports that are listed are added automatically during product installation.

Creating a report

- To view a report, follow these instructions:
 1. Click the **Reports** tab
 2. Select a report
- The selected report editor opens in the Main Work panel
- Report buttons can vary, depending on which report you select

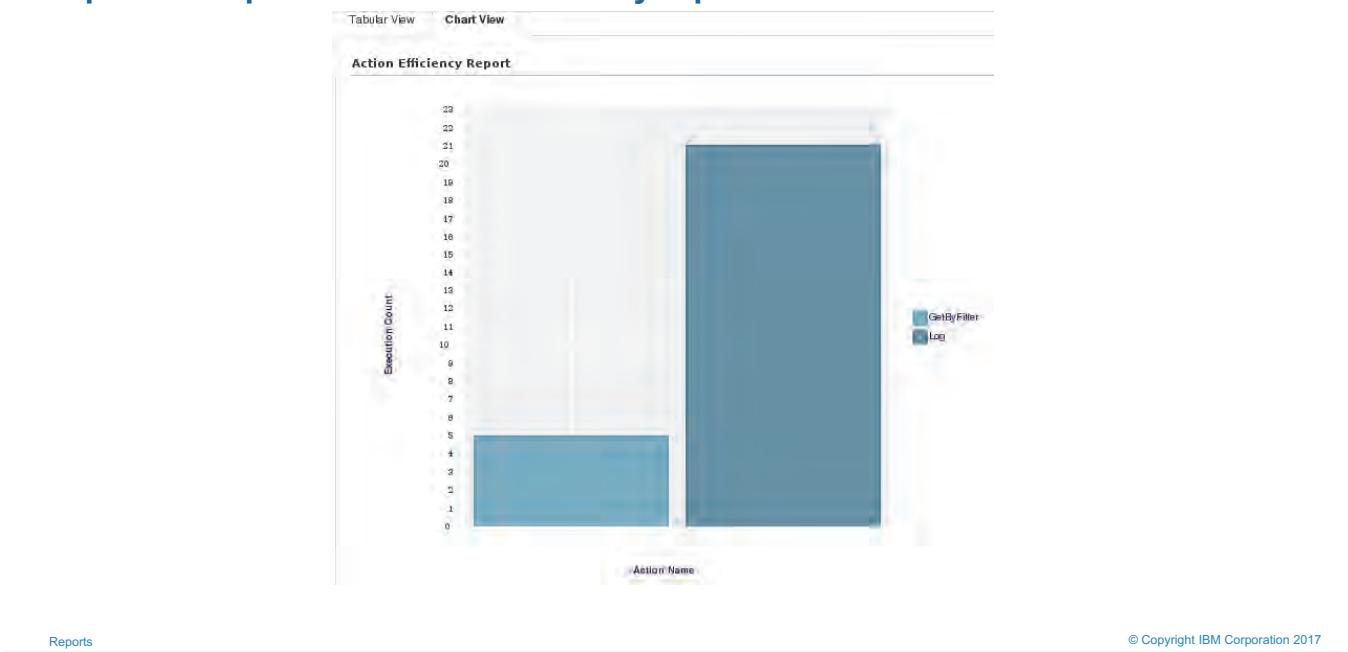
Creating a report

Creating a report in Netcool/Impact is a straightforward process. Click one of the available reports, and enter the required reporting time range.



Note: Refresh the information in a report by clicking **Refresh Report Data** while viewing the report.

Report example: The Action Efficiency report



Report example: The Action Efficiency report

The **Action Efficiency Report** shows the total number of actions that were processed over a selected time range. Specifically, the report shows how many actions the Netcool/Impact server performed and which actions used the most. The Action Name is on the X axis, and Execution Count is on the Y axis. The key on the right shows the colors of specific action functions: **Log**, **SendEmail**, **GetbyFilter**, and **GetByKey**. In this case, **Log** was used most often during the time range indicated.

Lesson 2 Data type reporting

IBM Training

IBM

Data type reporting

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Data type reporting is useful in determining the efficiency of a particular data type, and therefore the efficiency of the parent data source.

Overview of data type reporting

- Data type reports collect statistical data about a specific tables
- Data produced for these reports is collected in the historical database
- The reports are available from the **Data Model** tab
- Not all data types provide this option
For example: data types defined to the internal data source
- The policy logger must be set to **Collect Reports**

General Settings:

Error-handling Policy:	DefaultExceptionHandler
Highest Log Level:	0
<input type="checkbox"/> Policy Profiling	
<input checked="" type="checkbox"/> Collect Reports	

Cache settings

- Edit the data type
- Open the **Cache Settings** tab

The screenshot shows the 'SQL Data Type Config Page' for the 'HS_Node' table. The 'Cache Settings' tab is selected. On the left, there's a navigation pane with 'Welcome', 'Data Model', and 'Policies' tabs, and a tree view of database structures under 'HS_TRAIN'. A red arrow points from the 'HS_Node' item in the tree view towards the configuration page. The configuration page has sections for 'Data Caching' and 'Invalidate Cached Data Items After'. In the 'Data Caching' section, there's a checkbox for 'Enabled' which is checked, and a dropdown for 'Maximum number of data items' set to 1. In the 'Invalidate Cached Data Items After' section, there are three dropdowns: 'Days' (0), 'Hours' (0), and 'Minutes' (0). A red arrow also points down towards the 'Days' dropdown.

Reports

© Copyright IBM Corporation 2017

Cache settings

The **Cache Settings** provide the configuration options for the indicators in the report by regulating the flow of data between Netcool/Impact and the external data source.

Caching helps you to decrease the load on the external databases that Netcool/Impact uses. Data caching also increases system performance by temporarily storing data items that were retrieved from a data source.

Configuring cache settings

The screenshot shows the 'Configuring cache settings' page with several sections:

- Data Caching:**
 - Enabled checkbox (unchecked)
 - Maximum number of data items: 1
 - Invalidate Cached Data Items After:
 - * Days: 0
 - * Hours: 0
 - * Minutes: 0
 - * Seconds: 0
- Count Caching:**
 - Enabled checkbox (unchecked)
 - Invalidate Cached Count After:
 - * Days: 0
 - * Hours: 0
 - * Minutes: 0
 - * Seconds: 0
- Query Caching:**
 - Enabled checkbox (unchecked)
 - Maximum Number of queries: 1
 - Invalidate Cached Queries After:
 - * Days: 0
 - * Hours: 0
 - * Minutes: 0
 - * Seconds: 0
- Performance Measurements Intervals:**
 - Set the intervals to measure how fast queries are run against a data type.
 - Polling Interval (seconds): 0
 - Query Interval (queries): 0

Reports

© Copyright IBM Corporation 2017

Configuring cache settings

Caching works best for static data sources and for data types where the data does not change often.

Enable Data Caching toggles data caching on and off.

- **Maximum number of data items:** Sets the total number of data items to be stored in the cache during the execution of the policy.
- **Invalidate Cached Data Items After:** Invalidates the cached items after the time periods selected.

Enable Query Caching toggles query caching on and off.

- **Maximum number of queries:** Limits the maximum number of database queries to be stored in the cache.
- **Invalidate Cached Queries After:** Invalidates the cached items after the time periods selected.

Enable Count Caching: Do not set is available for compatibility with earlier versions only.

- **Performance Measurements Intervals:** Sets the reporting parameters for measuring how fast queries against a data type are executed.
- **Polling Interval:** Select a polling interval for measuring performance statistics for the data type.

Query Interval is where you select the query interval for the performance check.

View performance Report

View Performance Report is found on the Data Model page

The screenshot shows the IBM Data Model interface. In the center, there's a panel titled "Data Type Performance Report: HS_Node". Below it, under "Performance Averages", is a list of metrics. At the bottom of the interface, there's a "Cache Status" section. On the left, a tree view shows a folder "HS_TRAIN" containing "HS_Customer", "HS_Department", "HS_Inventory", "HS_Location", and "HS_Node". A context menu is open over "HS_Node", listing "Edit", "View Data Items", "View Performance Report" (which is circled in red), "Unlock", and "Delete". A red arrow points from the "View Performance Report" item in the context menu towards the main report panel.

Reports

© Copyright IBM Corporation 2017

View performance Report

Select **View Performance Report** to view the statistics for a particular data type. Allow time for history data to be generated and collected to get better performance indicators.

Lesson 3 Service Level Objective reporting

IBM Training



Service Level Objective reporting

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Service Level Objective reporting

Service Level Objective or SLO reporting package is a new, optional feature that you can add to the Netcool/Impact 6.1.1 release:

- The SLO package provides policies and schemas that you can use to store SLO metrics in DB2
- You can use the metrics to generate reports with Tivoli Common Reporting
- You have the following capabilities with Tivoli Common Reporting:
 - Data stores
 - Reporting engines and their corresponding web user interfaces displayed in Dashboard Application Services Hub
 - A command-line interface

Installing Service Level Objective reporting

- Netcool/Impact 6.1.1 must be installed and configured before you can apply the SLO Reports package
- The SLO Reports package is in the `<install_home>/impact/add-ons/slo` directory
The **importData**, **db**, and **Report** directories are included in the SLO Reports package
- The Reports package extensions are a set of policies, data source, and data type configurations
- The version of DB2 must be 9.7 Fix Pack 4 or later, which is available as a bundle with Netcool/Impact
- Tivoli Common Reporting version 2.1 or higher is required to install the SLO Reports package
- A database named SLORPRT must be created in the DB2 system, cataloging the TCP/IP remote node and the remote database

For more information on configuration, see the *IBM Tivoli Netcool/Impact* documentation.

Lesson 4 The Configuration Documenter

IBM Training



The Configuration Documenter

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Use the Configuration Documenter to view all of the components in a Netcool/Impact installation. The configuration documenter also provides an alternative way to view the data model. It is an invaluable tool in understanding and conveying the information in the data model.

Introduction to the Configuration Documenter

- The Netcool/Impact Configuration Documenter is a web-based tool that you can use to view the configuration of a Netcool/Impact installation
- With this tool, you can view details about the following items:
 - Cluster status
 - Service status
 - Data sources
 - Data types
 - Policies
 - Services
- The Configuration Documenter is available automatically when the Netcool/Impact server is started
- No additional steps are required to start this tool

Opening the Configuration Documenter

To open the documenter, open the following address:

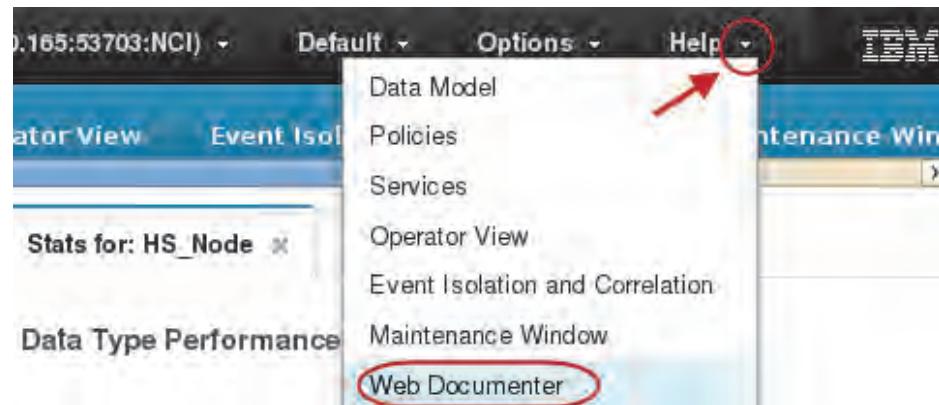
http://hostname:port/clustername_instance_documenter/index.html

where:

- hostname is the name of the host where you are running Netcool/Impact
- port is the port of the Netcool/Impact HTTP service (9080 is the default)
- clustername is the name of the server cluster
- instance is the name of the Netcool/Impact server

Configuration Documenter links

Click the **Web Documenter** link



Reports

© Copyright IBM Corporation 2017

Configuration Documenter links

The web address varies, depending upon the Netcool/Impact installation configuration parameters.
For this class, use the following location:

http://jazz01.tivoli.edu:9080/NCICLUSTER_NCI_documenter/index.html

The Configuration Documenter application

IBM Tivoli Netcool/Impact 7.1.0.5

Configuration Documenter

Refresh IBM

Server Name: NCI Version: 7.1.0.5 (201609221218) Generated on: Wed Jun 22 20:30:01 UTC 2016

Status | Data Sources | Data Types | Policies | Services

Cluster Status for NCICLUSTER

Primary Server	Host
NCI (Current Instance)	jazz01.tivoli.edu

Server Status

NCI

Memory Status	Current Usage (In MB)	Max Limit (In MB)
Heap Memory Utilization	187	1200

Event Status

NCI

Service Name	Number of Events in Queue	Event Source
EventProcessor	0	ProcessSeasonalityEvents

Data Sources

Data Source Name	Data Source Type
EIC_alertsdb	ObjectServer
EventrulesDB	DB2
HS_TRAIN	Derby
IBMConnectionsObjectServerDSA	ObjectServer
ITNM	DirectMediator

Reports © Copyright IBM Corporation 2017

The Configuration Documenter application

Click **Help** and then **Web Documenter** to access the application in the GUI. The Configuration Documenter application provides a snapshot of all the data in Netcool/Impact. You can click many of the fields to view more detailed information.

Student exercises



Reports

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What types of reports provide information about the network?
2. What is the Configuration Documenter?

Review answers

1. What types of reports provide information about the network?

The Node Efficiency and the Operator Efficiency reports

2. What is the Configuration Documenter?

The Configuration Documenter is a web-based tool that you can use to view the configuration of a Netcool/Impact installation.

Summary

You now can perform the following tasks:

- Describe the types of reports available
- Create a report
- Describe Service Level Objective reporting
- Use the Configuration Documenter

Unit 11 Operator views

IBM Training

IBM

Operator views

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes how to customize and use the operator view.

Objectives

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

- Configure an operator view from the user interface
- Customize an operator view in the HTML files and Netcool/Impact policies

Lesson 1 Introduction to operator views

IBM Training

IBM

Introduction to operator views

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This lesson covers the *operator view*, which is a custom web-based tool that you can use for the following tasks:

- View events and Netcool/Impact data in real time
- Run Netcool/Impact policies based on that data

Operator view benefits

- An operator view is a custom web-based tool that you use to view and process events and data in real time
- Operator views can show extra information to operators that does not show in the event list
- Some questions that can be answered using an operator view include these examples:
 - Has the event occurred previously?
 - What is the original trouble ticket number?
 - How many times in the past 3 months has this type of error occurred?
- Operator views are often used for the following tasks:
 - Provide a recommended course of action to fix a problem
 - Run a policy that correlates the event data with business data that is stored in your environment
 - Show the correlated business data to a user
 - Run one or more policies based on the event or business data
 - Start another operator view based on the event

Operator view benefits

There are a variety of uses for displaying data from external databases and presenting them in the Operator View.

Types of operator views

- Netcool/Impact includes two types of operator views:
 - Basic
 - Advanced
- Netcool/Impact operator views include two primary components:
 - Display page
 - Operator View policy

Types of operator views

Netcool/Impact supports two types of operator views: basic operator views and advanced operator views. With the *basic* operator views, you can display Netcool/Impact data in a preformatted web page. With the *advanced* operator views, you can display Netcool/Impact data using any HTML formatting that you prefer.

An operator view consists of two components:

- **Display page:** A text file with HTML content and special instructions called *smart tags* that tell Netcool/Impact what data to display and how to display it
- **Operator view policy:** A Netcool/Impact policy that contains the logic required to retrieve and manipulate the data displayed in the view

Configuring basic operator views



Operator views

© Copyright IBM Corporation 2017

Configuring basic operator views

Operator views are located in **System Configuration > Event Automation**. Clicking the **New Operator View** icon opens the configuration screen. There you provide a name and basic layout options.

Layout Options specify the position of the event panel and action panel in the operator view. The **Preview** at the bottom gives an indication of what the operator view will look like when it is complete.

To set up a basic operator view, use the Netcool/Impact GUI to specify a page layout and specify the Netcool/Impact data to display in the view. The GUI automatically creates the display page and the operator view policy.

To set up an advanced operator view, use a text editor or the Netcool/Impact GUI to create the operator view policy. Then use the text editor to create the display page associated with the view.

Action Panel settings



Action Panel settings

From the Action panel, you can select one or more policies to be used by the operator view. This is an optional part of the operator view function. You use the Action panel only to start policies. It does not display data that is returned by a policy. An advanced operator view, however, does provide the capability to display this data.

