

Course Exercises

# 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](http://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

<b>Unit 1 Introduction to IBM Tivoli Netcool/Impact exercises .....</b>	<b>1-1</b>
Exercise 1 Installing Netcool/Impact V7.1.0.1 on SUSE Linux .....	1-1
Exercise 2 Verifying the installation .....	1-12
Exercise 3 Installing Fix Pack 5 .....	1-16
Exercise 4 Verifying Fix Pack installation .....	1-25
Exercise 5 Initializing the Derby database .....	1-27
Exercise 6 Modifying the OMNIBus Object server .....	1-29
<b>Unit 2 The Netcool/Impact user interface exercises .....</b>	<b>2-1</b>
Exercise 1 Starting the Netcool/Impact components .....	2-1
Exercise 2 Creating a project .....	2-2
Exercise 3 Removing project members .....	2-7
<b>Unit 3 The Netcool/Impact data model exercises .....</b>	<b>3-1</b>
Exercise 1 Creating the HS_TRAIN data source .....	3-1
Exercise 2 Creating data types for the HS_TRAIN data source .....	3-7
Exercise 3 Creating dynamic links .....	3-13
<b>Unit 4 Policies exercises.....</b>	<b>4-1</b>
Exercise 1 Creating a policy .....	4-1
Exercise 2 Creating a context policy .....	4-3
Exercise 3 Using the length function .....	4-4
Exercise 4 Using the Eval function .....	4-10
Exercise 5 Using the extract function .....	4-11
<b>Unit 5 Services exercises .....</b>	<b>5-1</b>
<b>Unit 6 The Enrichment policy exercises .....</b>	<b>6-1</b>
Exercise 1 Importing a policy with the policy uploader .....	6-1
Exercise 2 Configuring event mapping in the OMNIBus Event Reader service .....	6-6
Exercise 3 Verifying the results of the POLICY-Event_Enrichment-Basic policy .....	6-12
Exercise 4 Adding functions to the enhanced event enrichment policy .....	6-17
Exercise 5 Testing the enhanced enrichment policy .....	6-22
Exercise 6 Verifying the results of the POLICY-EventEnrichment-Adv policy .....	6-26
<b>Unit 7 Controlling policy execution sequence exercises .....</b>	<b>7-1</b>
Exercise 1 Observing chaining and thread locking .....	7-1
Exercise 2 Configuring thread locking .....	7-8
<b>Unit 8 Policy wizards exercises .....</b>	<b>8-1</b>
Exercise 1 Creating an enrichment policy using the wizard .....	8-1

<b>Unit 9 Notification policies exercises .....</b>	<b>9-1</b>
Exercise 1 Notification .....	9-1
Exercise 2 Configuring the EmailSender service .....	9-3
Exercise 3 Creating the policy .....	9-4
Exercise 4 Verifying SendEmailTest results .....	9-6
<b>Unit 10 Reports exercises .....</b>	<b>10-1</b>
Exercise 1 Enabling report collection .....	10-1
Exercise 2 Creating the Impact Profile report .....	10-3
Exercise 3 Creating an Action Efficiency report .....	10-4
Exercise 4 Viewing the configuration documenter contents .....	10-6
Exercise 5 Disable reporting .....	10-7
<b>Unit 11 Operator views exercises .....</b>	<b>11-1</b>
Exercise 1 Creating a basic operator view .....	11-1
Uploading the policy for the Action panel .....	11-1
Creating and configuring the data source .....	11-3
Creating the operator view .....	11-4
Exercise 2 Creating an advanced (custom) operator view .....	11-14
Creating the base operator views .....	11-14
Sample database .....	11-16
Creating the data source and data types required by the policies .....	11-16
Building the web pages .....	11-20
Policies .....	11-20
Modifying the policies .....	11-22
HTML pages .....	11-23
Viewing the advanced operator view .....	11-25
<b>Unit 12 Working with web services exercises .....</b>	<b>12-1</b>
Exercise 1 Creating a policy using the Web Services wizard .....	12-1
Building the Web Service policy .....	12-1
Testing the policy .....	12-9
Exercise 2 The web service response .....	12-10
Parsing the response .....	12-10
<b>Unit 13 Hibernation, X in Y, and synthetic events exercises .....</b>	<b>13-1</b>
Exercise 1 Hibernations .....	13-1
Exercise 2 Using a policy to create synthetic events .....	13-12
Exercise 3 Testing synthetic events .....	13-15
Exercise 4 X in Y policy .....	13-20
Exercise 5 Testing the policy .....	13-25
<b>Unit 14 Maintenance window management exercises .....</b>	<b>14-1</b>
Exercise 1 Maintenance window management .....	14-1
Exercise 2 Modifying Maintenance Window Management properties .....	14-8
<b>Unit 15 Command-line tools and self-monitoring exercises .....</b>	<b>15-1</b>
Exercise 1 Encrypting a policy .....	15-1
Exercise 2 Enabling self-monitoring .....	15-5

<b>Unit 16 The Netcool/Impact UI data provider exercises . . . . .</b>	<b>16-1</b>
Exercise 1 Configuring a data source . . . . .	16-1
Exercise 2 Building a widget . . . . .	16-3
Exercise 3 Opening the Operator View using the Web Widget . . . . .	16-13
<b>Unit 17 Server utilities exercises . . . . .</b>	<b>17-1</b>
Exercise 1 Exporting Netcool/Impact server . . . . .	17-1
Exercise 2 Creating a second server instance . . . . .	17-5
Exercise 3 Importing Netcool/Impact Server . . . . .	17-22



---

# **Unit 1 Introduction to IBM Tivoli Netcool/Impact exercises**

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

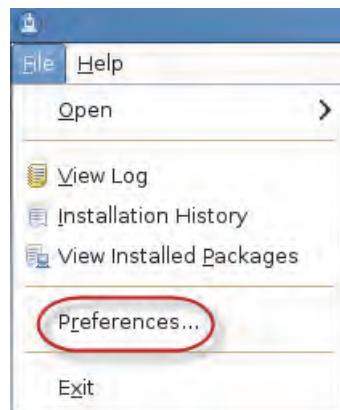
## **Exercise 1 Installing Netcool/Impact V7.1.0.1 on SUSE Linux**

In this exercise, you use IBM Installation Manager to install Netcool/Impact 7.1.0.1 and the Netcool/Impact extensions for Netcool Operations Insight (NOI) 1.4. IBM Installation Manager was installed with the non-root user ID **netcool** during the virtual image configuration. A Netcool/OMNIbus ObjectServer is installed during the virtual image configuration. The ObjectServer user repository is used during the Netcool/Impact installation.

1. Log in to the Impact71\_IMG1 virtual image as the user **netcool**, with the password **object00**.
2. Open a command prompt and start the Netcool OMNIbus ObjectServer. Enter the following command:  
`/opt/IBM/tivoli/netcool/omnibus/bin/nco_objserv -name NCOMS &`
3. Verify that the ObjectServer is operational. Enter the following command:  
`/opt/IBM/tivoli/netcool/omnibus/bin/nco_ping NCOMS`
4. Change to the Installation Manager directory:  
`cd /home/netcool/IBM/InstallationManager/eclipse`
5. Use the following command to start the Installation Manager tool:  
`./IBMIM`

```
netcool@jazz01:~/Desktop> cd /home/netcool/IBM/InstallationManager/eclipse  
netcool@jazz01:~/IBM/InstallationManager/eclipse> ./IBMIM
```

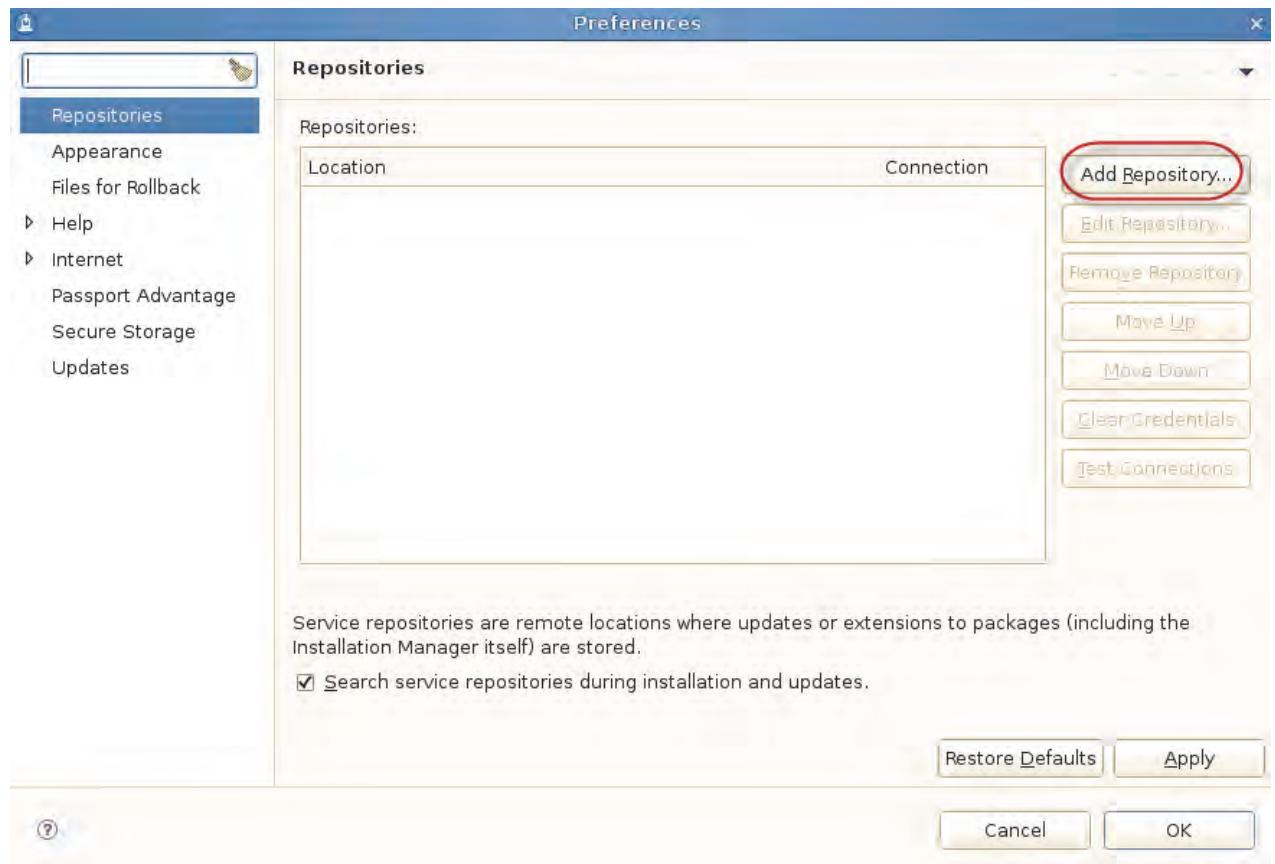
6. Configure the installation repositories. Select **File > Preferences**.