Information group settings



Information group settings

An information group is a set of dynamic data that is displayed when you open the view. The definitions are stored in a table:

- **Group Name:** Provide a unique name for the group.
- **Type:** Select By Filter or By Key to specify whether the information group retrieves data from a data type by filter or by key.
- **Data Type:** Select the data type that contains the information you want to display.
- **Value:** Choose a filter or key expression. If the type is **By Filter**, adding a value is optional. If the type is **By Key**, the value is mandatory.
- **Style:** Choose **Tabbed** or **Table** to specify how the operator view shows the resulting data.



Note: Press Esc on your keyboard to cancel the edit.

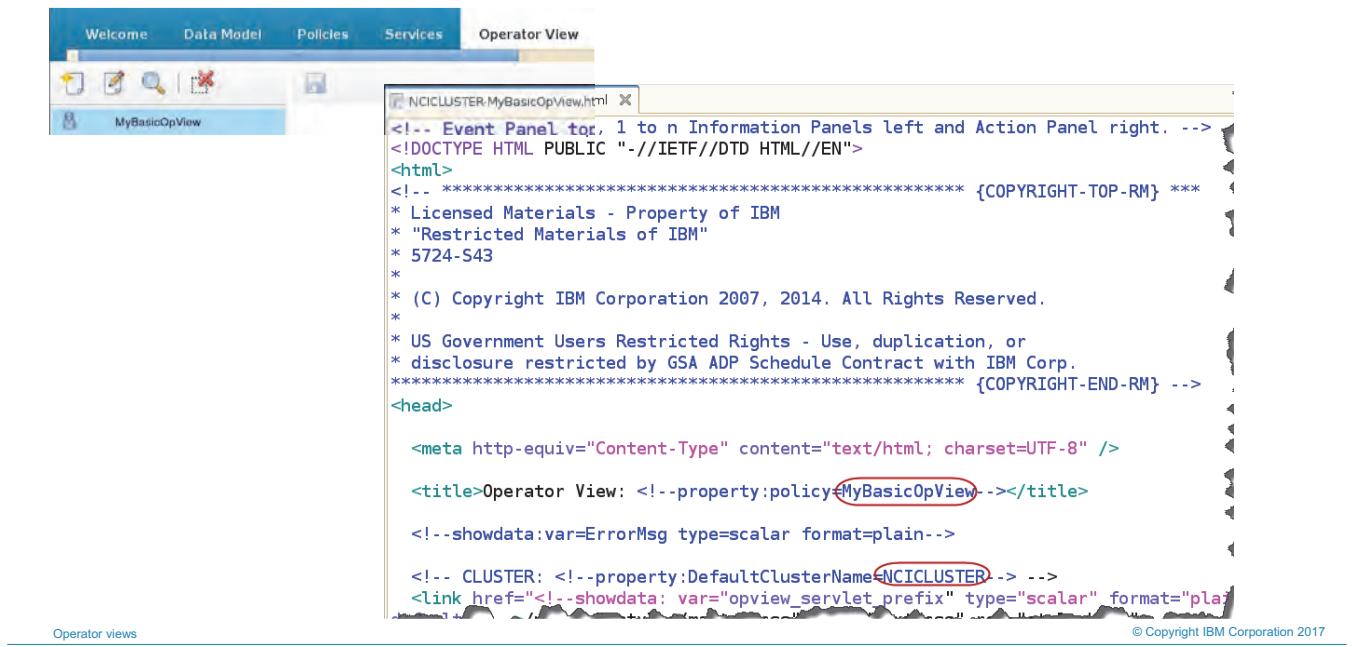
The basic operator view policy

```
// This policy generated by Impact Operator View.  
// Modify at your own risk!  
// Once modified, this policy may no longer be configurable  
// through the Impact Operator View GUI.  
  
// INFO PANEL - START  
// each smart tag on display page should have corresponding  
// _OVUpdateTAGVARNAME() function, where TAGVARNAME is the var ;
```

The basic operator view policy

After you configure the operator view, clicking **Save** automatically creates a policy with the name **General Properties screen**. A policy is also generated with **Opview_** added to the beginning of the policy name. A corresponding HTML file called **NCICLUSTER-BasicView.html** is created in **\$NCHOME/opview/displays**.

The basic operator view HTML



The screenshot shows the Netcool/Impact interface with the 'Operator View' tab selected. A window titled 'NCICLUSTER:MyBasicOpView.html' displays the source code of the operator view. The code includes standard HTML tags like <html>, <head>, and <title>, along with specific property tags such as <!--property:policy=MyBasicOpView--> and <!--property:DefaultClusterName=NCICLUSTER-->. The code also contains copyright and disclosure notices.

```
<!-- Event Panel top, 1 to n Information Panels left and Action Panel right. -->
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<!-- ***** {COPYRIGHT-TOP-RM} ****
* Licensed Materials - Property of IBM
* "Restricted Materials of IBM"
* 5724-S43
*
* (C) Copyright IBM Corporation 2007, 2014. All Rights Reserved.
*
* US Government Users Restricted Rights - Use, duplication, or
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
***** {COPYRIGHT-END-RM} -->
<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Operator View: <!--property:policy=MyBasicOpView--></title>

<!--showdata:var=ErrorMsg type=scalar format=plain-->

<!-- CLUSTER: <!--property:DefaultClusterName=NCICLUSTER--> -->
<link href="<!--showdata: var="opview_servlet_prefix" type="scalar" format="plain-->" href="http://10.10.10.10:9043/nci/Opview_Servlet/Opview_BasicView.ipl" type="text/css" rel="stylesheet" />
```

The basic operator view HTML

The HTML file was automatically generated. The **property:policy** tag includes the name of the policy. A few basic tags are required to create a operator view. Every display page must contain at least two property tags:

- One tag that specifies the name of the Netcool/Impact server cluster
- One tag that specifies the operator view policy: Opview_BasicView.ipl

Every operator view display page must contain both of these types of property tags. If you are creating a basic operator view, these tags are automatically inserted when the view is created in the Netcool/Impact GUI.

In advanced operator views, you must manually add the property tags to the corresponding display page.

Running the operator view

- The operator view is a web-based tool that you can launch from a native or Netcool/Webtop Event List
- After you create the view, a URL is created that you can launch in a web browser
- You can display the operator view by opening pasting the URL into web browser
- Clicking the **View Operator View Display** (magnifying glass) icon also shows the Operator View browser data

Running the operator view

Each operator view that is created has a unique URL. You launch or view the operator view in the same ways as any other type of URL, including these examples:

- Opening the URL in a web browser by address or bookmark
- Starting a web browser from the command line and passing the URL as a command-line argument
- Embedding a link to the URL in another web page on the Internet or intranet
- Configuring the Netcool/OMNIbus Event List or another application to launch a web browser and open the URL as a custom tool



Note: Tivoli Netcool Web GUI is a visualization tool that delivers graphical maps, tables, and event lists to remote operations using HTML and Java.

Sample operator view

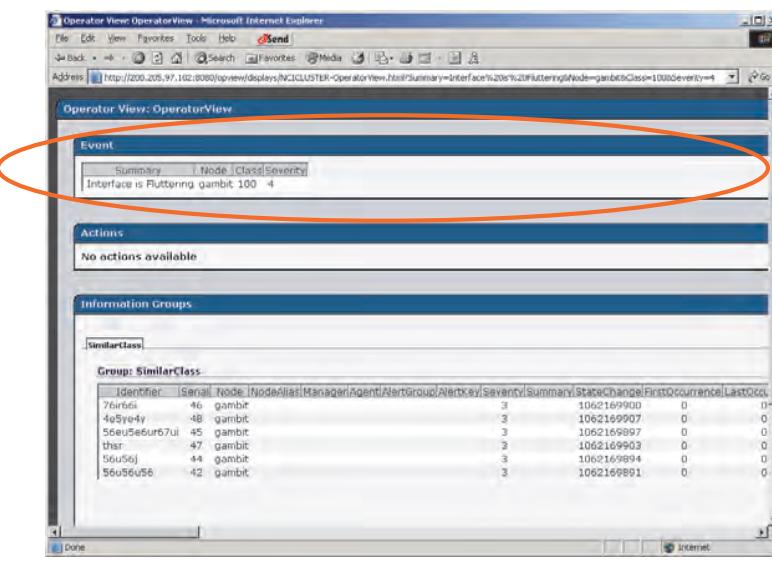
Operator View: MyBasicOpView						
Event						
Actions						
ClearAllEvents						
Information Groups						
AllNodes						
	NODE	IP	NODEID	CUSTID	DEPTID	SERVID
gambit	10.10.10.10	0	1	1	1	
wombat	10.10.10.1	1	2	2	2	
orac	10.10.10.2	2	3	3	3	
muppet	10.10.10.3	3	4	4	0	
moose	10.10.10.4	4	5	5	5	
hal	10.10.10.5	5	6	6	6	
vixen	10.10.10.6	6	7	7	7	
dewey	10.10.10.7	7	8	8	1	

Sample operator view

In this slide, the Event panel shows the **NODE, IP, CUSTID,DEPTID,SERVID** fields for an event that originated with Netcool/OMNIbus. If the Actions panel shows the name of a policy, you can launch this policy by clicking the policy name. The Information Groups panel displays information for the nodes.

The simplest operator views present a basic display of event or business data. More complex operator views can function as stand-alone GUIs in which you can view and interact with event and business data in many ways.

Display event context



Operator views

© Copyright IBM Corporation 2017

Display event context

The **event panel** smart tag is used to display the **Event** panel in an operator view. In advanced operator views, you can freely format and display incoming event values using the scalar and list tags.

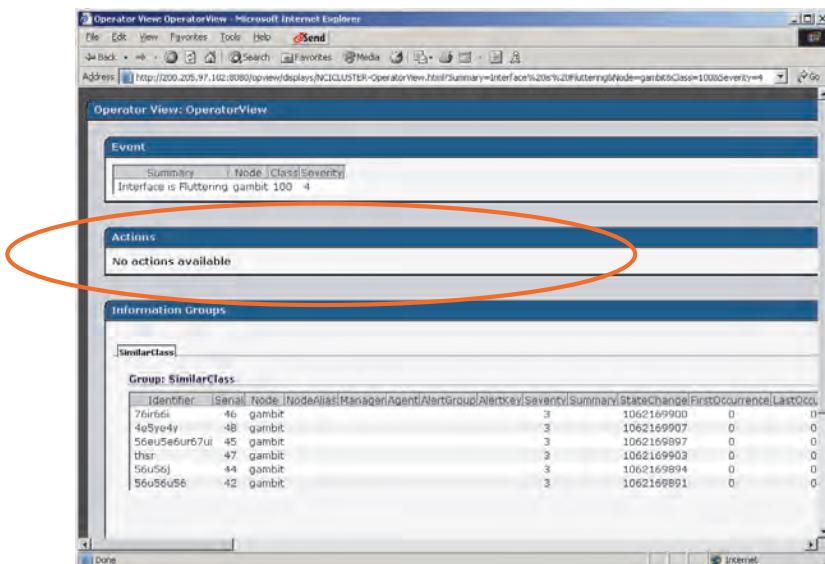
Syntax: The event panel tag has the following syntax:

```
<!--showdata:type=panel-event-->
```

The following example shows how to use the event panel tag in a display page:

```
<html><head>
<title>My Operator View</title>
<!-- <!--property:policy="BasicView" --> -->
<!-- <!--property:DefaultClusterName="NCICLUSTER" --> -->
</head> <body>
<!--showdata:type="panel-event"-->
</body> </html>
```

Display action buttons



Operator views

© Copyright IBM Corporation 2017

Display action buttons

Use the **action panel** tag to show the Actions panel in an operator view. This tag is used in basic display pages, such as those created with the Netcool/Impact GUI. In advanced operator views, you can use the **scalar**, **list**, and **OrgNodes** tags to freely format and show links that trigger other Netcool/Impact policies.



Syntax: The action panel tag has the following syntax:

```
<!--showdata: type=panel-action format=format -->
```

The following example shows how to use the action panel tag in a display page:

```
<html>
<head> <title>Operator View</title> <!-- <!--property:policy=BasicView --> -->
<!-- <!--property:DefaultClusterName=NCICLUSTER --> -->
</head> <body>
<!--showdata: type="panel-action" format="horiz" --> </body>
</html>
```

Display data groups

The screenshot shows an Internet Explorer window titled "Operator View: OperatorView". The main content area is divided into sections: "Event" (with tabs for Summary, Node, Class, Severity), "Actions" (with a message "No actions available"), and "Information Groups". The "Information Groups" section contains a table with the following data:

Group: SimilarClass						
Identifier	Serial	Node	NodeListManager	Agent	AlertGroup	AlertKey
74fb6f	46	gambit			3	1062169905
4b87e4	48	gambit			3	1062169907
56e15e6ur67ui	45	gambit			3	1062169907
ther	47	gambit			3	1062169903
56u56j	44	gambit			3	1062169904
56u56v56	42	gambit			3	1062169901

Display data groups

Use the **information group panel** tag to display the **Information Groups** panel in an operator view. Use this tag in basic display pages, such as those created with the Netcool/Impact GUI. For advanced operator views, you can freely format and show Netcool/Impact data using the **OrgNodes** tag.

The following example shows how to use the information groups panel tag in a display page:

```
<html>
<head><title>My Operator View</title> <!-- <!--property:policy="BasicView" -->
--> <!-- <!--property:DefaultClusterName="NCICLUSTER" --> --></head>
<body>
<!--showdata: type="orgnodes" format="tabbed" var="InfoGroupAdmins" -->
</body>
</html>
```

Modifying operator views beyond the GUI

- You can build operator views using the features in the user interface, but you can realize the power of operator views by modifying the operator view at the page level
- Because the operator view is web-based, you can modify the display of any component
- At the base of the operator view is HTML enhanced with operator view smart tags
- Take advantage of the flexibility of HTML and JavaScript design, and use the smart tags to insert information from Netcool/Impact policies

You can freely configure and modify operator views, both within and outside of the GUI.

Accessing operator view source

- You can find all operator views in the displays in
\$NCHOME/guiserver/webapps/opview/displays
- Operator views are stored as
clustername_Opview_displayname.html
- You can modify files, which are automatically loaded when a user accesses the page from the browser
- Follow these steps for creating a customized operator view:
 1. Create a placeholder operator view from the user interface.
 2. Modify the HTML file as needed to change the look and function of the operator view page
Most of what is automatically created can be deleted
 3. Modify the automatically created operator view policy to control and populate data in the new operator view page

Accessing operator view source

The files used to configure operator views are in the following directory:

\$NCHOME/guiserver/webapps/opview/displays/*.html

The operator view requires a policy (.ipl) file to be defined. The operator view policy is a Netcool/Impact policy that contains the logic required to retrieve and manipulate the data displayed in the view.

This policy is named **Opview_viewname**, where *viewname* is the name of the operator view. There is one such policy for each operator view. When Netcool/Impact handles a request to display an operator view, it runs the corresponding policy.

The content of the operator view policy differs, depending on whether it is associated with a basic view or an advanced view.

When you create a basic operator view using the Netcool/Impact GUI, the GUI automatically creates a corresponding policy that contains all the required content.

Smart tags

- **Scalar tag:** Shows scalar data, either standard variables from the policy, URLs listed in the policy, or an action link
- **List tag:** Similar to a scalar, but the values are a list of data separated by a delimiter

Smart tags

Smart tags are text elements in a display page that contain special instructions for Netcool/Impact. These instructions identify the Netcool/Impact server cluster and specify the policy associated with the operator view. They also specify what kind of data to display in the view and how to display it.

You use the **scalar** tag to display the value of a scalar variable (for example, numerical, Boolean, or string) that is set by an operator view policy. You set this value in the policy with the standard Netcool/Impact policy language (IPL) assignment syntax.

You use the **list** tag to display a list of values that an operator view policy sets. The operator view shows the list as a formatted table. The list is specified in the policy using the standard IPL assignment syntax.

The OrgNodes tag

OrgNodes tag: Similar to list, but handles the display of a two-dimensional list, usually that of an SQL query as a table or series of delimited rows

The OrgNodes tag

You use the **OrgNodes** tag to display a set of Netcool/Impact data items that the operator view policy retrieves from a data source. The operator view shows the data items as a custom table or in per-item format. You retrieve the data items in the policy with the GetByFilter, GetByKey, or DirectSQL functions or with another function that returns a set of data items.

Student exercises



Operator views

© Copyright IBM Corporation 2017

Student exercises

Open your Student Exercise book and perform the exercises for this unit.

Review questions

1. What is an operator view?
2. What are the components of the operator view?
3. Which two HTML tags are required when creating an operator view display page?

Review answers

1. What is an operator view?

An operator view is a custom web-based tool that you use to view events and Netcool/Impact data in real time and to run Netcool/Impact policies based on that data.

2. What are the components of the operator view?

There are two components: a display page and an operator view policy.

3. Which two HTML tags are required when creating an operator view display page?

There are two property tags: the name of the Netcool/Impact server cluster and the name of the operator view policy.

Summary

You now can perform the following tasks:

- Configure an operator view from the user interface
- Customize an operator view in the HTML files and Netcool/Impact policies

Unit 12 Working with web services

IBM Training



Working with web services

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes how to configure web services using Netcool/Impact.

Reference: *IBM Tivoli Netcool/Impact DSA Reference Guide*

Objectives

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

- Describe the Web Services DSA
- Configure a web service using the Web Services wizard

Lesson 1 Web services

IBM Training

IBM

Web Services

© Copyright IBM Corporation 2017
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 work with web services in Netcool/Impact.

The Web Services DSA

- The Web Services DSA provides an interface between the Netcool/Impact systems and external systems, applications, and devices providing web services
- Messages are passed using SOAP (Simple Object Access Protocol) messaging
- Message format is defined by a compiled WSDL (web service description language) file that is provided by the creator of the external web service
- The Web Services DSA operates on a request-and-response mechanism
- The Web Services DSA is automatically installed when you install Tivoli/Netcool Impact

The Web Services DSA

A **web service** is a container for a set of system functions that have been exposed to the web using web-based protocols. The following web services-related functions are used with the Netcool/Impact Web Services data source adapter (DSA):

- WSInvoke
- WSInvokeDL
- WSNewArray
- WSNewEnum
- WSNewObject
- WSNewSubObject
- WSSetDefaultPKGName
- WSDMGetResourceProperty
- WSDMInvoke
- WSDMUpdateResourceProperty

With these functions, you can send messages to a web service provided by another application and handle the message replies.

For more information on web services and the Web Services DSA, see the *Netcool/Impact DSA Reference Guide*.

Using the Web Services DSA: Prerequisite tasks

Complete the following tasks before you use the web services features of the DSA:

- Compile WSDL files
- Create and configure a web services listener
- Write policies that send messages to a web services interface
- Write policies that handle incoming SOAP XML messages

Using the Web Services DSA: Prerequisite tasks

You must complete the tasks listed on this slide before you can use the Web Services DSA. Netcool/Impact Version 6.1 provides a Web Services wizard that completes these steps automatically.

Netcool/Impact uses the Web Services DSA policies to exchange data with external systems, devices, and applications using web services interfaces.

The Web Services wizard

- You can generate the web service policy with the Web Services wizard
- The wizard automatically compiles and deploys the WSDL file

The Web Services wizard

Follow the windows provided in the Web Services wizard to compile and deploy the WSDL and then generate a policy.

Web service example

The screenshot shows a web browser window with the title 'WebserviceX.NET :: XML Web S...'. The address bar contains 'www.webservicex.net/new'. Below the browser is a navigation bar with 'IBM' and 'Most Visited' links, and menu items 'WEBSERVICEX.NET', 'Home', 'Webservices', and 'Contact'. A main content area features a large button labeled 'Explore'. Above the button, the text reads: 'Explore services for all devices' and 'The WebserviceX.NET Data Protocol is a SOAP-inspired technology for reading, writing, and modifying information on the web.' Below the 'Explore' button are several service categories with icons: Business and Commerce (chart), Messaging (envelope), Standards and Lookup Data (database), Value Manipulation / Unit Converter (barcode), Graphics and Multimedia (image), Other Web Services (cloud), and Utilities (cloud with plus). At the bottom of the page, there are copyright notices: '© 2016 - WebserviceX.NET' and '© Copyright IBM Corporation 2017'.

Web service example

This web service provides current weather and weather conditions for major cities around the world. The following information is provided:

- Endpoint: <http://www.webservicex.net/globalweather.asmx>
- WSDL location: <http://www.webservicex.net/globalweather.asmx?wsdl>

Web services description language (WSDL)

```
<?xml version="1.0" encoding="UTF-8" ?>  
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"  
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"  
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://www.webserviceX.NET"  
    xmlns:s="http://www.w3.org/2001/XMLSchema"  
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"  
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"  
    targetNamespace="http://www.webserviceX.NET"  
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">  
.....  
.....
```

Web services description language (WSDL)

The Web Services Description Language defines **services** as collections of network endpoints, or ports. The WSDL specification provides an XML format for documents for this purpose. In other words, a WSDL specification describes the public interface to the web service.

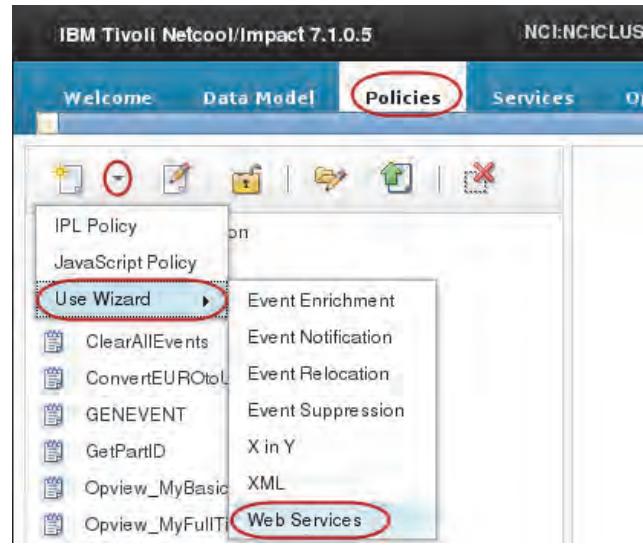
WSDL service information

```
<wsdl:service name="GlobalWeather">
  <wsdl:port name="GlobalWeatherSoap" binding="tns:GlobalWeatherSoap">
    <soap:address location="http://www.webservicex.net/globalweather.asmx" />
  </wsdl:port>
  <wsdl:port name="GlobalWeatherSoap12" binding="tns:GlobalWeatherSoap12">
    <soap12:address location="http://www.webservicex.net/globalweather.asmx" />
  </wsdl:port>
  <wsdl:port name="GlobalWeatherHttpGet" binding="tns:GlobalWeatherHttpGet">
    <http:address location="http://www.webservicex.net/globalweather.asmx" />
  </wsdl:port>
  <wsdl:port name="GlobalWeatherHttpPost" binding="tns:GlobalWeatherHttpPost">
    <http:address location="http://www.webservicex.net/globalweather.asmx" />
  </wsdl:port>
</wsdl:service>
```

WSDL service information

This slide and the previous slide provide the WSDL information for the global weather web service. The information in the WSDL is used as input data when you create the web service policy in Netcool/Impact.

Creating a web service policy with the wizard



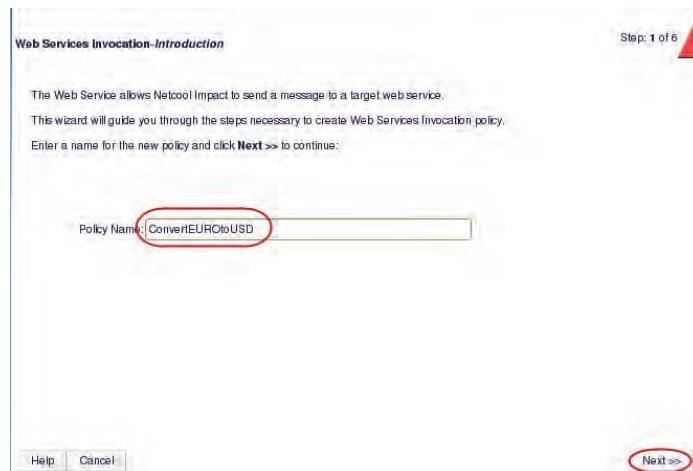
Web Services

© Copyright IBM Corporation 2017

Creating a web service policy with the wizard

The Web Services wizard is in the Wizards panel in the Netcool/Impact GUI.

Introduction window



Web Services

© Copyright IBM Corporation 2017

Introduction window

The Introduction window prompts for a policy name. Select a name that reflects the type of service this web service provides.

WSDL File and Client Stub window

Web Services Invocation - WSDL File and Jar File Step: 2 of 5

WSDL File

URL or Path to WSDL: (This field is highlighted with a red circle)

Jar File

Select a previously generated jar file for the WSDL file. Choosing a value from the Jar File dropdown will dynamically update the Package Name input field.

Jar File: (This field is highlighted with a red circle)

Package Name:

Provide a package name for the new jar file:

Package Name:

(This button is highlighted with a red circle)

Web Services

© Copyright IBM Corporation 2017

WSDL File and Client Stub window

The second window prompts you for a pointer to the WSDL file, either as a URL or as a file stored in the local directory structure. Provide the package name as described in the WSDL file when this web service is created.

Redeploying the impact application EAR

Web Services Invocation - Web Service Name, Port and Method

Step: 3 of 6

General Web Service Information:

The provided WSDL supports the following web services, portTypes and methods. Please choose the web service name and method name you would like to invoke.

Web Service:

Web Service Port Type:

Web Service Method:

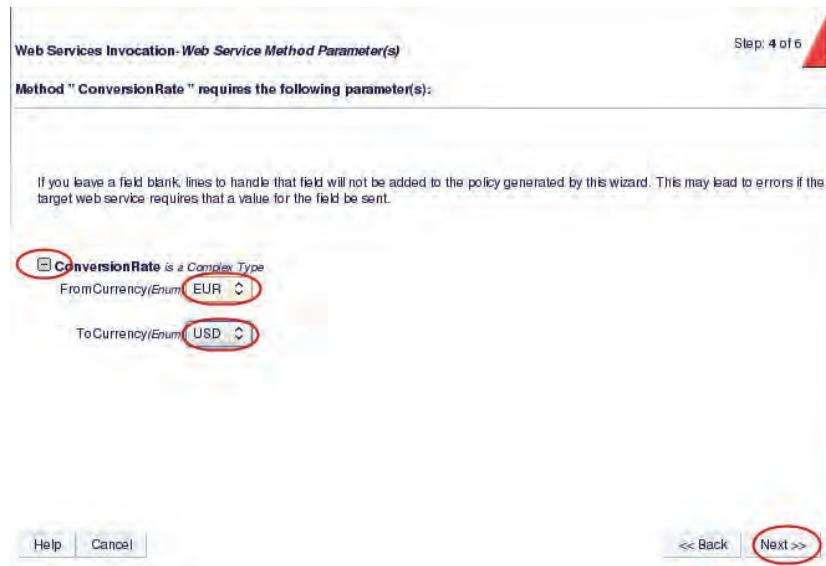
Web Services

© Copyright IBM Corporation 2017

Redeploying the impact application EAR

The **Web Service**, **Web Service Port Type**, and **Web Service Method** fields are automatically populated based on the supplied WSDL.

Web service method parameters window



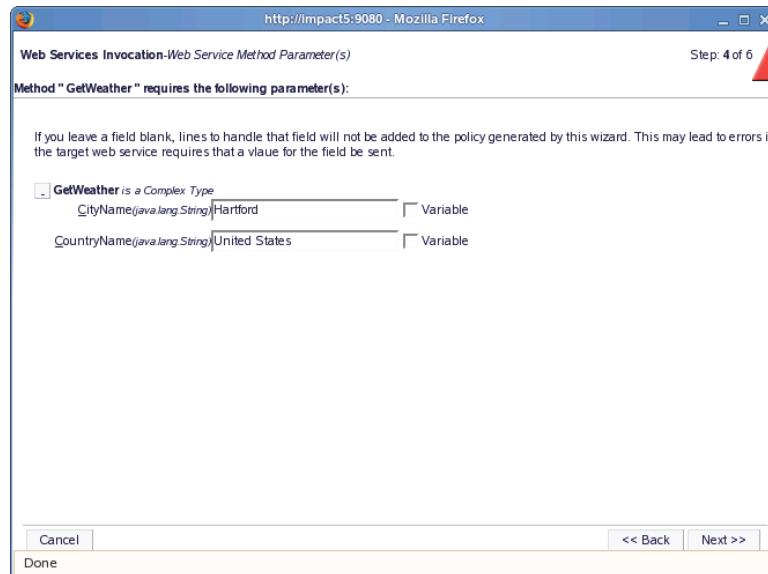
Web Services

© Copyright IBM Corporation 2017

Web service method parameters window

The fourth step prompts for parameters. Click the plus sign (+) icon to expand the option. The parameters vary depending on the web service.

Web Service Method Parameters window (continued)



Web Services

© Copyright IBM Corporation 2017

Web Service Method Parameters window (continued)

Two parameters are required for this web service: *CityName* and *CountryName*.

Web Service End Point window

Web Services Invocation- Web Service End Point And Security

Step: 5 of 6

Web Service End Point

End point is the address provided by the service for access to the target SOAP interface.

The target WSDL specified the following end point for the service:

Edit

End Point URL:

Web Service Security

Enable web service security

Authentication Type: HTTP user name authentication SOAP message user name authentication

Username:

Password:

Next >>

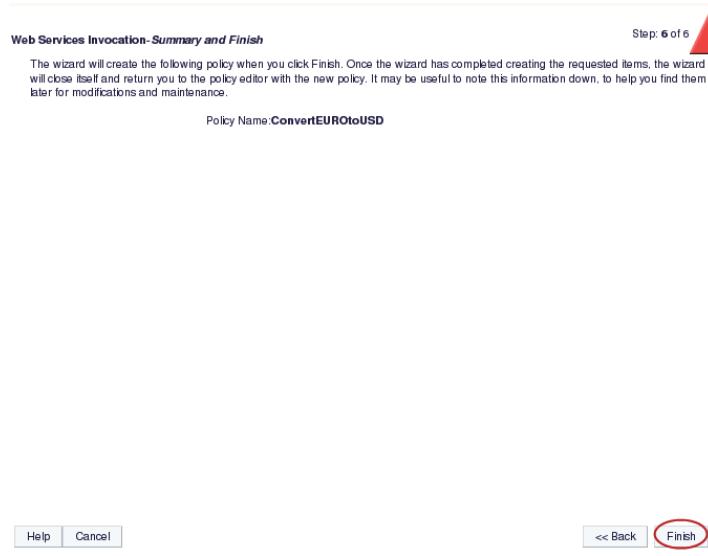
Web Services

© Copyright IBM Corporation 2017

Web Service End Point window

In Step 5, type the endpoint URL as supplied in the WSDL file.

Summary and Finish window



Web Services

© Copyright IBM Corporation 2017

Summary and Finish window

Click the **Finish** button on the Summary and Finish window to deploy the WSDL file and create the policy.

Web service policy generated

The screenshot shows a window titled "ConvertEUROtoUSD" containing a code editor with the following content:

```
//This policy generated by Impact Wizard.  
  
//This policy is based on wsdl file at /labfiles/CurrencyConvertor.asmx  
  
log("Start policy 'ConvertEUROtoUSD'...");  
//Specify package name as defined when compiling WSDL in Impact  
WSSetDefaultPKGName("CurrencyConvertor");  
  
//Specify parameters  
ConversionRateDocument=WSNewObject("net.webservicex.www.ConversionRateDocument");  
_ConversionRate=WSNewSubObject(ConversionRateDocument, "ConversionRate");  
  
_ToCurrency = WSNewEnum("net.webservicex.www.Currency", "USD");  
_ConversionRate["ToCurrency"] = _ToCurrency;  
_FromCurrency = WSNewEnum("net.webservicex.www.Currency", "EUR");  
_ConversionRate["FromCurrency"] = _FromCurrency;  
  
WSParams = {ConversionRateDocument};  
  
//Specify web service name, end point and method  
WSService = 'CurrencyConvertor';  
WSOperation = 'ConvertEUROtoUSD';  
WSOperationPath = "/labfiles/www/webservicex/CurrencyConvertor";
```

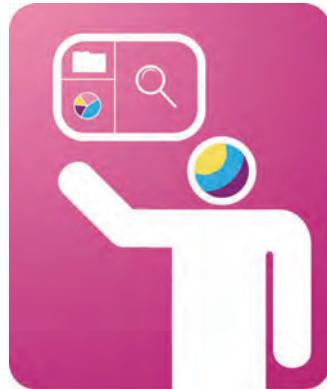
Web Services

© Copyright IBM Corporation 2017

Web service policy generated

After the policy has been generated, you can view and edit it in the Policy editor.