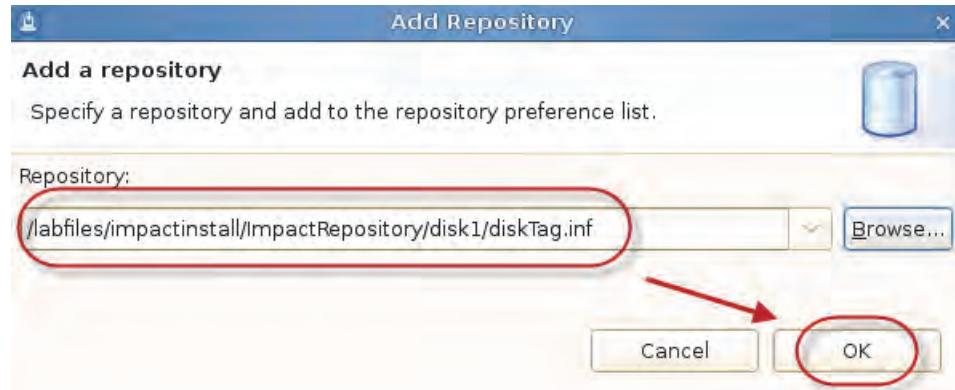
7. Add the repository for the Netcool/Impact 7.1.0.1 installation files. Click **Add Repository**.



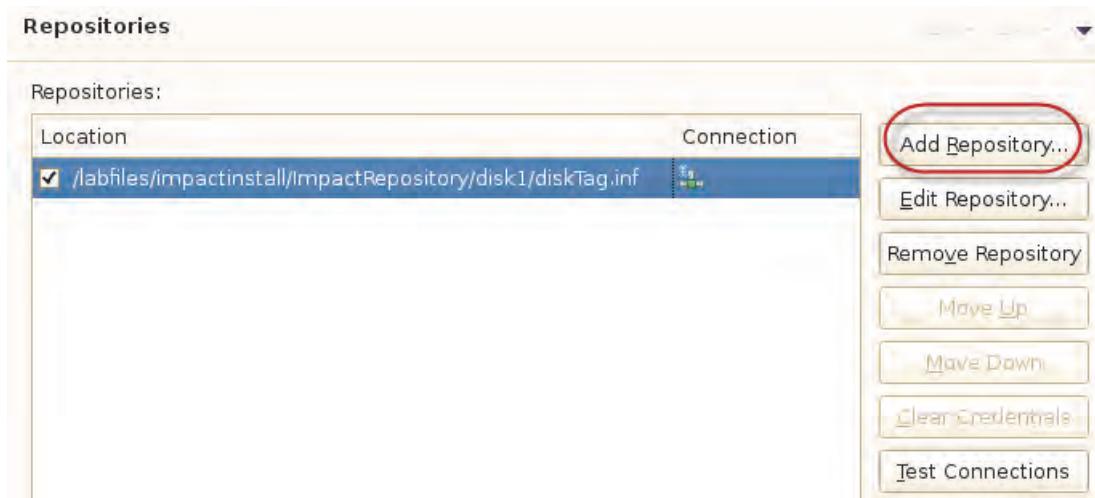
**Important:** Delete any existing repositories before adding the new ones.



8. Enter **/labfiles/impactinstall/ImpactRepository/disk1/diskTag.inf** in the **Repository** field and click **OK**.



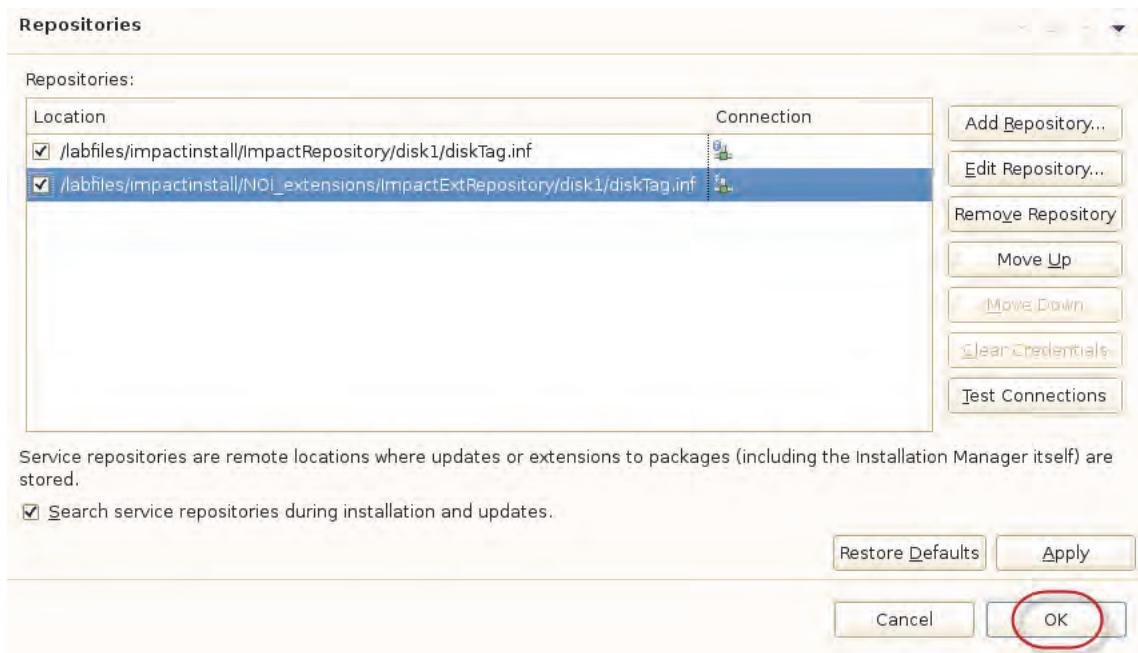
9. Add the repository for the Netcool/Impact 7.1.0.1 installation files. Click **Add Repository**.



10. Enter **/labfiles/impactinstall/NOI\_extensions/ImpactExtRepository/disk1/diskTag.inf** in the **Repository** field and click **OK**.

The two repositories are shown in the Repositories section.

11. Click **OK**.



12. Click the icon to the left of **Install**.



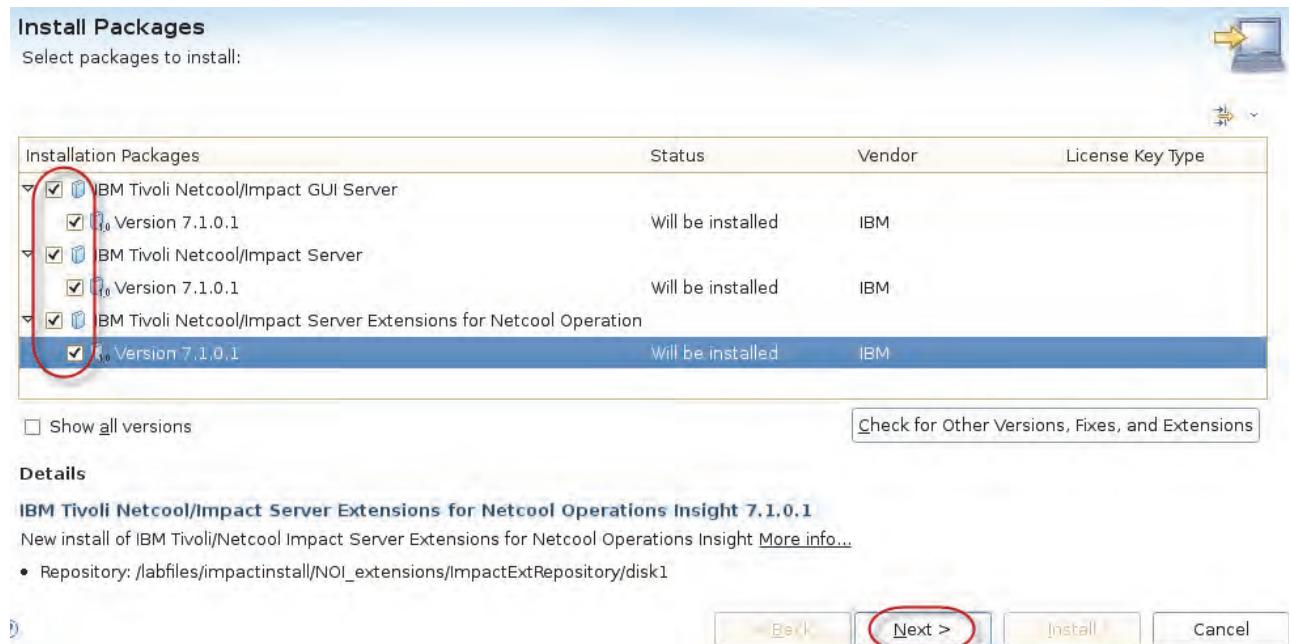
13. Select the following installation packages:

- IBM Tivoli Netcool/Impact GUI Server
- IBM Tivoli Netcool/Impact Server
- IBM Tivoli Netcool/Impact Server Extensions for Netcool Operation

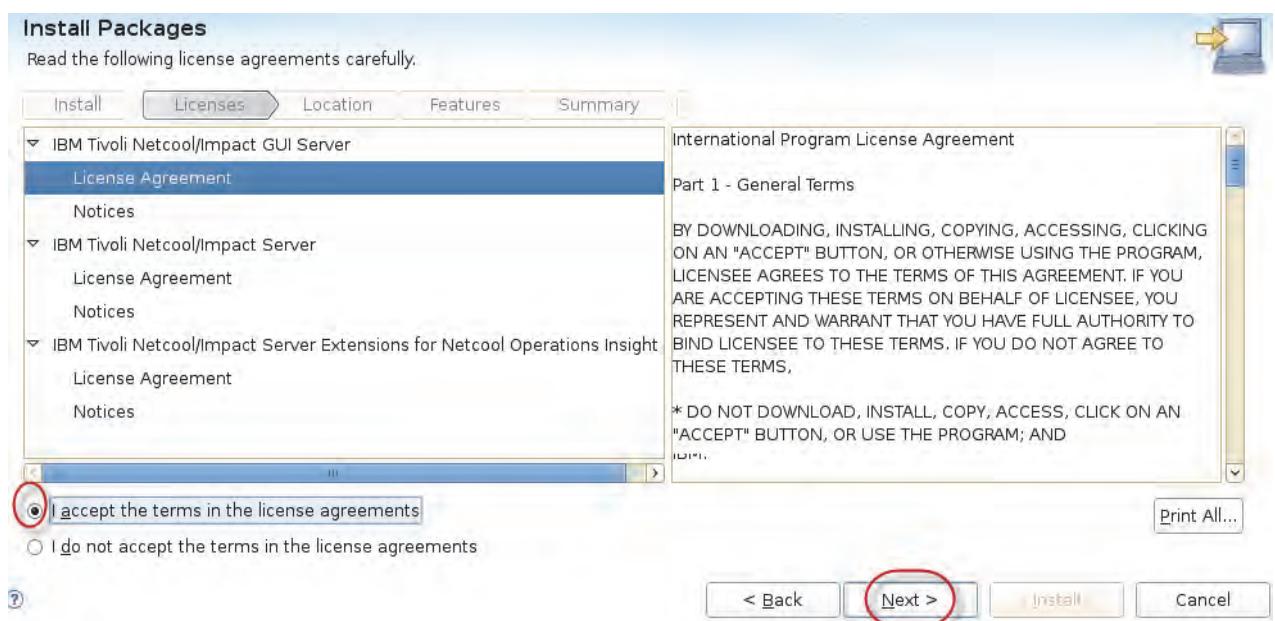


**Note:** The **Version 7.1.0.1** subentries are automatically selected as each installation package is selected.

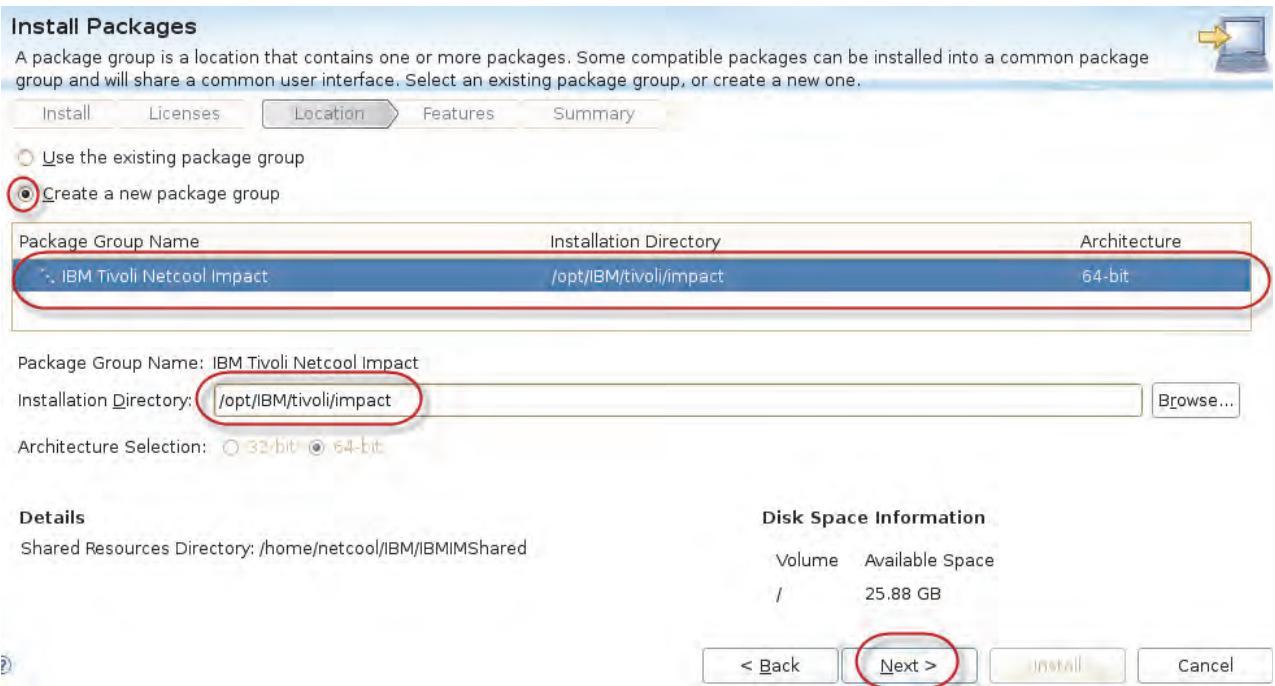
14. Click Next.



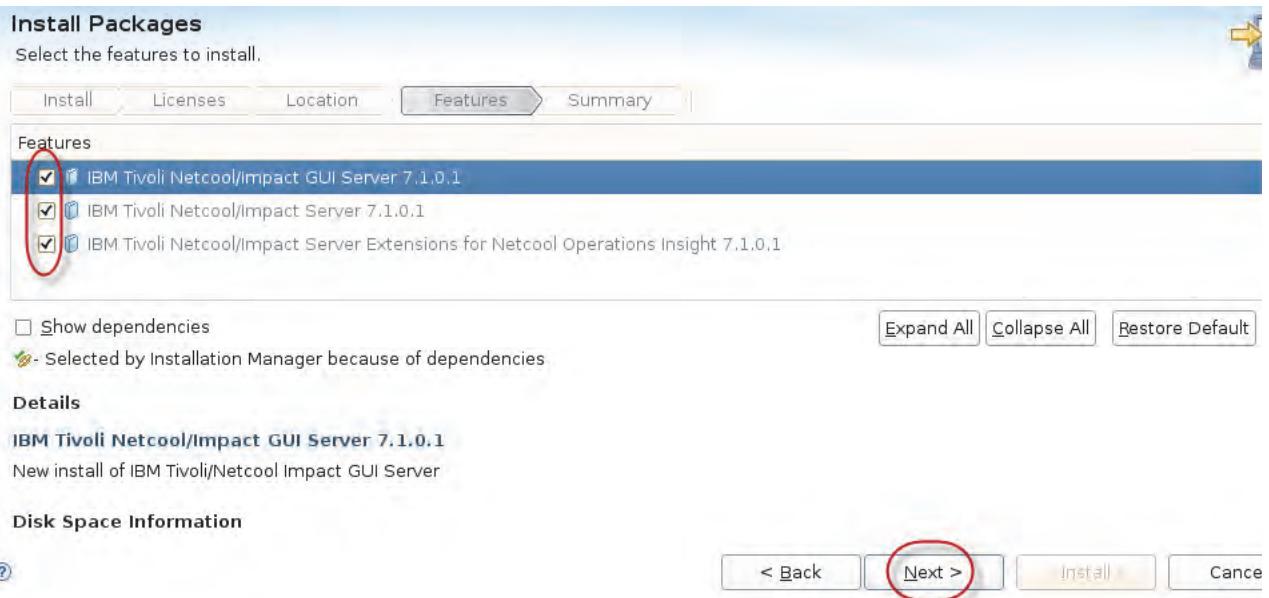
15. Select I accept the terms in the license agreements and click Next.