Instructor demonstration



Student exercises



Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What is a web service?
2. How is a WSDL file used?

Review answers

1. What is a web service?

A web service is a container for a set of system functions that have been exposed to the web using web-based protocols.

2. How is a WSDL file used?

A WSDL file describes the public interface to the web service.

Summary

You now can perform the following tasks:

- Describe the Web Services DSA
- Configure a web service using the Web Services wizard

Unit 13 Hibernation, X in Y, and synthetic events

IBM Training



Hibernation, X in Y, and synthetic events

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes how to customize and use Netcool/Impact advanced features.

Reference: *IBM Tivoli Netcool/Impact Solutions Guide*

Objectives

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

- Set up hibernations
- Create and test an X in Y policy
- Use a policy to create and test synthetic events

Hibernations

- **Hibernations** are policies that have been temporarily paused, or *put to sleep*
- Effects of pausing a policy:
 - It is stored internally at its current state
 - All processing is paused until it is awakened by the hibernating policy activator service or by another policy

Hibernations

One of the internal services available in Netcool/Impact is the hibernating policy activator service. This service monitors hibernating policies and awakens them at specified intervals. When configured, this service specifies how often the service checks the policies to see whether they are due to be awakened.

Hibernation data types

- The Hibernation data type is a system data type that stores hibernating policies
- You do not modify hibernation data items using the Netcool/GUI
- You can use the GUI to delete stored hibernations

Hibernation data types

When the **Hibernate** function is called in a policy, it causes Netcool/Impact to store the policy as a Hibernation data item for a certain number of seconds. Typically you do not have to create or modify Hibernation data items using the GUI. However, stored hibernations can be deleted if an error condition occurs and the hibernations are not awakened by the policy activator or by another policy.

See the *Netcool/Impact Solutions Guide* for more information on handling hibernations.

Action keys and the hibernation timeout value

- **Action keys** uniquely identify the hibernation
- The **hibernation timeout value** is the number of seconds that a policy hibernates before the hibernating policy activator can restart it

Action keys and the hibernation timeout value

Key fields used in creating a hibernating policy are action keys and the hibernation timeout value:

- **ActionKey:** String format; uniquely identifies the hibernating policy
- **Reason:** String format; describes the reason for hibernating the policy (optional)
- **TimeOut:** Integer format; number of seconds before the hibernating policy is available to be awoken by the hibernating policy activator

Hibernation functions

Hibernate a policy by calling the **Hibernate** function:

Hibernate (ActionKey, Reason, Timeout)

```
// Call Hibernate and pass an action key and the timeout
// value for the hibernation.
```

```
ActionKey = @Identifier;
Reason = NULL; Timeout = 60;
Hibernate(ActionKey, Reason, Timeout);
```

A shorter version of this policy is:

```
Hibernate (@Identifier, NULL, 60);
```

Note: The action key is used to uniquely identify the hibernation

The **Hibernate** function causes a policy to hibernate. When a policy hibernates, it stops running and is stored as a data item of type Hibernation. At intervals, the hibernating policy activator queries the Hibernation data type and awakens those policies whose timeout value has expired. You can also awaken a hibernating policy from within another policy by using the **ActivateHibernation** function.

In this example, the action key is the value of the Identifier field in an incoming ObjectServer event. This policy will hibernate for 60 seconds.

Hibernation functions (continued)

- **ActivateHibernation (HibernationOrgNode)**
 - To activate a hibernation, retrieve the hibernation in OrgNode
- **GetHibernatingPolicies (StartActionKey, EndActionKey, MaxNum)**
 - Look up hibernations individually or in ordered groups
 - There can be a reason behind ActionKeys
- **RemoveHibernation (ActionKey)**
 - Always remove the hibernation
 - Expiration does not remove them

Hibernation functions (continued)

The **ActivateHibernation** function continues running a policy that was previously put to sleep using the Hibernate function. Netcool/Impact continues the policy at the statement that follows the Hibernate function call. After the policy finishes running, Netcool/Impact returns to the original policy and processes any remaining statements that it contains.

The **GetHibernatingPolicies** function retrieves data items from the Hibernation data type by performing a lexicographical search of action key values.

The **RemoveHibernation** function deletes a data item from the Hibernation data type. To remove a hibernation, call RemoveHibernation and pass the action key for the data item as an input parameter.

Retrieving hibernations

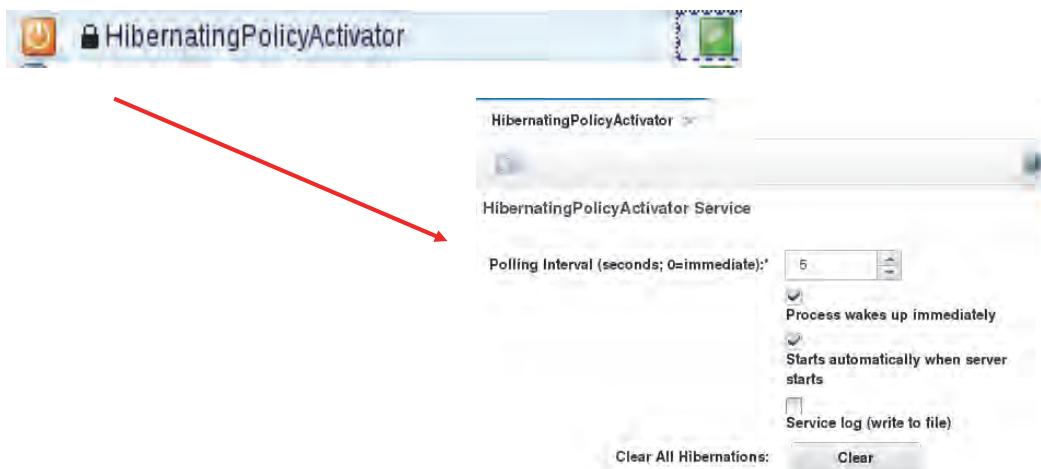
- The method by which you get data items from the Hibernation data type is called ***retrieving hibernations***
- You must retrieve a hibernation before you can awaken it
- Retrieve hibernations by these methods:
 - Action key search
 - FilterRetrieving
- Use the GetHibernatingPolicies function to retrieve hibernations
- Using a lexicographical search of action key values, GetHibernatingPolicies returns an array of Hibernation data items whose action keys fall within the specified start and end action keys

Waking a hibernation

You can awaken a hibernation by using one of the following methods:

- Retrieve the hibernation using these methods:
 - GetHibernatingPolicies or
 - GetByFilter
- Call ActivateHibernation

The HibernatingPolicyActivator Service



The HibernatingPolicyActivator

The HibernatingPolicyActivator service monitors hibernating policies and awakens them at specified intervals. When this service is configured, it specifies how often the service checks the policies to see whether they are due to be awakened.

Setting a hibernation

```
Policy Name: SetHibernation

ActionKey = "TestHibernation:" + getdate();

log("Hibernating with " + ActionKey);
Hibernate(ActionKey, "TestingHibernation", 60);

log("Returned from Hibernation : " + ActionKey + " Reason: " + WakeupReason);

// Reason Can be "EscalationRooster"
// Always remove the hibernation when finished

RemoveHibernation(ActionKey);
log("HIBERNATION: Hibernation removed, Action Key: " + ActionKey);
```

Setting a hibernation

To hibernate a policy, call the **Hibernate** function and pass an action key and the number of seconds for it to hibernate. The action key can be any unique string you want to use to identify the policy. Typically, you obtain this string by performing any combination of the following tasks:

- Use the value of the **Identifier** field in an incoming ObjectServer event. The ObjectServer generates a unique Identifier value for each event.
- Use the **Random** function to generate a random value.
- Use the **GetDate** function to generate a value based on the current system time. This function is only appropriate for scenarios in which you are processing fewer events per second.

Activating the hibernation

Policy Name:

ActivateHibernation

```
// The parameter ActionKey should be passed when executing this policy
log("HIBERNATION: Action Key Passed : " + ActionKey);

HibernationDataItem = GetHibernatingPolicies(ActionKey, ActionKey, 1);

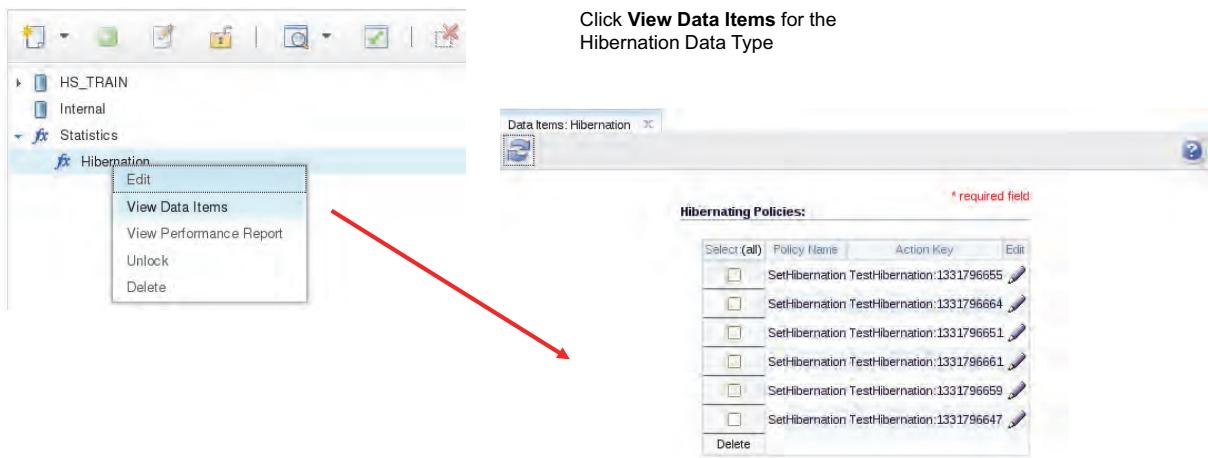
log("HIBERNATION: Hibernation Policy Retrieved: " + HibernationDataItem[0]);
log("HIBERNATION: Activating Hibernation: " + HibernationDataItem[0]);

ActivateHibernation(HibernationDataItem[0]);
```

Activating the hibernation

A hibernation often is activated when it expires. However, hibernations are not just used as a timed pause in a policy. Using this simple policy, you can activate a hibernation manually before it expires. In this case, the policy activates the hibernation based on the passed in parameter *ActionKey*.

Viewing hibernations



Viewing hibernations

You can view hibernations in the **Data Sources and Types** panel. Each hibernating policy shows the policy name and action key. To remove a hibernation, select the check box next to the hibernation and click **Delete**.

X in Y policies: X events in Y seconds

- You can use the X in Y policy to monitor events from OMNIbus over a period of time
- Reasons are varied and include these situations:
 - Security: Multiple login failures or firewall security errors
 - Network problems: Bouncing trunks or links

X in Y policies: X events in Y seconds

If Netcool/Impact detects that an invalid login has occurred for any user more than three times in 30 seconds, it performs the following actions:

- Escalates the alarm to Critical
- Changes the summary to Multiple Login Failures on the account

This process tests for duplicate events.

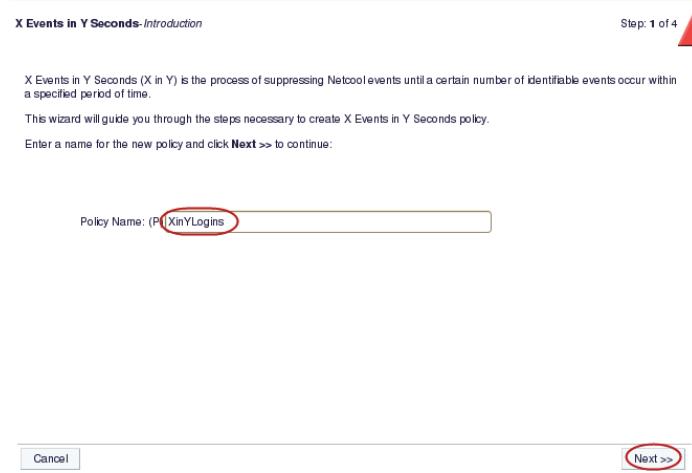
The X in Y wizard

- X in Y is a complex policy
- Using the wizard is a good way to review this type of policy
- Additional functions can be added to X in Y, such as altering the **Summary** field

The X in Y wizard

Because the X in Y policy is common but complex, there is a wizard for its creation. The efficiency of the X in Y policy is critical, and it can affect the performance of the Netcool/Impact server if it is incorrectly configured. When making any changes to the policy, keep performance issues in mind.

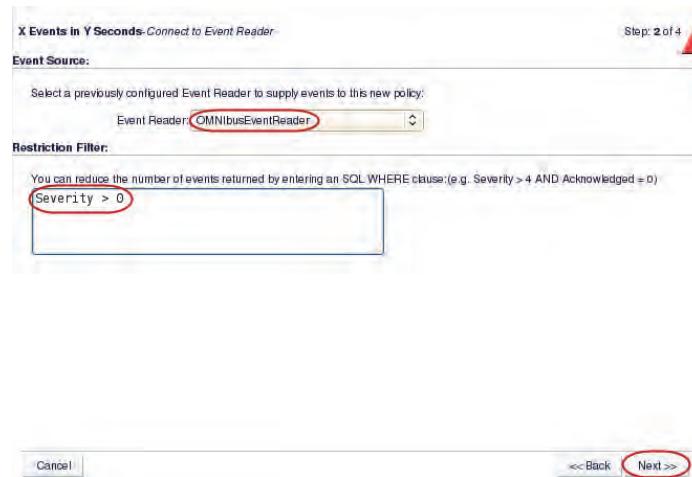
X in Y wizard introduction window



X in Y wizard introduction window

The wizard **Introduction** window provides a brief description of the purpose of the wizard and prompts for a policy name.

X in Y wizard Connect to Event Reader window



X in Y wizard Connect to Event Reader window

The second step is to connect to the event reader. In this case, the default is **OMNIBusEventReader**. There is also a **Restriction Filter** panel. The restriction must be exact.

X in Y wizard Set Thresholds window

X Events in Y Seconds- Set Thresholds Step: 3 of 4

Event Volume Threshold

Set a threshold for the number of events that must occur:

Event Threshold: (E) events

Time Threshold

Set the length of the time window in seconds:

Time Window: (W) seconds

Updated Event Properties

When the threshold is crossed, the event will be updated. Set the Severity level for this new event:

Severity: (S)

X in Y wizard Set Thresholds window

In the third step, set the policy thresholds:

- Event volume threshold
- Time threshold
- Updated event properties

Enter the information in the fields, and click **Next**.



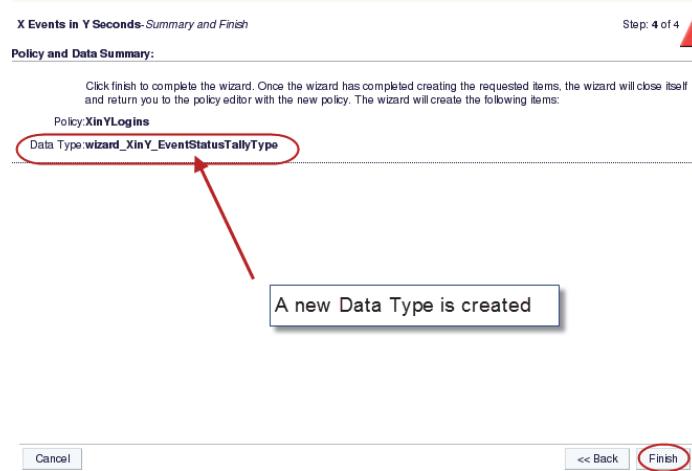
Note: The X value is the event threshold, and the Y value is the time window.

The **Severity** field is set to a value specified when the threshold is breached.