16. Use the default values for creating a new package group and the installation directory and click **Next**.



17. Verify that all features are selected and click **Next**.

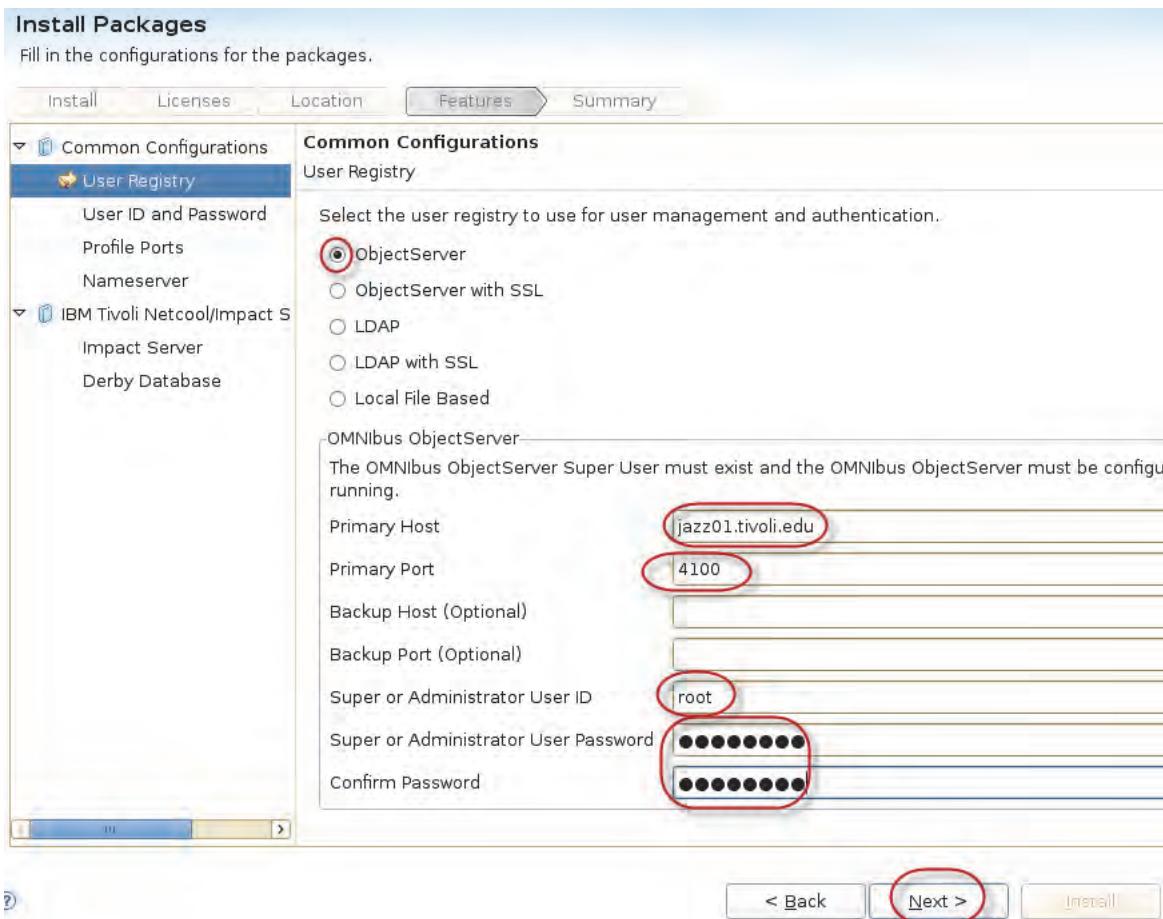


18. You must use the ObjectServer as the user registry. Enter **jazz01.tivoli.edu** in the **Primary Host** field.

19. Enter **4100** in the **Primary Port** field.

20. Enter **root** in the **Super or Administrator User ID** field.

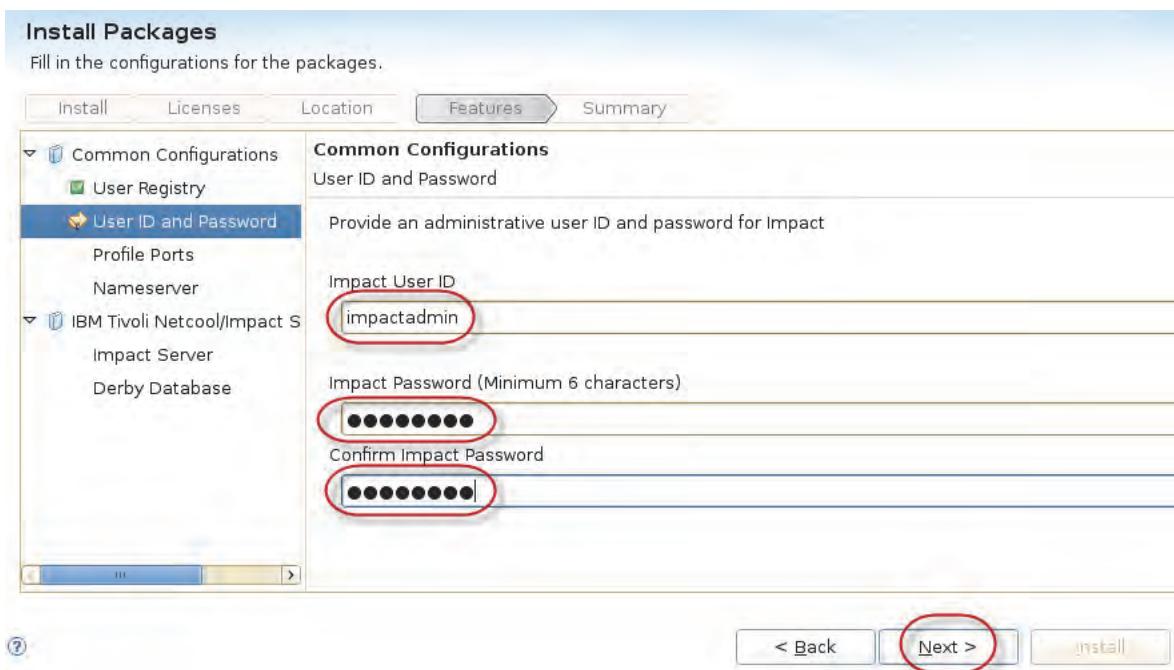
21. Enter **object00** in the password and password confirmation fields and click **Next**.



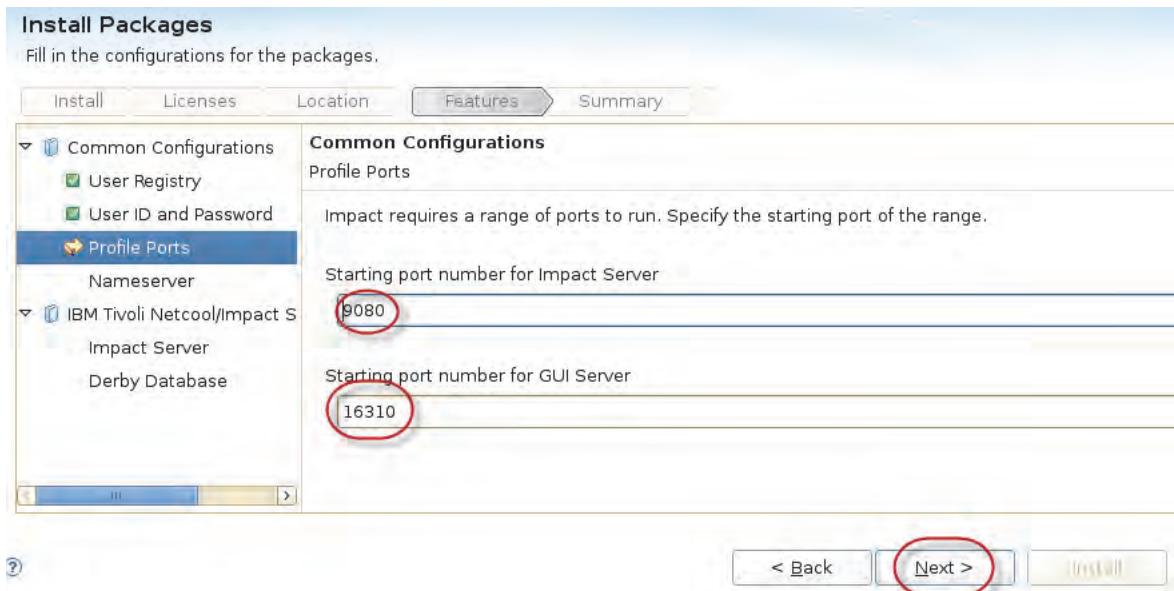
22. Enter **impactadmin** in the **Impact User ID** field.

The user ID **impactadmin** was created in the ObjectServer repository during the image configuration.