X in Y wizard Summary and Finish window

EventReaderFilter:

- Severity > 0
- Example of A summary: Attempt to login as user.*failed

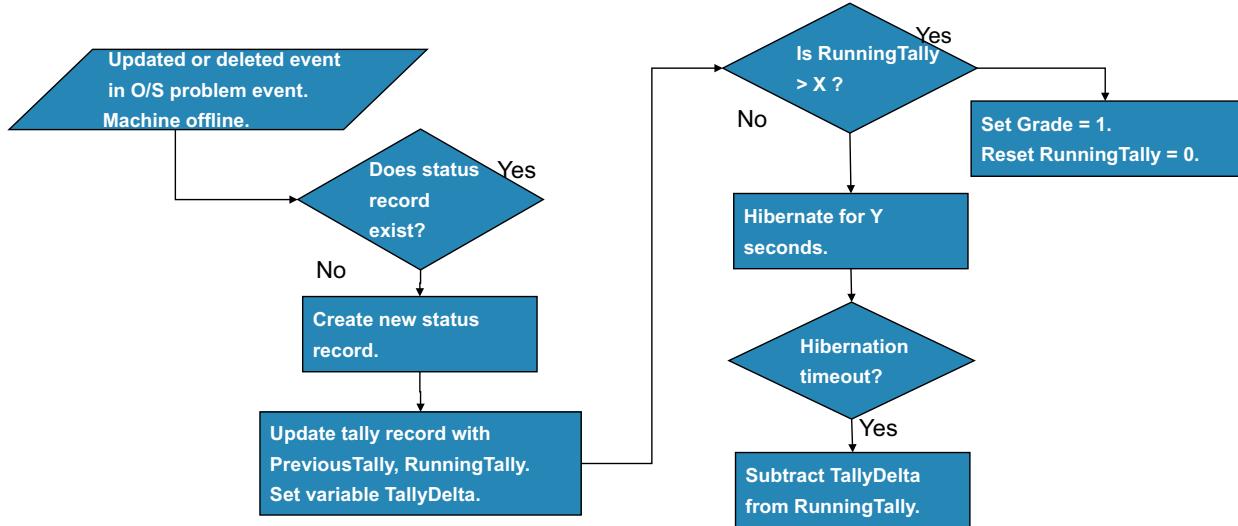


X in Y wizard Summary and Finish window

When the wizard has finished, it shows the policy name and data types that it has created. The wizard creates the internal data type **wizard_XinY_EventStatusTally**.

Static information must be stored to track the status of each alarm over time. An event updates the fields, **RunningCount** which is the current number of events in the Y window, and **PreviouslyProcessedCount**, which can be identified when multiple increments are made to **@Tally**. The number of previously processed events is stored in the **Count** field.

Counting is necessary because multiple events can be deduplicated in the time between event reader polling, causing the **Tally** field to increase.

X in Y program flow**X in Y program flow**

An event is hibernated in Netcool/Impact during the sliding window Y. If during this time, a running tally of an event breaches the threshold Y, the escalation is processed. If the event threshold is not breached before the end of the window, the event is removed from hibernation.

Synthetic events

- Synthetic events are events which are manually or programmatically created
- A new event created from Netcool/Impact has these characteristics:
 - Is achieved by accessing the ObjectServer's alerts.status table
 - Can be used to manipulate the events in alerts.status table
 - Can be created using the NewEvent or NewObject functions
- Synthetic events are used for these purposes:
 - Test new policy function
 - Reproduce production problems
 - Isolate testing of services and other Netcool/Impact tools
- Test services are created to handle synthetic events for test purposes

Ensure that a field exists in the ObjectServer that identifies the event as a synthetic event. If one does not exist, create a new field that can identify the event as a synthetic event.

The NewEvent function

You use the NewEvent function to create a new event container for a specific event reader

```
MyEvent = NewEvent ("OMNIbusEventReader");  
  
// Set the event field variables  
MyEvent.EventReaderName = "OMNIbusEventReader";  
MyEvent.Identifier = "XYZ123";  
MyEvent.Node = "DB SERVER_01";  
MyEvent.Class = "99999";  
MyEvent.Manager = "Netcool/Impact";  
MyEvent.Acknowledged = 0;  
MyEvent.Severity = 5;  
MyEvent.Type = 0;  
  
//Sends the event to the event reader  
ReturnEvent (MyEvent);
```

The NewEvent function

You can use the **NewEvent** function to create a new event in the ObjectServer that any event reader is connected to. The NewEvent function creates a new instance of an event container.

The values are then set in EventContainer before the event is returned to the ObjectServer. Make sure that the **Identifier** field is properly set to ensure that this event is unique.

The NewObject function

The NewObject function creates a new context

```
MyContext = NewObject () ;

// Set the event field variables
MyContext.Identifier = "XYZ123";
MyContext.Node = "DB_SERVER_01";
MyContext.Class = "99999";
MyContext.Manager = "Netcool/Impact";
MyContext.Acknowledged = 0;
MyContext.Severity = 5;
MyContext.Type = 0;

//Adds context to the table
AddDataItem ("NCOMS_ALERTS", MyContext);
```

The NewObject function

Use the **NewObject** function to add a new event to a data type (table). The **Identifier** field should be unique.

Student exercises



Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What are hibernations?
2. What is an example use of the X in Y policy?
3. How are synthetic events used?

Review answers

1. What are hibernations?

Hibernations are policies that have been temporarily declared inactive.

2. What is an example use of the X in Y policy?

Someone attempts to log in incorrectly 10 times in 20 seconds.

3. How are synthetic events used?

Synthetic events are events created to allow testing without affecting production event processing. Usually, test services are created to handle synthetic events.

Summary

You now can perform the following tasks:

- Set up hibernations
- Create and test an X in Y policy
- Use a policy to create and test synthetic events

Unit 14 Maintenance window management

IBM Training



Maintenance Window Management

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes how Netcool/Impact uses Maintenance Window Management (MWM) to manage events.

Objectives

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

- Describe the purpose of Maintenance Window Management
- Implement Maintenance Window Management
- View Maintenance Window Management events in Netcool/OMNIbus

Maintenance Window Management

- Maintenance window management (MWM) manages Netcool OMNIbus maintenance time windows
- A maintenance time window is a prescheduled period of downtime for a particular asset
- Events that occur during this downtime can be ignored or flagged as maintenance events
- You can create several kinds of maintenance windows:
 - One-time windows: emergency situations
 - Recurring windows: weekly, monthly , every *n*th day

Maintenance Window Management

MWM creates maintenance time windows and ties them to Netcool OMNIbus events that are based on OMNIbus fields values such as Node or Location. Netcool/Impact watches the Netcool OMNIbus event stream and puts these events into maintenance according to the maintenance time windows. OMNIbus version 7.1, 7.2, or higher is required to use this capability.

Maintenance window management (MWM) configuration

MWM configuration components include these items:

- MWM properties
- MWM policy
- MWMActivator service

There are three components used to configure Maintenance Window Management. The Maintenance Window (MWM) policy is automatically triggered by the MWMActivator service. The MWM_properties policy builds the MWM policy parameters.

MWM properties

The MWM policy sets parameters

clearFlag (Boolean)

- TRUE clears the maintenance flag on events when the maintenance window has expired
- FALSE leaves the events tagged as in maintenance after the maintenance window expires

flagExistingEvents (Boolean)

- TRUE flags events as *in-maintenance* events, which come in before the maintenance window (and during)
- FALSE flags events as *in maintenance* if they come in during the maintenance window

propsContext

- Contains values retrieved from a properties file

The Default value is FALSE

Several variables determine how the MWM policy handles the incoming events: **clearFlag**, **flagExistingEvents**, and **propsContext**.

The MWMProperties policy getProperties function

```
function getProperties(propsContext)
{
    propsContext = newobject();

    //SET YOUR VALUES HERE
    clearFlag = TRUE;
    flagExistingEvents = TRUE;
    //THANKS :)

    propsContext.clearFlag = clearFlag;
    propsContext.flagExistingEvents = flagExistingEvents;
}

/* FUNCTION TESTING
MWM_Properties.getProperties(mwmProps);
log(mwmProps.clearFlag);
log(mwmProps.flagExistingEvents);
*/
```

The MWMProperties policy getProperties function

The MWM_Properties policy defines the **getProperties** function. The **getProperties** function builds a synthetic event using the **newobject function** and returns the event in the **propsContext** field.

MWM policy

- Policy runs from a policy activator
- Policy checks for active maintenance windows and places OMNIbus events into maintenance
- Policy optionally takes events out of maintenance when maintenance windows expire
- Policy calls the MWM_Properties policy to get maintenance flags

MWM policy

Based on the default values, the following scenario is possible:

Given: ClearFlag = TRUE and flagExistingEvents = TRUE

An event for Node TEST1 arrives at 6:30pm.

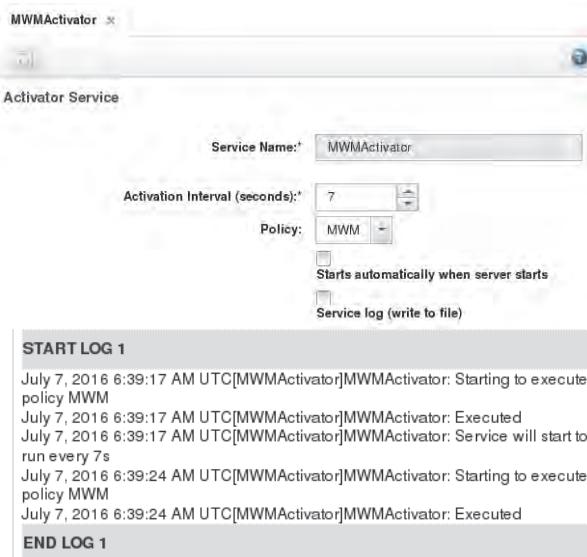
A maintenance window for TEST1 starts at 7:00pm.

The event at 6:30 pm will be tagged **in maintenance**, because flagExistingEvents is TRUE.

When the maintenance window expires, the in-maintenance flag will be removed because the ClearFlag is set to TRUE.

You can define your own policy activator policy to replace the MWM policy.

The MWMActivator Service



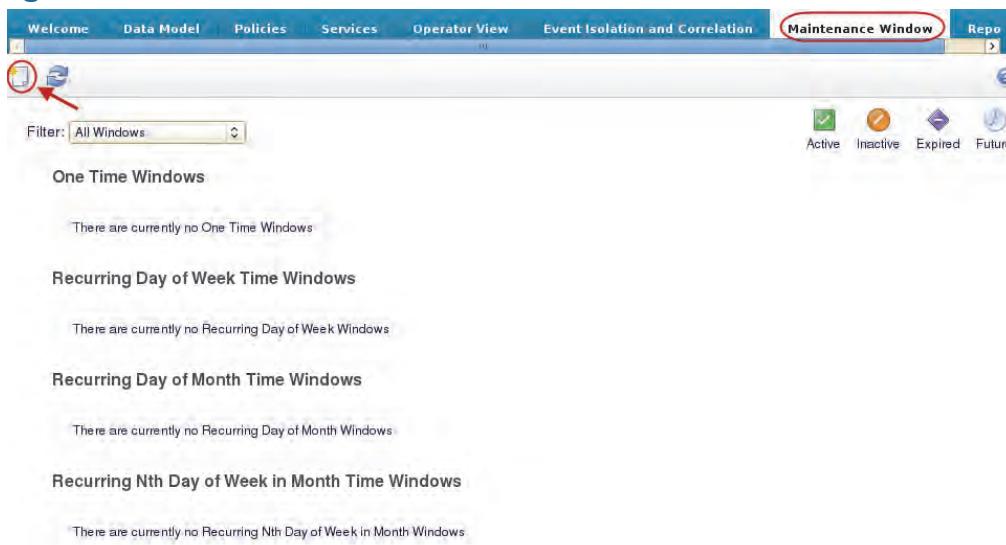
Maintenance Window Management

© Copyright IBM Corporation 2017

The MWMActivator Service

The **MWMActivator** service by default checks OMNIbus every seven seconds for events that require maintenance. You can change the time interval and the policy.

Creating a maintenance window



The screenshot shows the 'Maintenance Window' management interface. The top navigation bar includes links for Welcome, Data Model, Policies, Services, Operator View, Event Isolation and Correlation, Maintenance Window (which is highlighted with a red oval), and Repo. Below the navigation bar is a toolbar with icons for New, Edit, Delete, and Help. A filter dropdown is set to 'All Windows'. The main content area is divided into sections for 'One Time Windows', 'Recurring Day of Week Time Windows', 'Recurring Day of Month Time Windows', and 'Recurring Nth Day of Week in Month Time Windows'. Each section displays a message indicating there are currently no windows of that type.

Maintenance Window Management

© Copyright IBM Corporation 2017

Creating a maintenance window

The Maintenance Window Management tool is located under **Troubleshooting and Support > Event Automation**. The maintenance options show as links in the blue bar under the window title. In this case, **Add One Time** was selected. A **Start Time** and **End Time** indicate the duration of the maintenance window. The available alerts.status field matching parameters are **Node**, **Alert Group**, **Alert Key**, and **Location**.

Viewing the maintenance window

The screenshot shows a software interface for managing maintenance windows. At the top, there are icons for New, Open, Save, and Close. Below them is a toolbar with buttons for Active (green), Inactive (orange), Expired (blue), and Future (light blue). A filter dropdown is set to 'All Windows'. The main area is titled 'One Time Windows' and contains a table with two rows. The columns are: Select (all), Suppression Filter, Start Time, End Time, Time Zone, Edit, and Status. The first row has a checkbox checked under 'Select (all)', a suppression filter of 'AlertGroup = 'Link'', a start time of 'Sat 25 Jun 2016 04:39:00 AM UTC', an end time of 'Mon 27 Jun 2016 04:39:00 AM UTC', a time zone of '(UTC-05:00) New York, Toronto', and a status icon (green checkmark) circled in red. The second row has a 'Delete' button. The bottom right of the table has a 'Status' button.

Viewing the maintenance window

The color of the status icon indicates whether the window is active (green), expired (red), or has not started yet (blue, in the future).

When Impact servers are clustered, the MWM GUI interacts with the primary Impact server in the Impact cluster. Data is stored in the HSQL database of the primary Impact server. Data is not replicated between HSQL instances in the Impact cluster. If the primary Impact server changes, the MWM GUI and policies cannot interact with previously entered data.

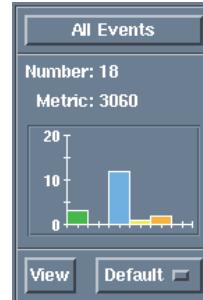
Additionally, overlapping time windows can cause events to be temporarily flagged as out of maintenance. If this situation occurs, the event is flagged as in maintenance the next time the MWMActivator service runs. However, if the clearFlag = FALSE, then the event is never marked as out of maintenance.

Maintenance Window Management requires the default cluster name to be NCICLUSTER.

Starting The Netcool/OMNibus event viewer

- Type the command to start Netcool/OMNibus:

```
tivoli@tip01:/opt/IBM/tivoli/netcool/omnibus/bin>
./nco_event
```
- Username: **root**
- Password: **object00**
- Expand the window so that **Event List** shows
- View **All Events**



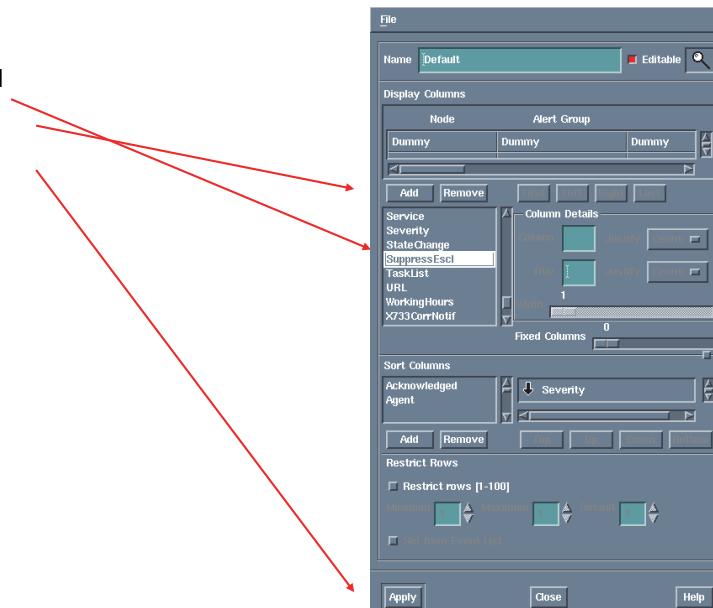
There are several ways to view events in Netcool/OMNibus.

Netcool/OMNIbus view builder

1. Select **SuppressEscI**

2. Click **Add**

3. Click **Apply**



Maintenance Window Management

© Copyright IBM Corporation 2017

Netcool/OMNIbus view builder

You can modify the event viewer according to your own specifications. For example, you can add or remove fields. In this case, **SuppressEscI** is added to the view. Clicking **Apply** at the bottom of the screen modifies the view to include the selected field.

Viewing maintenance events in Netcool/OMNIbus

Maintenance

SuppressEscI flag set to 6

Normal

SuppressEscI flag set to 0

Count	Type	ExpireTime	Agent	Manager	Suppr./EscI.
2	Problem	Not Set		SecurityWatch	Normal
1	Problem	Not Set		SecurityWatch	Normal
3	Problem	86:430	OMNIbus SelfMonitoring	OMNIbus Self Monitoring @NCO	Normal
697	Information	90	OMNIbus SelfMonitoring	OMNIbus Self Monitoring @NCO	Normal
697	Information	90	OMNIbus SelfMonitoring	OMNIbus Self Monitoring @NCO	Normal
697	Information	90	OMNIbus SelfMonitoring	OMNIbus Self Monitoring @NCO	Normal
140	Information	330	OMNIbus SelfMonitoring	OMNIbus Self Monitoring @NCO	Normal
140	Information	330	OMNIbus SelfMonitoring	OMNIbus Self Monitoring @NCO	Normal
140	Information	330	OMNIbus SelfMonitoring	OMNIbus Self Monitoring @NCO	Normal

Student exercises



Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What is the purpose of the **getProperties** function?
2. Which **alerts.status** field holds the maintenance status value?
3. True or False: **clearFlag** set to TRUE means the **in maintenance** flag is not set during the maintenance window.

Review answers

1. What is the purpose of the **getProperties** function?

*Located in the MWM_Properties policy, the **getProperties** function builds a synthetic event using the **newobject** function and returns the event in the **propsContext** field.*

2. What **alerts.status** field holds the maintenance status value?

suppressEscI

3. True or False: **clearFlag** set to TRUE means the **in maintenance** flag is not set during the maintenance window.

*False. The **clearFlag** causes the **in maintenance** flag to be removed after the maintenance window expires.*

Summary

Now that you have completed this unit, you can perform the following tasks:

- Describe the purpose of Maintenance Window Management
- Implement Maintenance Window Management
- View Maintenance Window Management events in OMNIbus

Unit 15 Command-line tools and self-monitoring

IBM Training



Command-line tools and self-monitoring

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes how Netcool/Impact uses command-line tools and self-monitoring to manage the Netcool/Impact server.

Reference: *IBM Tivoli Netcool/Impact Administration Guide*
IBM Tivoli Netcool/Impact Policy Reference Guide

Objectives

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

- Describe Netcool/Impact command-line tools
- Describe the Netcool/Impact command-line manager
- Encrypt a policy from the command line
- Enable self-monitoring

Lesson 1 Command-line tools

IBM Training

IBM

Command-line tools

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This lesson lists the tools available for Netcool/Impact from the command line.

String and policy tools

- String tool
 - **nci_crypt**: encrypts a string using the Netcool/Impact encryption algorithm
`nci_crypt string`
 - **nci_encryptpolicy**: encrypts a policy
Encryption is accomplished through the use of a creator-supplied password
`nci_encryptpolicy NCI object00 /tmp/test.ipl /tmp/test_enc.ipl` (`test_enc.ipl` is the encrypted result)
- Policy tools
 - **nci_trigger**: starts a policy from the command line
`nci_trigger NCI tipadmin/object00 POLICY_01`
 - **nci_policy**: checks syntax or uploads a policy to a project
 - **Check Syntax**: `nci_policy NCI syntax /path/to/policy.ipl`
 - **Upload Policy**: `nci_policy NCI1 push user password /path/to/policy.ipl --project MyProject`

These tools are used with string data or with policies. JSSE services must be linked into the Netcool/Impact application for encryption to function properly.

Export and import tools

- **nci_export:** exports data source, data type, service, policy, and project information from an instance of the Impact server to a specified directory

```
./nci_export NCI /tmp/NCI_export      (entire server)  
./nci_export NCI --project ClassProject /tmp/NCI_export  (single project)
```

- **nci_import:** imports data that was previously exported from an instance of the Impact server
`nci_import server_name directory` (*use directory created in export*)

Export and import tools

You use these tools to back up and restore a copy of the objects that were created in a specific Netcool/Impact instance.



Note: This backup is not a full Netcool/Impact backup that includes Tivoli Integrated Portal components.

IBM Training



Cluster status tool

- Cluster state can be viewed in a browser by typing the following URL:

http://<Impact_hostname>:9080/nameserver/services

- The example shows the currently defined nameserver instances
- Cluster Status can also be viewed using the command line or the SelfMonitoring service in the GUI

RPL#	SELF	STATUS	URL
0	****	UP	http://jazz01.tivoli.edu:9080/nameserver/services

Last 100 commands received:

RECORD	RECV	TIME	PARAMETERS
4543	1467874451248	0	ID=aiVPIvjTIG7yTdr2Xug2j1q ACCS=api ACTN=Xgetdesat PROD=Impact SERV=NCICLUSTER DESI=0 TYPE=RESULT DATA=192.168.100.165:53703:NCI SHOWSYS=false SUPPRESS=true TERSE=false
4542	1467874448591	0	ID=mW4U8lOkN6CsEi8144d1Qtj ACCS=api ACTN=Xlistbind PROD=Impact SERV=NCICLUSTER DESI=-1 TYPE=RESULT DATA=192.168.100.165:53703:NCI SHOWSYS=false SUPPRESS=true TERSE=false
4541	1467874448588	0	ID=mW4U8lOkN6CsEi8144d1Qtj ACCS=api ACTN=Xlistserv PROD=Impact DESI TYPE=RESULT DATA=NCICLUSTER SHOWSYS=false SUPPRESS=true TERSE=false
4540	1467874448586	0	ID=mW4U8lOkN6CsEi8144d1Qtj ACCS=api ACTN=Xlistserv PROD=Impact DESI TYPE=RESULT DATA=NCICLUSTER SHOWSYS=false SUPPRESS=true TERSE=false
4539	1467874448584	0	ID=mW4U8lOkN6CsEi8144d1Qtj ACCS=api ACTN=Xlistserv PROD=Impact DESI TYPE=RESULT DATA=NCICLUSTER SHOWSYS=false SUPPRESS=true TERSE=false
4538	1467874448581	0	ID=mW4U8lOkN6CsEi8144d1Qtj ACCS=api ACTN=Xlistserv PROD=Impact DESI TYPE=RESULT DATA=NCICLUSTER SHOWSYS=false SUPPRESS=true TERSE=false

Command-line tools and self-monitoring

© Copyright IBM Corporation 2017

Cluster status tool

This tool is useful if more than one Netcool/Impact server instance is installed. See the *IBM Tivoli Netcool/Impact Administration Guide* for a list of all the command-line tools available.

Command-line manager

- Access the Netcool/Impact server from the command-line interface:
 - To start and stop services
 - To configure service parameters
- To connect to the command-line manager:
 - Specify the host name
 - Specify the command line port (*default is 2000*)
 - Optionally, specify if the service starts automatically when the impact server starts
`telnet <hostname> <commandlineport>`

Command-line manager

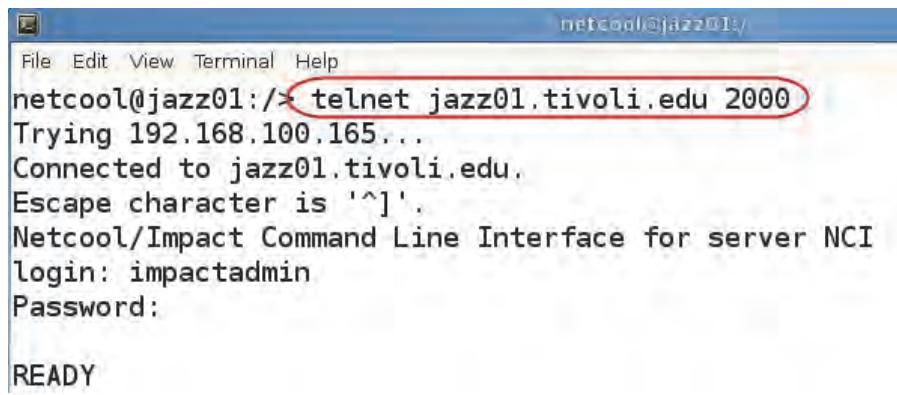
The telnet option is available for remote management or in instances where the GUI is not available. The file **\$NCHOME/etc/NCI_commandlinemanager.props** contains the required **port** number and the **autostartup** option.

Example of an **NCI_commandlinemanager.props** file:

```
impact.commandlinemanager.port=2000
impact.commandlinemanager.autostartup=true
impact.commandlinemanager.classname=com.micromuse.response.service.CmdLineSe
rvice
impact.commandlinemanager.logtofile=false
impact.commandlinemanager.description>No Description
```

Starting the command-line interface

Access the Netcool/Impact server from the command-line interface:



```
netcool@jazz01:/> telnet jazz01.tivoli.edu 2000
Trying 192.168.100.165...
Connected to jazz01.tivoli.edu.
Escape character is '^]'.
Netcool/Impact Command Line Interface for server NCI
login: impactadmin
Password:

READY
```

Starting the command-line interface

Start the command-line manager from a server with telnet enabled. Type the server name for the Netcool/Impact server. For this course, the server name is **tip01.tivoli.edu**.

Telnet prompts you for the Netcool/Impact user and password. The password does not show when you type it.

Command-line manager supported services

The following services can be accessed using command-line manager:

- Event Processor manages events coming into Netcool/Impact
- Email Reader polls host for email messages
- Email Sender configures the local email address to send email
- Policy Activator activates policies at specified intervals
- Hibernating Policy Activator wakes hibernating policies
- Policy Logger manages messages generated by policies during runtime
- OMNIbus Event Reader polls for events from the OMNIbus object server
- Database Event Reader polls a SQL data source at intervals specified
- Database Event Listener monitors an Oracle data source for events
- JMS Listener runs policies as a response to incoming JMS messages

Command-line manager examples

- OMNIbus Event Reader: Clear Queue

```
Update Service set ClearQueue=true where name='OMNIbusEventReader';
```

- OMNIbus Event Reader: Get the Queue Size

```
Select QueueSize from Service where Name='OMNIbusEventReader';
```

- Event Processor: Get Events Waiting to be Processed

```
Select NumThreads from Service where Name='EventProcessor';
```

- Event Processor: Clear the Event Queue

```
Update Service set ClearQueue=true where Name='EventProcessor';
```

Command-line manager examples

Many command-line functions are the same for every service. However, there are also unusual functions that are specific to a particular service that you can also query or modify.

See the *IBM Tivoli Netcool/Impact Administration Guide* for a list of all the commands available for each service.

JavaScript in the command-line manager

Switch from IPL to JavaScript in the command-line processor:

- **js<enter>**
- Issue the **script** command to display the current buffer
- Issue the **clear** command to remove any leftover data in the script buffer

```
login: impactadmin
Password:

READY
js
js 1>script
js 1>clear
js 1>script
js 1>
```

JavaScript is also supported in the command-line manager. When the command-line manager is in JavaScript mode, run any of the commands that are listed in the *IBM Tivoli Netcool/Impact Policy Reference Guide*.

Lesson 2 Self-monitoring

IBM Training

IBM

Self-monitoring

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

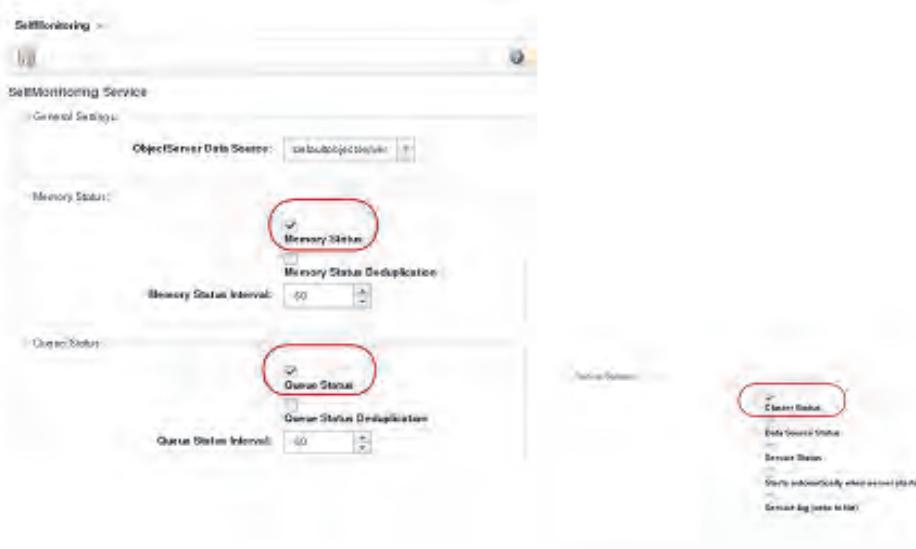
The Netcool/Impact product can self-monitor. For example, when Netcool/Impact is running as a cluster, it provides information on which server is the current primary and which is the current secondary. Netcool/Impact also sends updates when the primary has gone down and one of the secondary servers assumes the primary role.

Self-monitoring overview

- Netcool/Impact can gather performance information and send it to Netcool/OMNibus as events
- The events are the same as other Netcool/Impact events and can be managed the way other events are managed
- In a cluster, self-monitoring operates on a per-server basis
 - Each Netcool/Impact server instance is monitored separately
- You make configuration changes from the command line or the GUI

The Netcool/Impact server can monitor additional resources using the *self-monitoring* service.

Self-monitoring configuration using the GUI



Command-line tools and self-monitoring

© Copyright IBM Corporation 2017

Self-monitoring configuration using the GUI

The **SelfMonitoring Service** is located in the Global repository, which can be added to any project. The ObjectServer Data Source is where the monitoring events are sent. The following resources are monitored if selected:

- **Memory Status:** status events regarding queue conditions of the event readers that are currently running
- **Queue Status:** status of the event readers and listeners currently running
- **Cluster Status:** send events about the status of the cluster to which it belongs
- **Data Source Status:** select to enable the service to send the status when certain conditions occur with a data source
- **Service Status:** service stop and start events are monitored (auto startup is not monitored)

Deduplication is enabled by default. You can also set certain options including deduplication in the **\$NCHOME/etc/servername_selfmonitoring.props** file.

Self-monitoring configuration using the command-line manager

- To start the self-monitoring service, enter the following command:
`UPDATE Service SET Running = true WHERE Name = 'SelfMonitoring';`
- To stop the self-monitoring service, enter the following command:
`UPDATE Service SET Running = false WHERE Name = 'SelfMonitoring';`

Self-monitoring configuration using the command-line manager

The command line provides the ability to monitor the same features as the GUI.

Syntax: Telnet is used to connect to the service. The syntax is as follows:

`telnet <hostname> <commandlineport>`

The default commandlineport is 2000.

Here are some examples of command-line commands:

- To set the Data Source for monitoring, use the command:
`Select DataSource from Service where Name = 'SelfMonitoring';`
- To turn on QUEUE Status monitoring enter the following command:



Note: `UPDATE Service SET EnableQueueStatus = true WHERE Name = 'SelfMonitoring';` When you create the SELECT statements, TRUE equates to the checked box, and FALSE equates to an unchecked box.

See the *IBM Tivoli Netcool/Impact Administration Guide* for more information about specific event fields for each monitored feature.