23. Enter **object00** in the password and password confirmation fields and click **Next**.

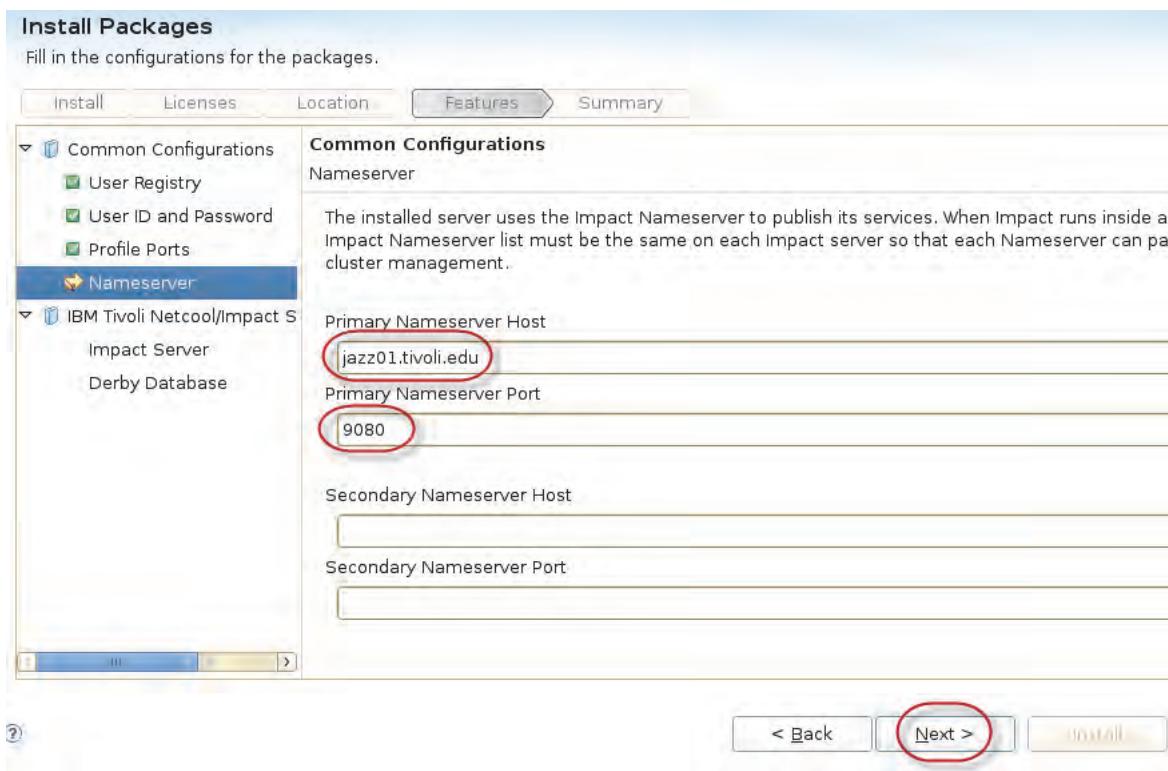


24. Use the default profile port values and click **Next**.

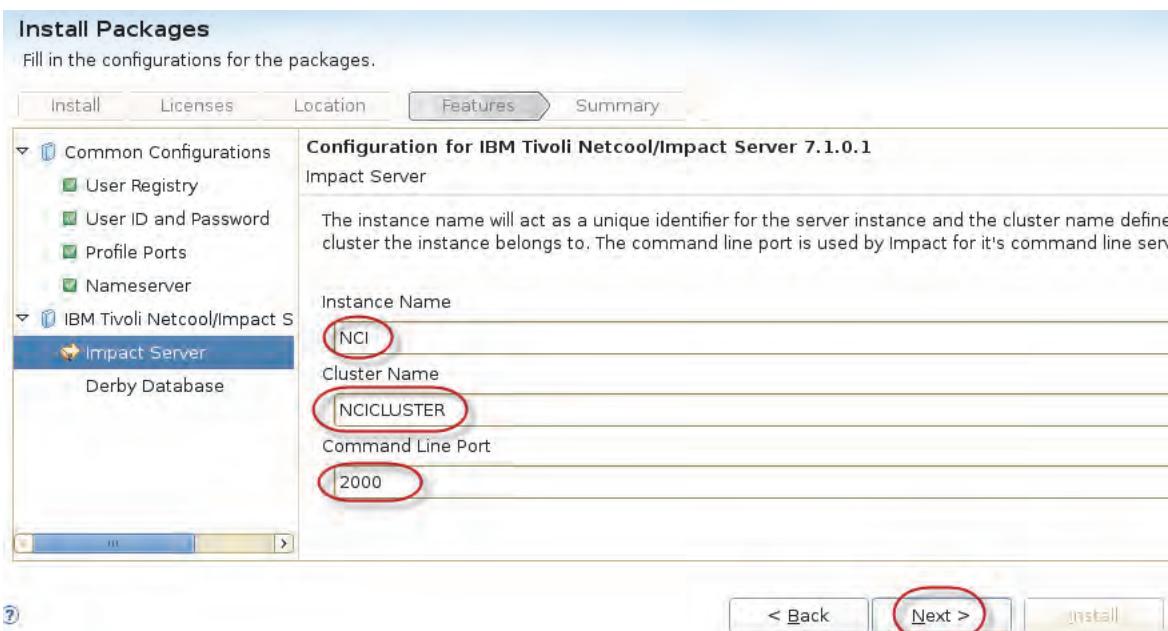


25. Enter **jazz01.tivoli.edu** in the **Primary Nameserver Host** field.

26. Enter **9080** in the **Primary Nameserver Port** field and click **Next**.



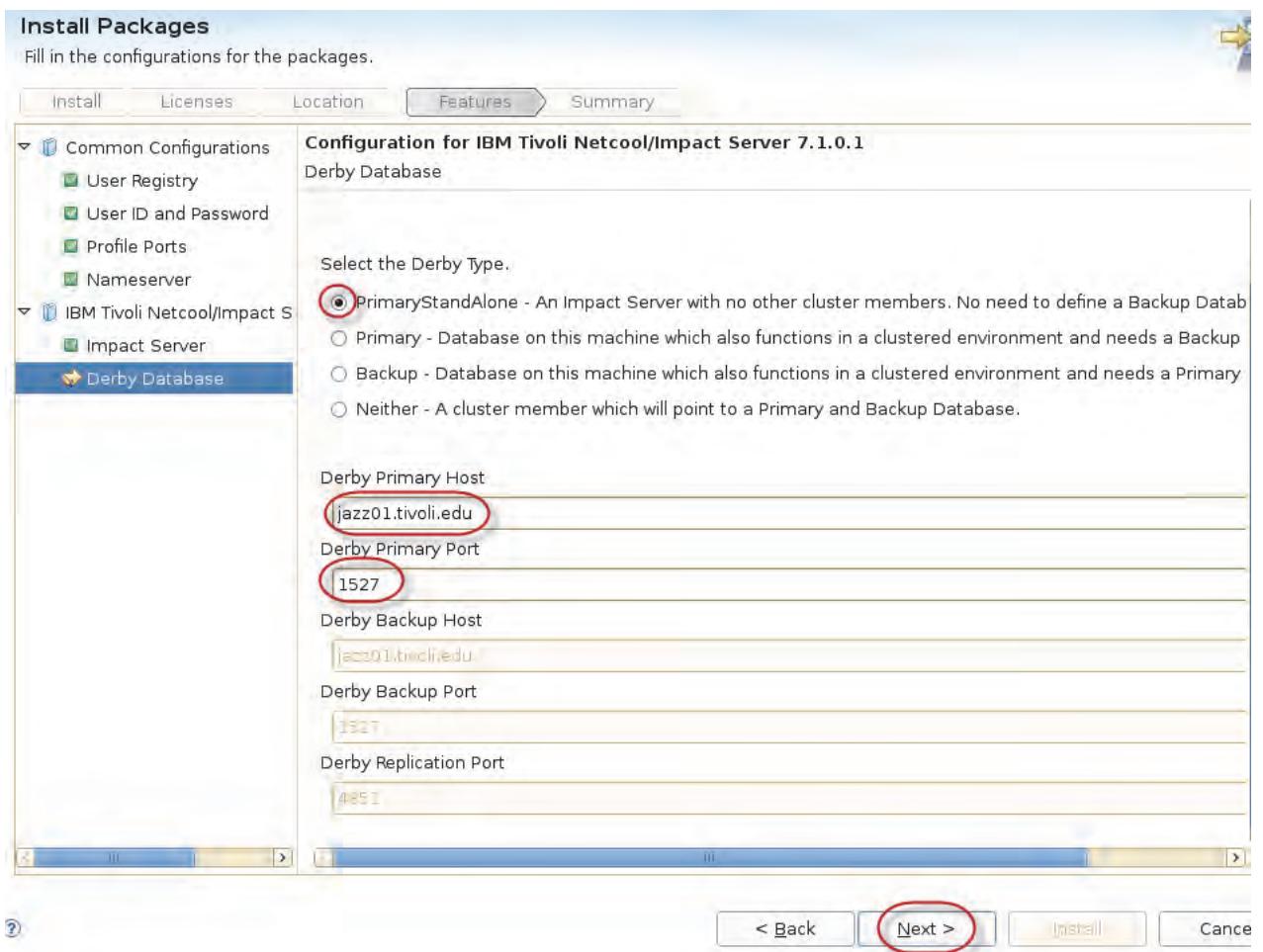
27. Use the default **Instance Name**, **Cluster Name**, and **Command Line Port** values and click **Next**.