Self-monitoring events

Node	Alert Group	Summary	Last Occurrence	Count
jazz01	DBStatus	Last 5 mins alerts.journal (inserts): 0	/16/2015 08:00:26 A	143
jazz01	DBStatus	Last 5 mins alerts.status (inserts/deduplications): 22	/16/2015 08:00:26 A	143
jazz01.tivoli.edu	NCI2:Impact	A NCI2:Impact process running on jazz01.tivoli.edu has connected as username	/16/2015 05:55:54 A	1
jazz01.tivoli.edu	Memory Status	Impact's Heap using 197M out of 1200M, Free System Memory Available: 4237M	/16/2015 08:02:18 A	1
jazz01.tivoli.edu	QueueStatus	EventProcessor : QueueSize: 0 (Min: 0 Max: 0) DeltaQueue: 0	/16/2015 08:02:17 A	1
jazz01.tivoli.edu	QueueStatus	TESTENRICH : QueueSize: 0 (Min: 0 Max: 0) DeltaQueue: 0	/16/2015 08:02:17 A	1

Count	Type	ExpireTime	Agent	Manager	Suppr./Escl.
143	Information	330	OMNibus SelfMonitoring	OMNibus Self Monitoring @NCO	Normal
143	Information	330	OMNibus SelfMonitoring	OMNibus Self Monitoring @NCO	Normal
1	Problem	86400		ConnectionWatch	Normal
1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICCLUS	Normal
1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICCLUS	Normal
1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICCLUS	Normal

Self-monitoring events

The OMNibus event shows the type of monitoring in the **AlertGroup** field. The **Summary** field includes status details that vary depending on the type of monitored resource. Several other fields indicate that the self-monitoring service generated the event.

Student exercises



Command-line tools and self-monitoring

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What is the purpose of command-line tools?
2. How do you start the command-line manager?
3. How do you allow JavaScript in the command-line manager?

Review answers

1. What is the purpose of command-line tools?

With command-line tools, Netcool/Impact users can perform a variety of Netcool/Impact functions from the command line.

2. How do you start the command-line manager?

telnet <hostname> <commandlineport>

3. How do you allow JavaScript in the command-line manager?

*Type **js** in the command-line manager.*

Summary

You now can perform the following tasks:

- Describe Netcool/Impact command-line tools
- Describe the Netcool/Impact command-line manager
- Encrypt a policy from the command line
- Enable self-monitoring

Unit 16 The Netcool/Impact UI data provider

IBM Training



The Netcool/Impact UI data provider

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit describes how to configure Netcool/Impact to use Jazz for Service Management and the UI Data Provider.

Objectives

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

- Install Jazz for Service Management and the IBM Dashboard Application Services Hub
- Configure Netcool/Impact data sources and policies as UI data providers
- Configure communication between Netcool/Impact and the IBM Dashboard Application Services Hub
- Build a Dashboard Widget using data from a Netcool/Impact Policy

Lesson 1 The UI data provider

IBM Training

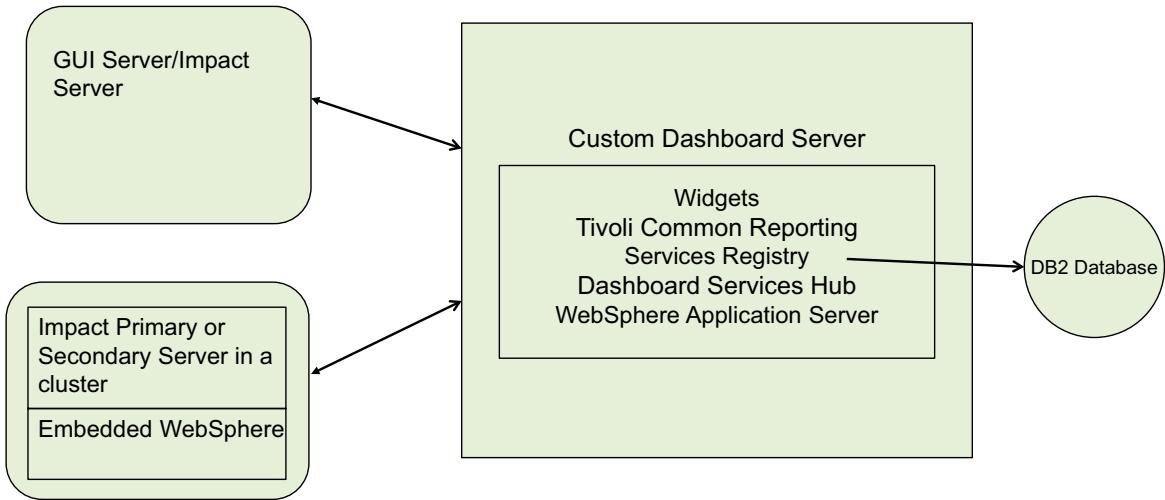


Lesson 1 The UI data provider

- The Netcool/Impact UI data provider or console is used to visualize data in the IBM Dashboard Application Services Hub
- Data types or Netcool/Impact policies are used to provide this data to create mashups of data from multiple sources
- To visualize data in the console, you must complete the following tasks:
 - Create the remote connection between the UI data provider and the console
 - Create the data model or policy that provides the data from Netcool/Impact
 - Create a page in the console
 - Create a widget on the page in the console
- Jazz for Service Management components are required

Use Netcool/Impact as a UI data provider to integrate data with other UI data providers in the IBM Dashboard Application Services Hub. Use the UI data provider DSA (client) to return data from any UI data provider. The DSA makes data from other UI providers available to process inside policies and to create combinations of data sets for display on a single dashboard widget.

The UI Data Model components



The graphic outlines the components that are required for using the UI data provider. Shown are a GUI server or Impact server, a secondary server or a primary server in a clustered environment, and a custom dashboard server.

Jazz for Service Management

- Jazz for Service Management is a separate product that is included with Netcool/Impact
- Jazz for Service Management contains many components
- Jazz for Service Management is an optional component that you can install if you want to use the following Netcool/Impact functions:
 - The UI data provider
 - Open Services for Lifecycle Collaboration or OSLC, an open community for software integration
OSLC requires Netcool/Impact 6.1.1 or higher
 - The self-service dashboard functions

Jazz for Service Management

Netcool/Impact can act as an Open Services for Lifecycle Collaboration (OSLC) provider and an OSLC client. Netcool/Impact can be used as a generic OSLC adapter for other OSLC and non-OSLC service providers.

Netcool/Impact uses the RDF/XML format for all OSLC responses, as required by the OSLC Core Specification v2. For more information about OSLC, see the OSLC Core Specification (<http://open-services.net/bin/view/Main/OslcCoreSpecification>).

Netcool/Impact does not support delegated UI dialogs or creation factories.

Installing Jazz for Service Management

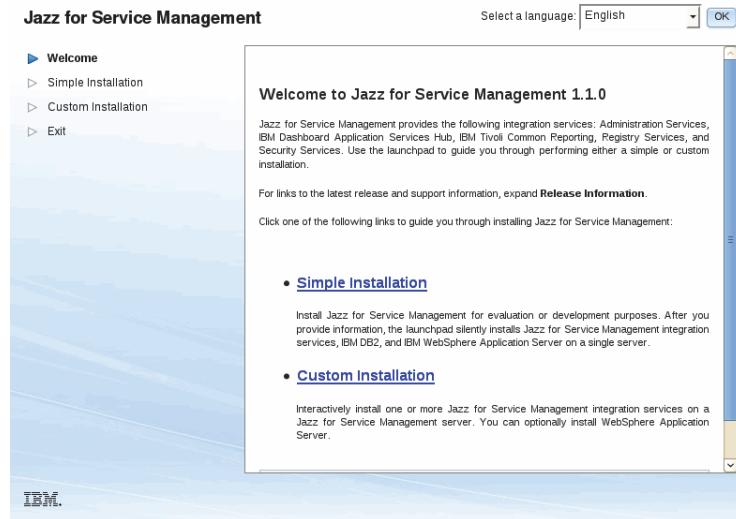
- Use the Jazz for Service Management installer to install Jazz for Service Management
- To use the UI data provider, install the IBM Dashboard Application Services Hub component first
- After you install the IBM Dashboard Application Services Hub component, install the self-service dashboards (SSD) component for Netcool/Impact on the IBM Dashboard Application Services Hub component
- To use Open Services for Life Cycle Collaboration (OSLC), you must install the Registry Services component also

Installing Jazz for Service Management

Download, extract, and install Jazz for Service Management as a separate component after **Netcool/Impact has been installed. This class uses the Linux version of the product, JSM_x.x_LAUNCHPAD_FOR_LINUX.zip.** A separate version of WebSphere is also required. This course uses **WAS_Vx.x_FOR_JAZZSM_LINUX_ML.zip.**

Always check for the latest product updates.

The Jazz for Service Management installer



The Netcool/Impact UI data provider

© Copyright IBM Corporation 2017

The Jazz for Service Management installer

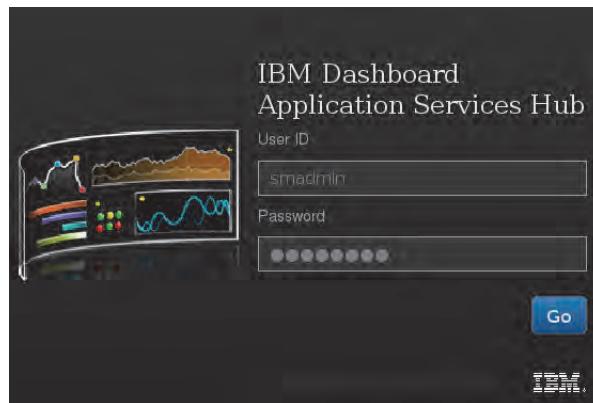
Install the product as the **root** user. Select **Simple installation** or **Custom Installation** depending upon which customization options are required.

The **JazzSM WebSphere profile** is created during the installation. A Jazz for Service Management administrator user ID and password are also created.

Install as the **root** user. The JazzSM WebSphere profile is created during the installation. A Jazz administrator is also created.

Starting the Jazz for Service Management console

```
netcool@jazz01:~/IBM/JazzSM/bin> pwd  
/home/netcool/IBM/JazzSM/bin  
netcool@jazz01:~/IBM/JazzSM/bin> ./tipLauncher.sh
```



Starting the Jazz for Service Management console

Run the Jazz for Service Management script to start the launcher; **tipLauncher.sh**. Enter the user ID and password created during the Jazz for Service Management installation.

In this example, **smadmin** user with password **object00** is used.

General steps to create a UI provider

- Before you can use the UI data provider, you must install the IBM Dashboard Application Services Hub component of Jazz for Service Management
- Create a policy, data type, or both to provide the data for a dashboard widget
- The policy must group all the items from a database table into a single Netcool/Impact object
 - Create communication between Netcool/Impact and the IBM Dashboard Application Services Hub
 - Create the dashboard widgets

General steps to create a UI provider

The general steps for creating the UI provider can differ, depending on where the data is located. For example: a data type or policy can provide data. In general, to integrate the UI data provider and the console, the steps described on the slide are required.

Example: UI data provider data type

The screenshot shows the 'SQL Data Type Config Page' for the 'HS_Location' data type. The 'Table Description' tab is selected. In the 'General Settings' section, the 'Data Type Name' is set to 'HS_Location' and the 'Data Source Name' is 'HS_TRAIN'. The 'Enabled' checkbox is checked, and the 'Access the data through UI data provider' checkbox is checked and circled in red. In the 'Table Description' section, the schema is set to 'IMPACT' and the table is 'LOCATION'. There is a 'Refresh Fields' button and a 'Show New / Deleted Fields' checkbox.

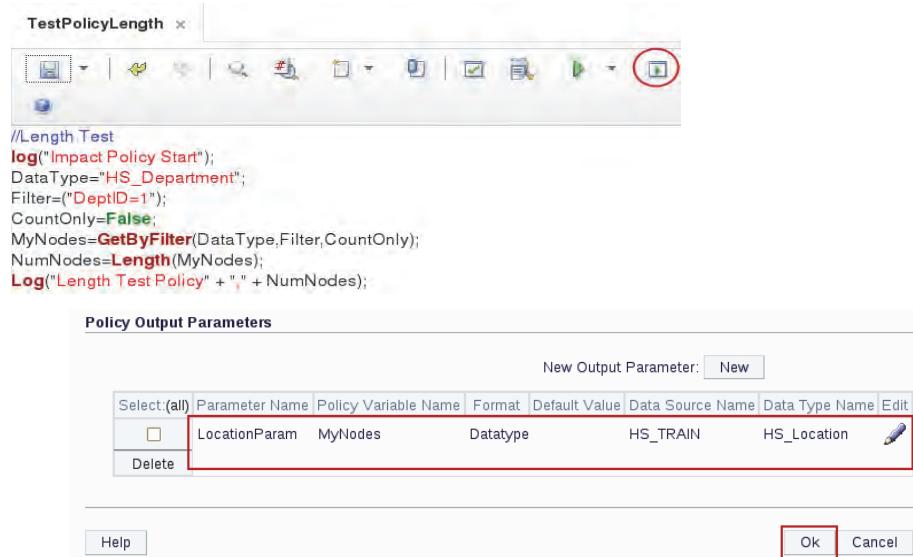
The Netcool/Impact UI data provider

© Copyright IBM Corporation 2017

Example: UI data provider data type

To set the parameter so that information from a data type can be used as a UI provider, select the **Access the data through UI data provider** check box. In this example, the HS_Location table definition has been modified.

Example: UI data provider policy user parameters



Example: UI data provider policy user parameters

Select or create a policy to supply user output parameters. The **Configure User Parameters** icon in the policy editor creates the output parameter definition.

Example: Configuring communication in WebSphere

The Netcool/Impact UI data provider

© Copyright IBM Corporation 2017

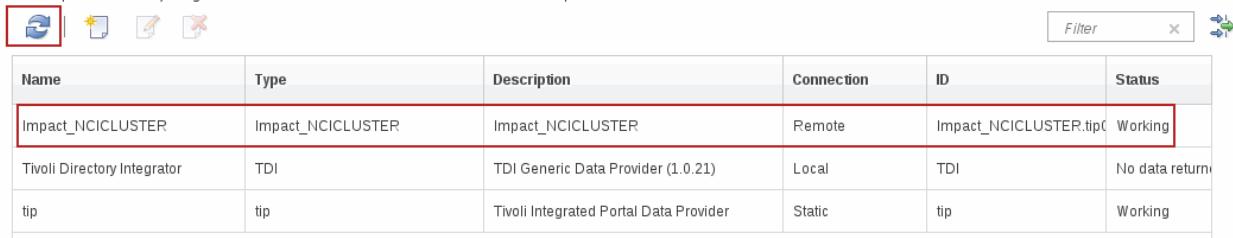
Example: Configuring communication in WebSphere

You configure communication from the WebSphere Integrated Solutions Console, which you can also access from the Jazz Console. The following steps are required:

1. Launch WebSphere administrative console.
2. Go to the NodeDefaultTrustStore page, click **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore**.
3. Select **Signer certificates** from the **Additional properties** menu.
4. Click the **Retrieve from port** button to retrieve a certificate from a remote server.
5. Provide the following required information:
 - Host or IP address
 - Port
 - Alias
6. Click **Retrieve signer information**.
7. Click **OK** to confirm and to return to the list of signer certificates.
8. Click **Save**.

Example: Connection in the Jazz console

Connections

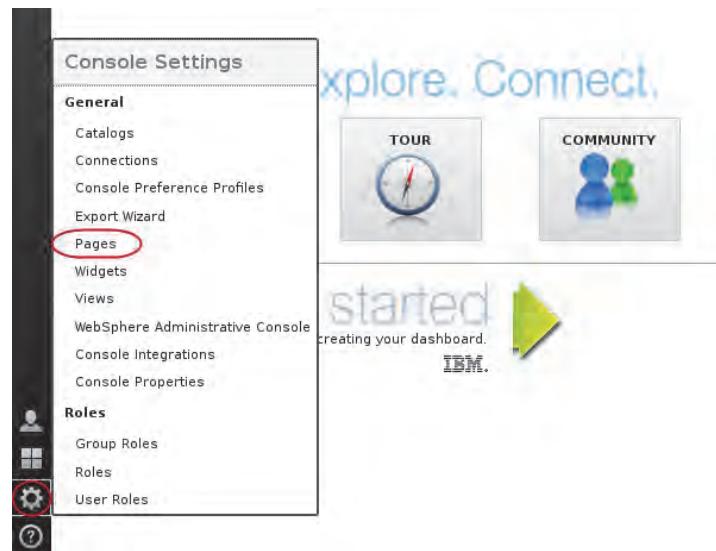


Name	Type	Description	Connection	ID	Status
Impact_NCICLUSTER	Impact_NCICLUSTER	Impact_NCICLUSTER	Remote	Impact_NCICLUSTER.tipC	Working
Tivoli Directory Integrator	TDI	TDI Generic Data Provider (1.0.21)	Local	TDI	No data returned
tip	tip	Tivoli Integrated Portal Data Provider	Static	tip	Working

Example: Connection in the Jazz console

When configured, the Connections table lists Netcool/Impact with a **Working** status.

Example: Console settings



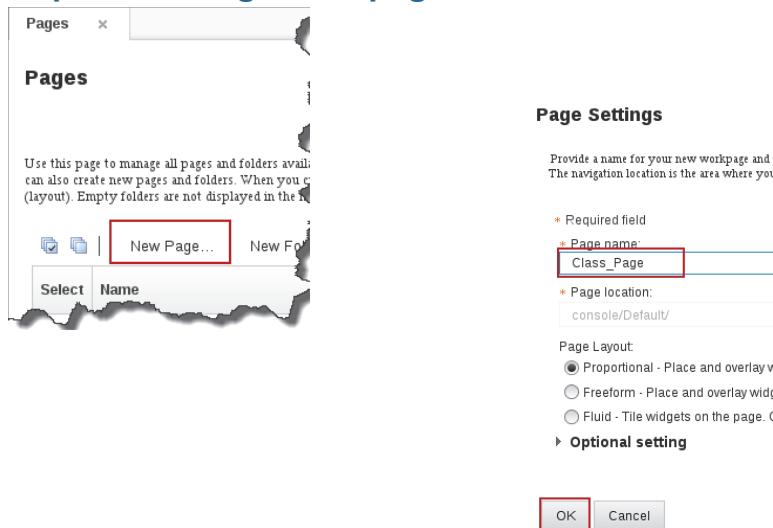
The Netcool/Impact UI data provider

© Copyright IBM Corporation 2017

Example: Console settings

In **Console Settings**, click **Pages** to create a new page.

Example: Creating a new page



The Netcool/Impact UI data provider

© Copyright IBM Corporation 2017

Example: Creating a new page

Click New Page. Supply a Page Name in this Class_Page.

Example: Adding and configuring the List widget

The screenshot shows the IBM Netcool/Impact UI interface. At the top, there's a toolbar with 'Pages', 'Search', 'Mobile', 'Layout', 'Widget' (which is selected), 'Save and Exit', 'Save', and 'Cancel'. Below the toolbar, a grid of dashboard widgets is displayed, with the 'List' widget highlighted. The main area is titled 'List' and contains a search bar and a list of items:

Location	Order
US:New York:Wall Street	1
UK:London:Bridas	2
US:San Francisco:Thompson	3
US:Dallas:Disraeli	4
US:Chicago:Gladstone	5
Australia:Sydney:Watershed	6

The Netcool/Impact UI data provider

© Copyright IBM Corporation 2017

Example: Adding and configuring the List widget

The **List** widget shows data supplied by the Netcool/Impact policy output field or from the HS_Location table.

Student exercises



The Netcool/Impact UI data provider

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. How do you configure a data type as a UI provider in Netcool/Impact?
2. What is OSLC?

Review answers

1. How do you configure a data type as a UI provider in Netcool/Impact?

*Select the **Access the data through UI data provider** check box on the Data Type definition screen.*

2. What is OSLC?

Open Services for Lifecycle Collaboration

Summary

You now can perform the following tasks:

- Install Jazz for Service Management and the IBM Dashboard Application Services Hub
- Configure Netcool/Impact data sources and policies as UI data providers
- Configure communication between Netcool/Impact and the IBM Dashboard Application Services Hub
- Build a Dashboard Widget using data from a Netcool/Impact Policy

Unit 17 Server utilities

IBM Training

IBM

Server utilities

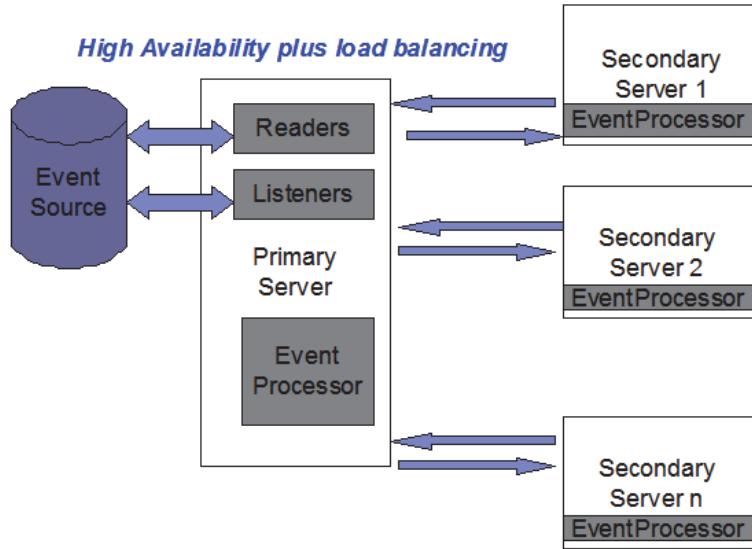
© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Objectives

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

- Describe the purpose of the export and import utilities
- Recover Netcool/Impact objects
- Create a second Netcool/Impact server instance
- Determine the primary server

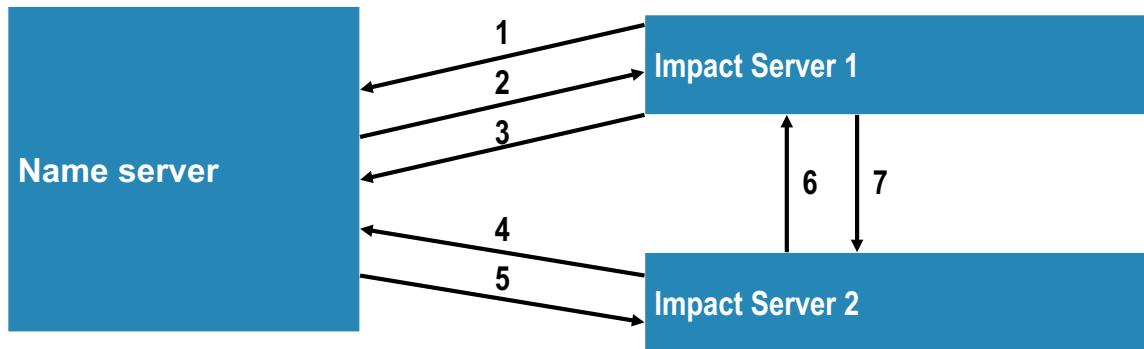
Clustering architecture



Multiple copies of Netcool/Impact can be configured in a clustered configuration. When configured in a cluster, one copy of Netcool/Impact assumes the role of master. The master retrieves events from the event source (ObjectServer) and distributes those events to other Netcool/Impact servers in the cluster for policy processing. All event updates are passed back to the master, which returns those updates to the respective event source.

When the master Netcool/Impact server fails, another server in the cluster assumes the role of the master and event processing continues. When the original master server recovers, it rejoins the cluster as a policy engine. The current master remains in that role. Configuration synchronization takes place automatically between all members of the cluster.

Name server initial registration

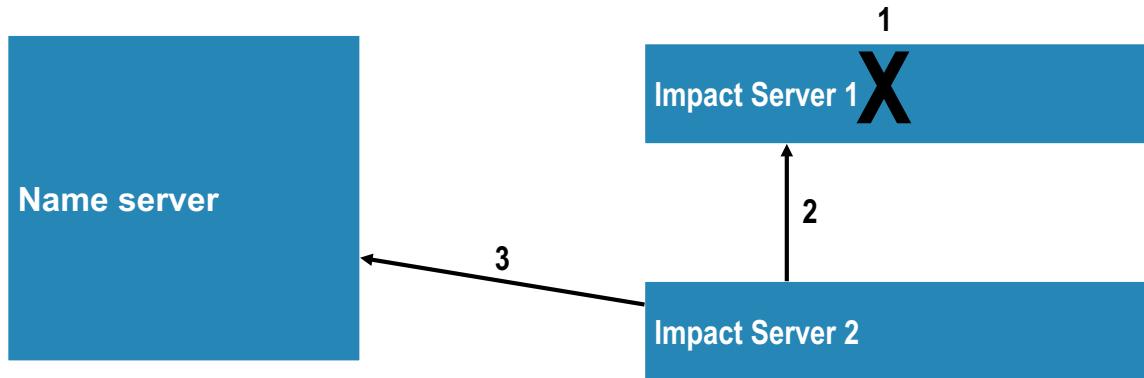


Name server initial registration

The name server is responsible for communicating with the Impact server and the GUI server. The name server initial registration process consists of the following steps:

1. Netcool/Impact Server 1 requests location of primary from the name server.
2. Name server responds with no primary available.
3. Netcool/Impact Server 1 requests to become the primary.
4. Netcool/Impact Server 2 requests location of primary from the name server.
5. Name server responds with Netcool/Impact Server 1 as primary.
6. Netcool/Impact Server 2 contacts Netcool/Impact Server 1 to register as a secondary Netcool/Impact server.
7. Netcool/Impact Server 1 acknowledges Netcool/Impact Server 2 and initiates replication activities.

Primary failure scenario



Primary failure scenario

This slide depicts a primary failure scenario:

1. Netcool/Impact Server 1 fails.
2. Netcool/Impact Server 2 sends request to Netcool/Impact Server 1. No response is received.
3. Netcool/Impact Server 2 requests to become the primary.

High-availability solution

- A high-availability solution eliminates a single point of failure that causes a Netcool/Impact processing outage
- The simplest solution focuses on making sure policy processing on events is always available
- GUI access is not required for normal operation

High-availability solution

Netcool/Impact provides a high-availability solution. A simple or a more complex solution may be required depending upon the production environment.

The name server can also be clustered. The secondary name servers are responsible for providing failover function for the primary name server.

nci_export command

- You can use the **nci_export** command to copy objects stored in a Netcool/Impact instance
- The command exports data source, data type, service, policy, and project information and store it in a local directory
- The command uses two arguments:
 - **server name:** The name of the server instance whose data you want to export
 - **Directory:** The directory where you want to store the exported data
- Example:
`./nci_export NCI /tmp/NCI_export`

nci_export command

The **/tmp** directory in the example must exist before the command will work properly. As a result of executing the **nci_export** command, a directory is created called **NCI_export**. The **NCI_export** directory contains all of the files and folders associated with the exported objects. Run this command for each server instance in a cluster before adding a new server to the cluster.

The **nci_import** command imports previously exported data back into a Netcool/Impact instance if there is a discrepancy. The **nci_import** syntax requires the name of the export directory:

nci_import server_name directory

nci_import command

- You can use the **nci_import** command to retrieve objects previously exported from a Netcool/Impact instance
- The command imports data source, data type, service, policy, and project information from the export directory
- The command uses two arguments:
 - **server name**: The name of the server instance
 - **Directory**: The directory where the exported data is stored
- **Example:**
`./nci_import NCI /tmp/NCI_export`

The **/tmp/NCI_export** directory in the example must exist. The **nci_export** command must have completed successfully before this command works properly.

Creating a second server on a separate host

- To create a second Netcool/Impact server on a separate host, use the Installation Manager
- Make the Netcool/Impact instance name within a shared cluster unique

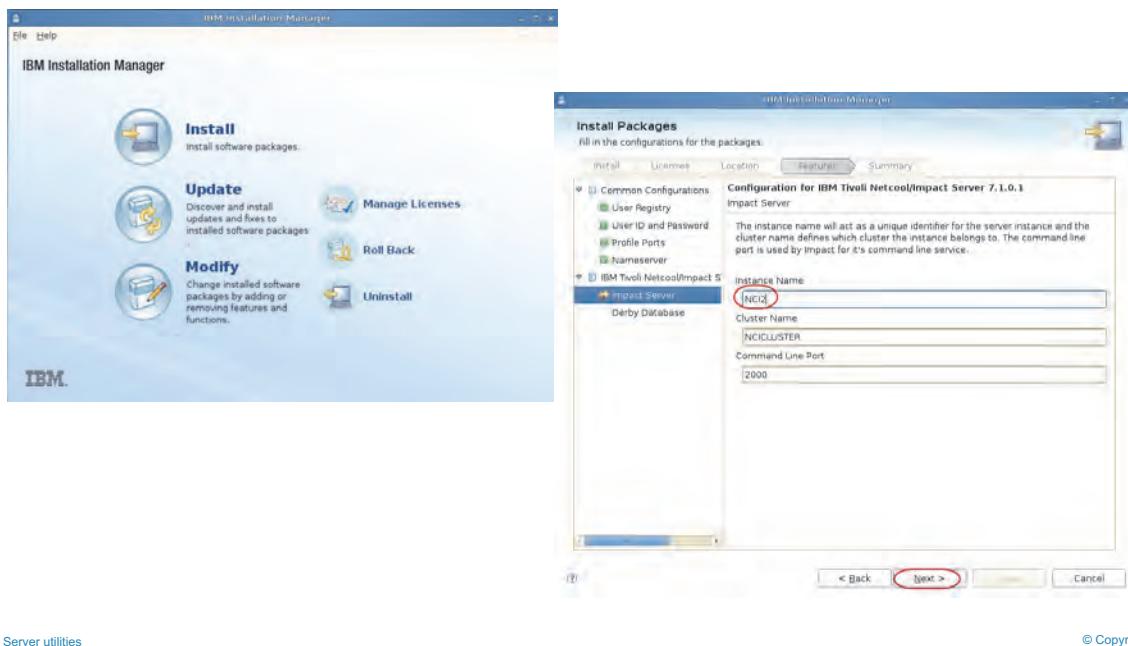
Creating a second server on a separate host

The installation binary is used to install a second Netcool/Impact server on a second host. To install a server cluster, a separate instance of the Netcool/Impact server is installed on each target system.

The Netcool/Impact installer prompts for the cluster name, which is used to identify the members of the server cluster at startup in the Netcool/Impact name server. All servers in a cluster must have the same cluster name. This name can be any unique identifying string.

Properties related to the Netcool/Impact name server are in the name server properties file. This file is named **nameserver.props** and is in the **\$NCHOME/impact/etc** directory.

Installing only the Impact server



Server utilities

© Copyright IBM Corporation 2017

Installing only the Impact server

During the installation process, you have the option to install only the Impact server and GUI server or a single component. When multiple Netcool/Impact instances are deployed, create unique and descriptive names applicable to the implementation environment.

Creating a new server on a single host

- Use ./IBMMIM command to create a new server in the **\$NCHOME/impact_1/install/directory**
- The nci_configuration_utility can no longer be used to create a secondary server as in Netcool/Impact version 6.1.1 and in previous releases

Creating a new server on a single host

The installer prompts you with a series of screens. Back up the primary server and verify the logs before creating a new server instances to be used for failover.

Starting the primary and secondary servers

- Use the **startImpactServer** command to start individual server instances
- The command is located in the **\$NCHOME/<impact instance install>/bin** directory
- Supply the server name as shown in the example:
`./startImpactServer.sh NCI2`

Starting the primary and secondary servers

Start and stop individual servers instances from their home install directories.

Determining the primary Netcool/Impact server

- Use the **nci_get_primary** command to find the name and the location of the primary server in a cluster
- The command is located in the **\$NCHOME/impact/bin** directory
- The command requires the following arguments:
 - nameServerHost: The host name or IP Address of the name server
 - nameServerPort: The port that the name server is listening on
 - clusterName: The name of the Impact cluster
- Example: `nci_get_primary jazz01.tivoli.edu 9080 NCICLUSTER`

```
< Result returned
    binding: 192.168.100.165:32941:NCI2
DONE
```

Determining the primary Netcool/Impact server

The example shows the results of using the **nci_get_primary** command. The name of the primary Netcool/Impact server instance is NCI2 in the cluster named NCICLUSTER. The IP address and port for the NCI2 instance is also shown. The command binds to the name server to acquire this information. The primary server can also be set in a properties file.

Student exercises



Server utilities

© Copyright IBM Corporation 2017

Student exercises

Open your *Student Exercises* book and perform the exercises for this unit.

Review questions

1. What is clustering?
2. How do you determine the primary server?

Review answers

1. What is purpose of clustering?

Clustering prevents the registry from acting as a single point of failure in a Netcool/Impact installation.

2. How do you determine the primary server?

*Use the **nci_get_primary** command.*

Summary

You now can perform the following tasks:

- Describe the purpose of the export and import utilities
- Recover Netcool/Impact objects
- Create a second Netcool/Impact server instance
- Determine the primary server



IBM Training



© Copyright IBM Corporation 2017 All Rights Reserved.