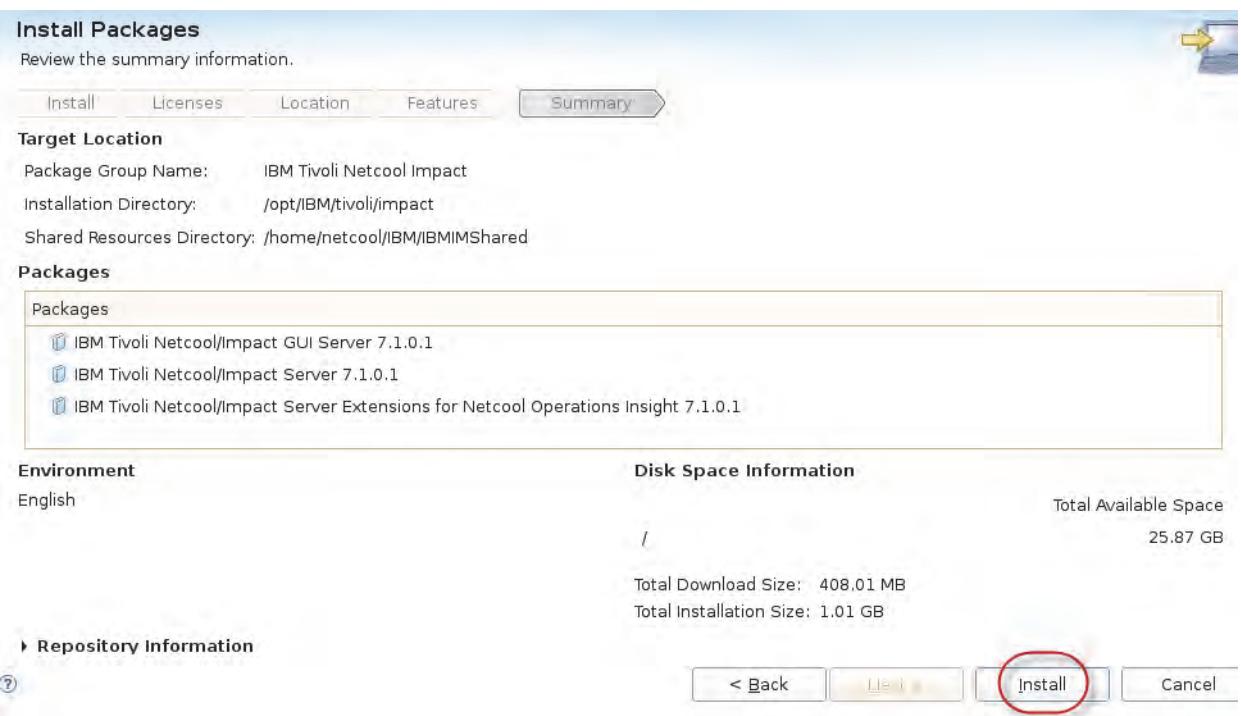
28. Select **PrimaryStandalone** in the **Derby Type** list.

29. Enter **jazz01.tivoli.edu** in the **Derby Primary Host** field.

30. Enter 1527 in the Derby Primary Port field and click **Next**.



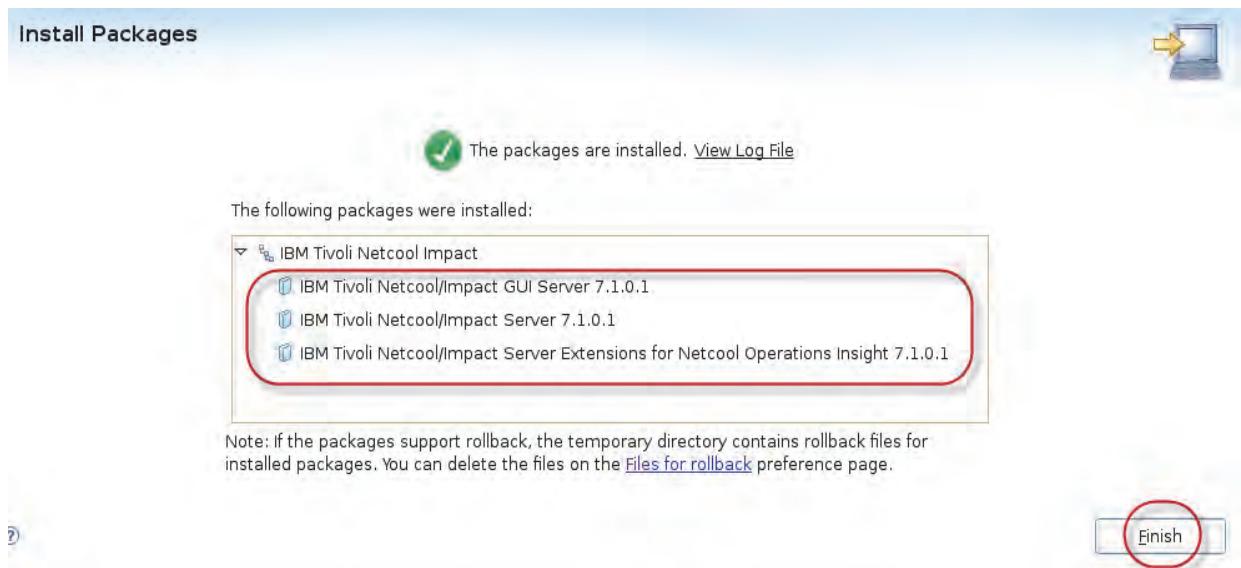
31. Review the installation package summary information and click **Install**.



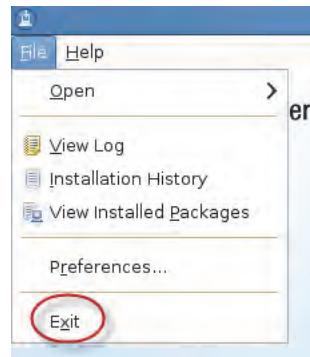
You see a progress bar that shows the component installation progress. Typical install time is approximately 40 minutes.

The Installation Manager tool shows a screen that indicates that all of the packages are successfully installed.

32. Click **Finish**.



33. Close the Installation Manager tool by selecting the **File > Exit** menu.



## Exercise 2 Verifying the installation

1. Start the web browser application on the jazz01 virtual image. Click **Computer** in the lower-left of the image desktop and select **Firefox** in the **Applications** tab.



2. Enter the default Netcool/Impact web console address:

<https://jazz01.tivoli.edu:16311/ibm/console/>

The web browser opens a certificate warning window.

3. Click I Understand the Risks.

The screenshot shows a Firefox browser window displaying a security warning. At the top left is a yellow icon of a person holding a shield. To its right, the text "This Connection is Untrusted" is displayed in bold. Below this, a message states: "You have asked Firefox to connect securely to **jazz01.tivoli.edu:16311**, but we can't confirm that your connection is secure." A note below says: "Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified." Underneath the message are two buttons: "Get me out of here!" and "I Understand the Risks". The "I Understand the Risks" button is highlighted with a red oval.

4. Click Add Exception.

This screenshot shows the same Firefox warning dialog as the previous one, but with a different focus. The "Add Exception..." button at the bottom of the dialog is circled with a red oval. The rest of the dialog content is identical to the first screenshot, including the yellow icon, the title, the message about untrusted connections, and the "Get me out of here!" and "I Understand the Risks" buttons.

5. Click **Confirm Security Exception** to import and store the application certificate.



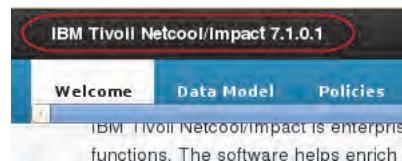
6. Log in to the Netcool/Impact console as the user **impactadmin**, with the password **object00**. Click **Go**.



You see the Getting Started page, indicating that the installation is successful.

The screenshot shows the 'Getting started' section of the IBM Tivoli Netcool/Impact 7.1.0.1 interface. At the top, the title bar displays 'IBM Tivoli Netcool/Impact 7.1.0.1' and the host information 'NCI:NCICLUSTER (192.168.100.165:46664:NCI)'. The navigation menu includes 'Welcome', 'Data Model', 'Policies', 'Services', 'Operator View', 'Event Isolation and Correlation', and 'Maintenance Window'. Below the menu, a large heading 'Getting started' is followed by the text 'Thank you for using IBM Tivoli Netcool/Impact'. A brief description explains that the software automates support for business-critical functions. A section titled 'Managing Netcool/Impact' lists key tasks: 'Data Model', 'Policies', 'Services', and 'Operator View', each accompanied by an icon. A 'Resources' section at the bottom contains a link to documentation.

The application version is shown in the upper left of the console.



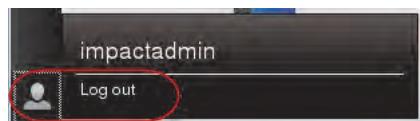
7. View links for the learning resources in the **Resources** section of the Welcome page.



**Note:** Some links require an Internet connection. The laboratory environment does not have Internet access.

The screenshot shows the 'Resources' section from the 'Getting started' page. It features a heading 'Resources' and a sub-instruction 'If you have questions, there is information available when you need it.' Below this, five links are listed with icons: 'User Documentation' (book icon), 'Wiki' (question mark icon), 'Video Gallery' (clapboard icon), 'Blogs' (RSS feed icon), and 'Forums' (speech bubble icon).

8. Log off the console. Click the person icon in the lower left of the console and select **Logout**.



9. Close the browser. Select the **File > Quit** menu.

## Exercise 3 Installing Fix Pack 5

1. Change to the Installation Manager directory:

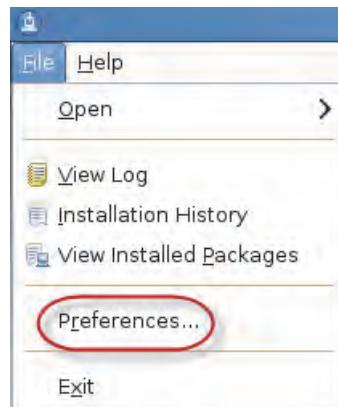
```
cd /home/netcool/IBM/InstallationManager/eclipse
```

2. Use the following command to start the Installation Manager tool:

```
./IBMMIM
```

```
netcool@jazz01:~/Desktop> cd /home/netcool/IBM/InstallationManager/eclipse
netcool@jazz01:~/IBM/InstallationManager/eclipse> ./IBMMIM
```

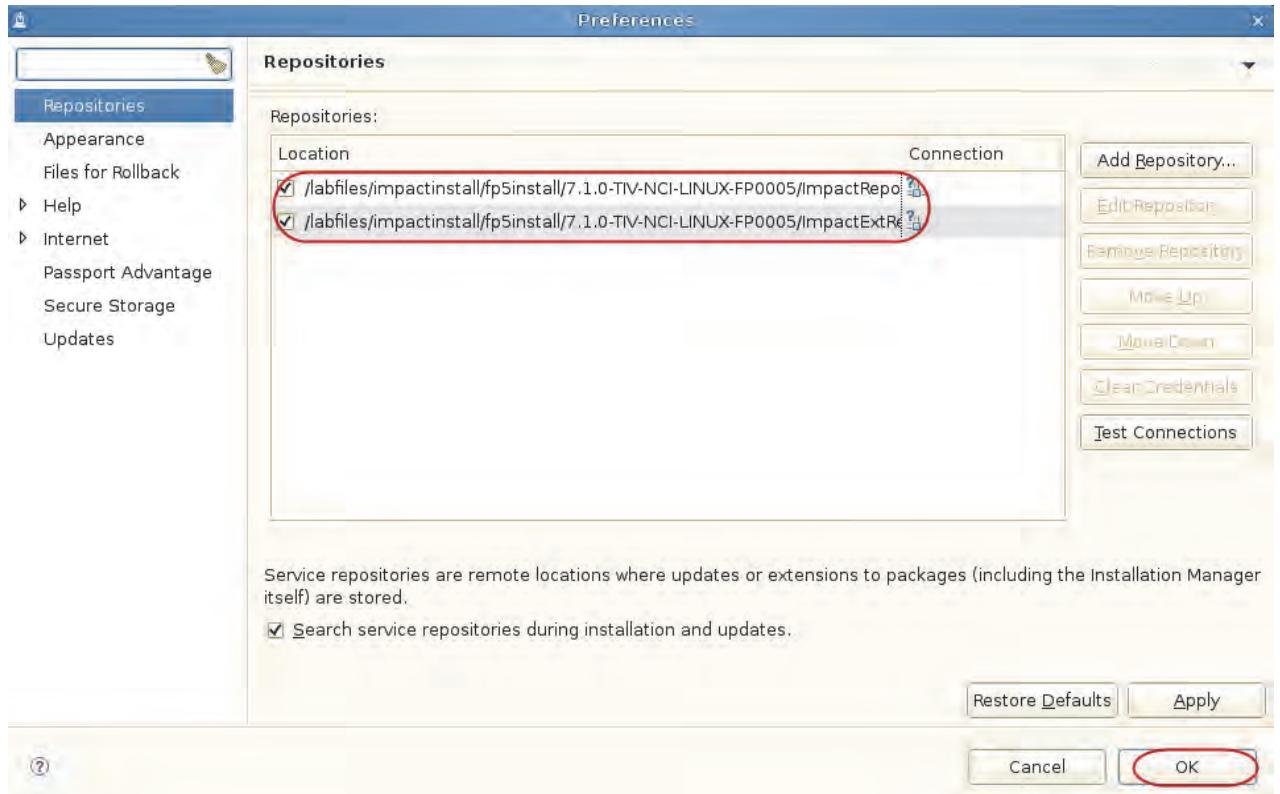
3. Configure the installation repositories. Select **File > Preferences**.



4. Add the repositories for the Netcool/Impact fp5 installation files. Click **Add Repository**.

```
/labfiles/impactinstall/fp5install/ImpactRepository/repository.config
/labfiles/impactinstall/fp5install/ImpactExtRepository/repository.config
```

5. The two repositories are shown in the Repositories section. Click **OK**.

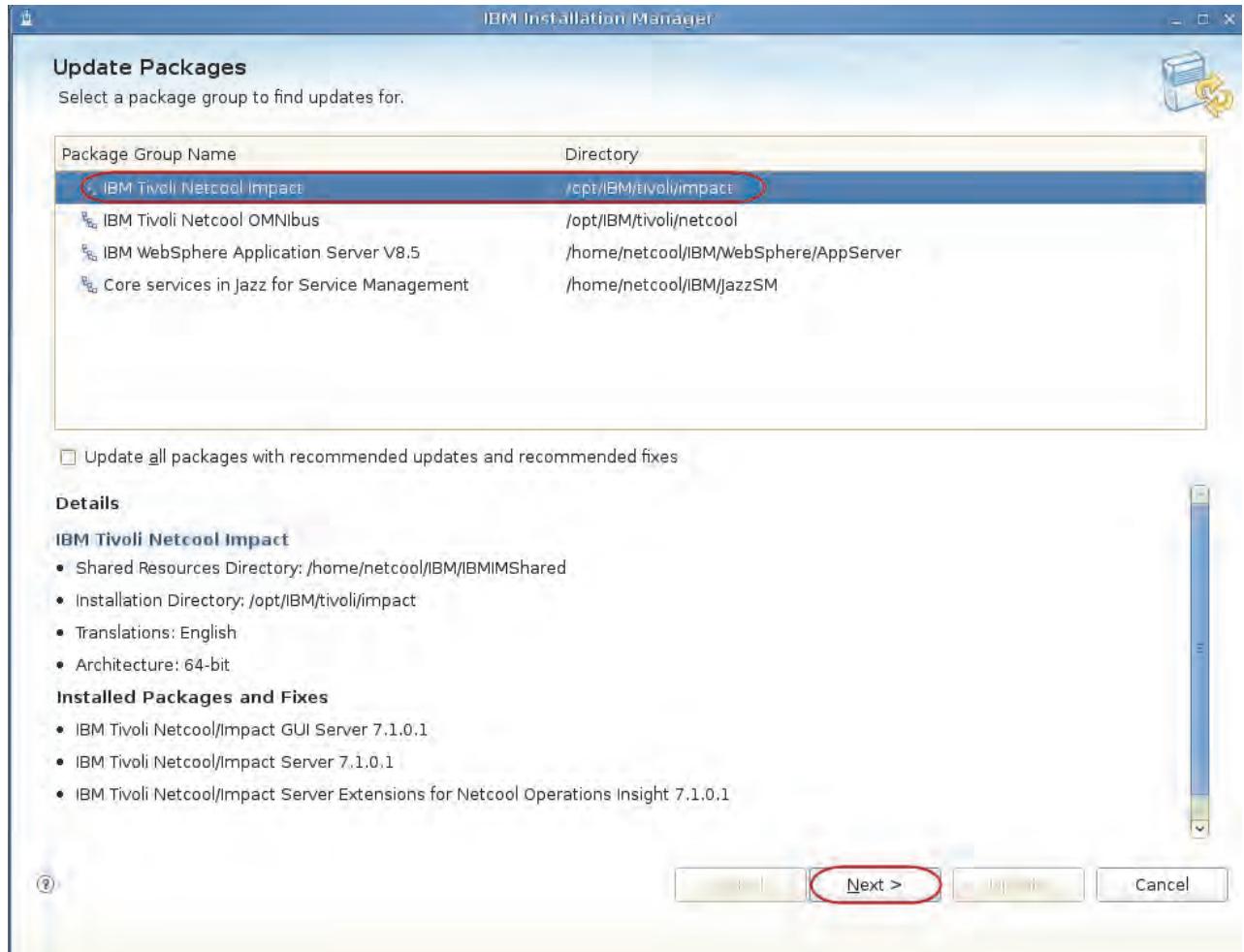


6. Click the icon to the left of **Update**.



7. Select the package to update:

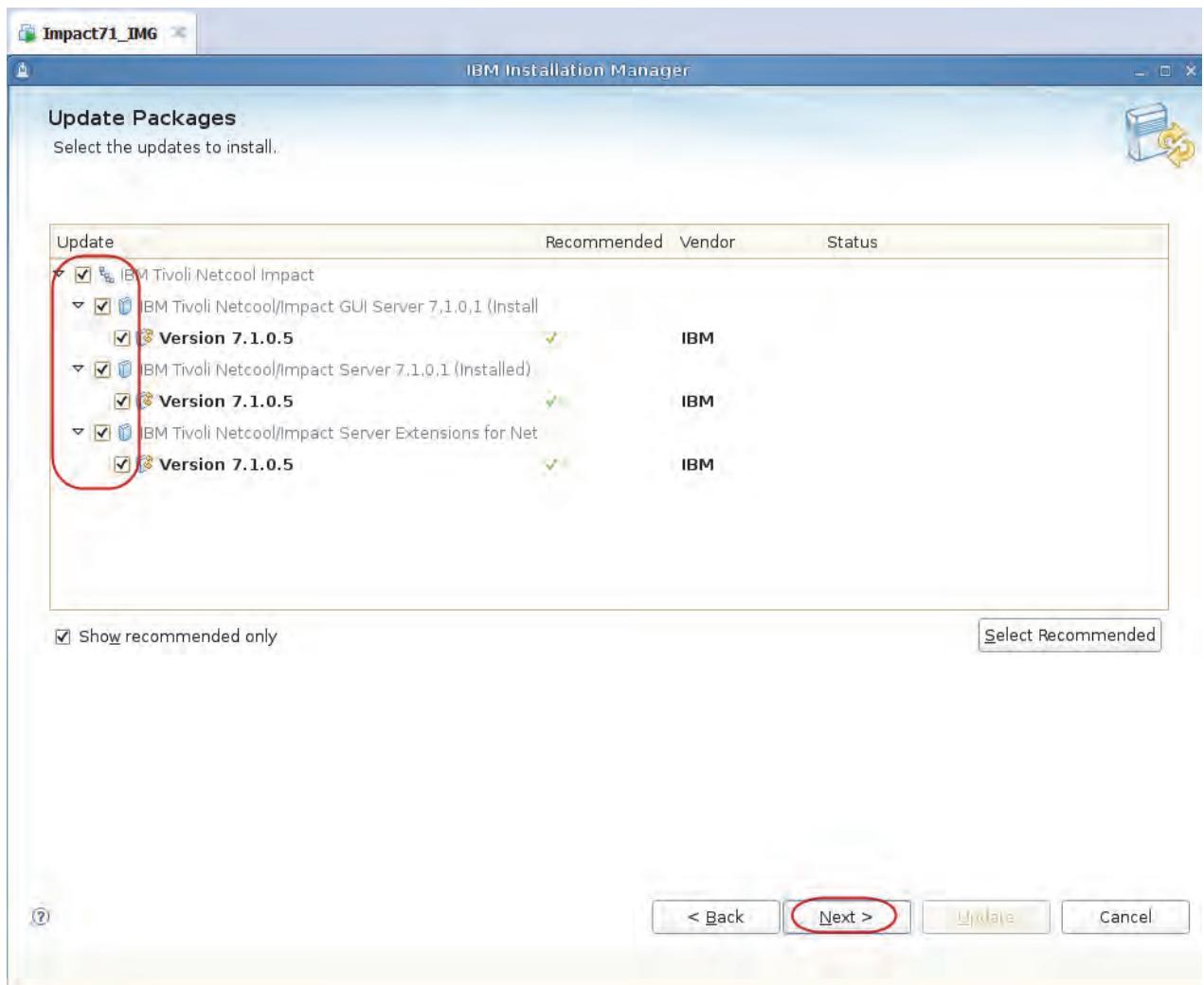
- IBM Tivoli Netcool/Impact



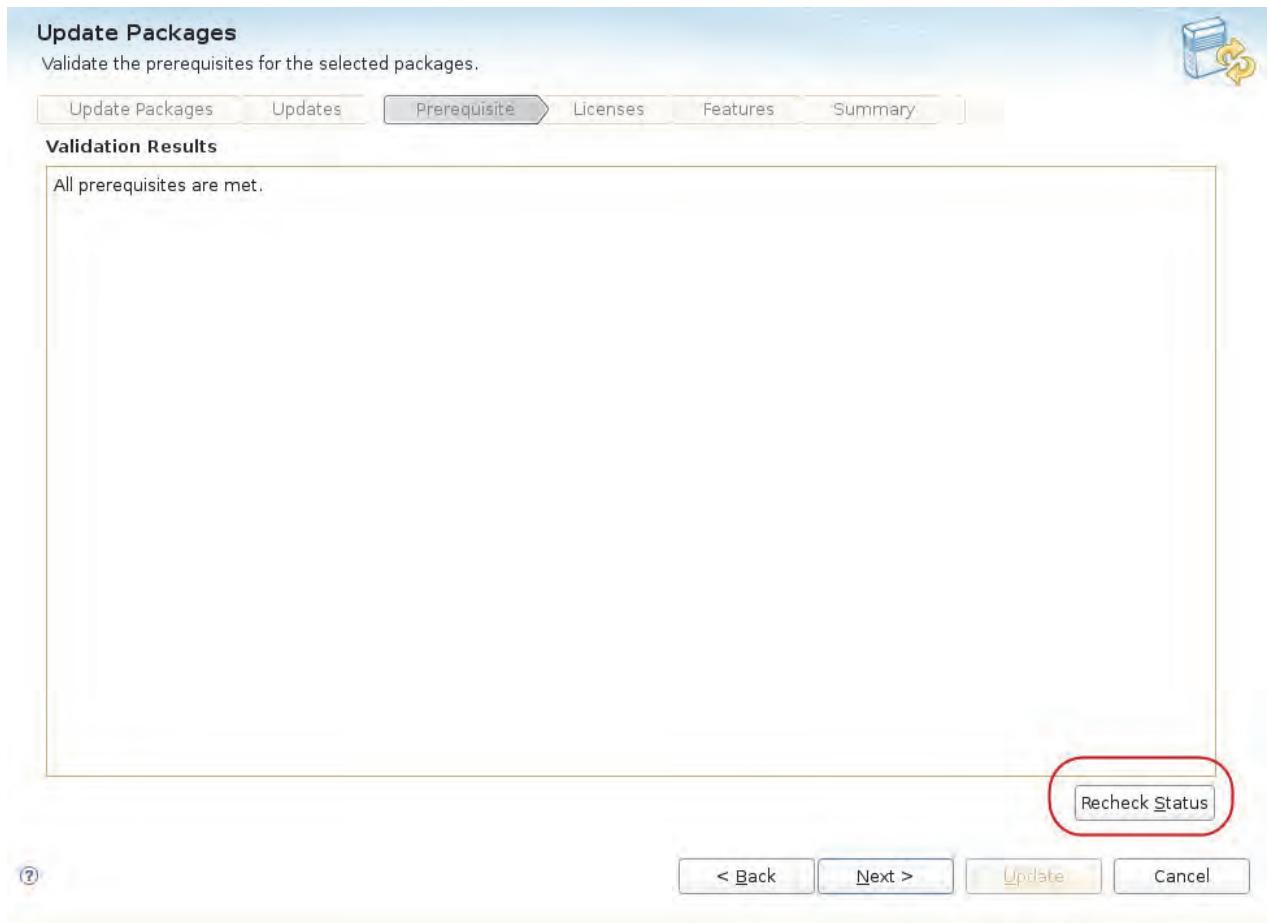
8. Select the following installation packages:

- IBM Tivoli Netcool/Impact GUI Server
- IBM Tivoli Netcool/Impact Server
- IBM Tivoli Netcool/Impact Server Extensions for Netcool Operations Insight

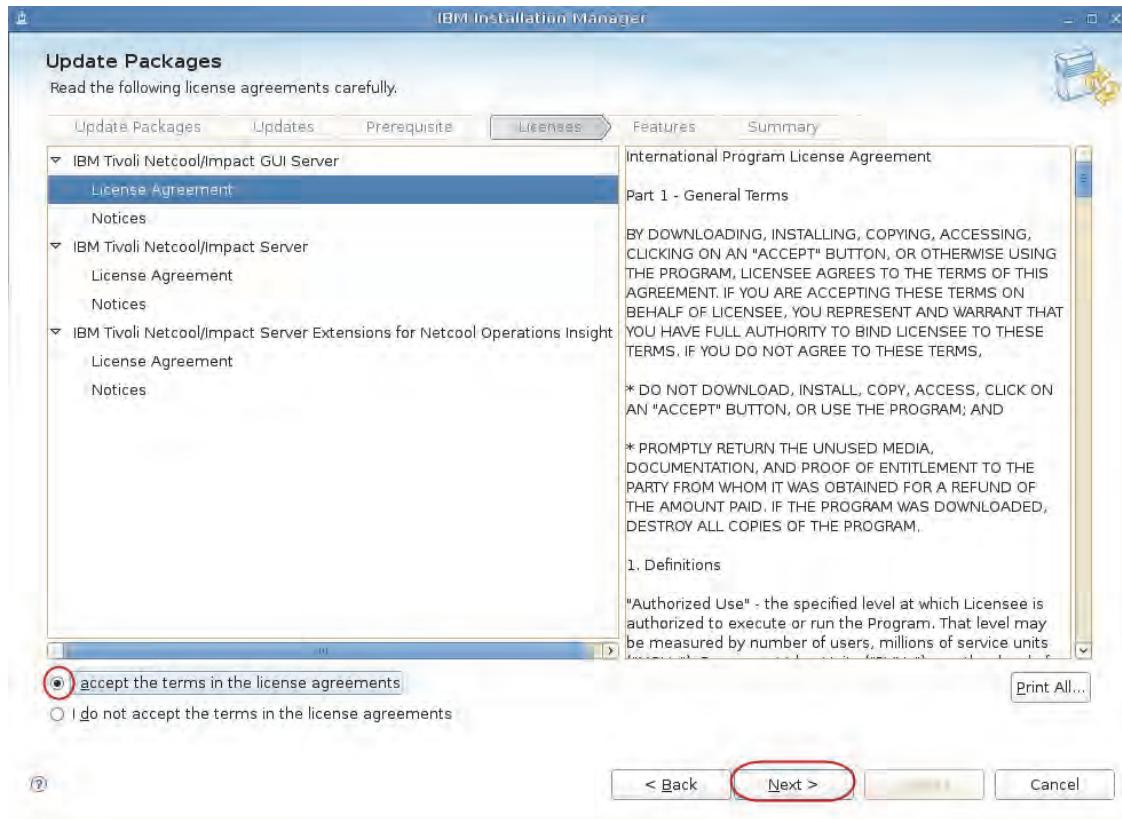
9. Click Next.



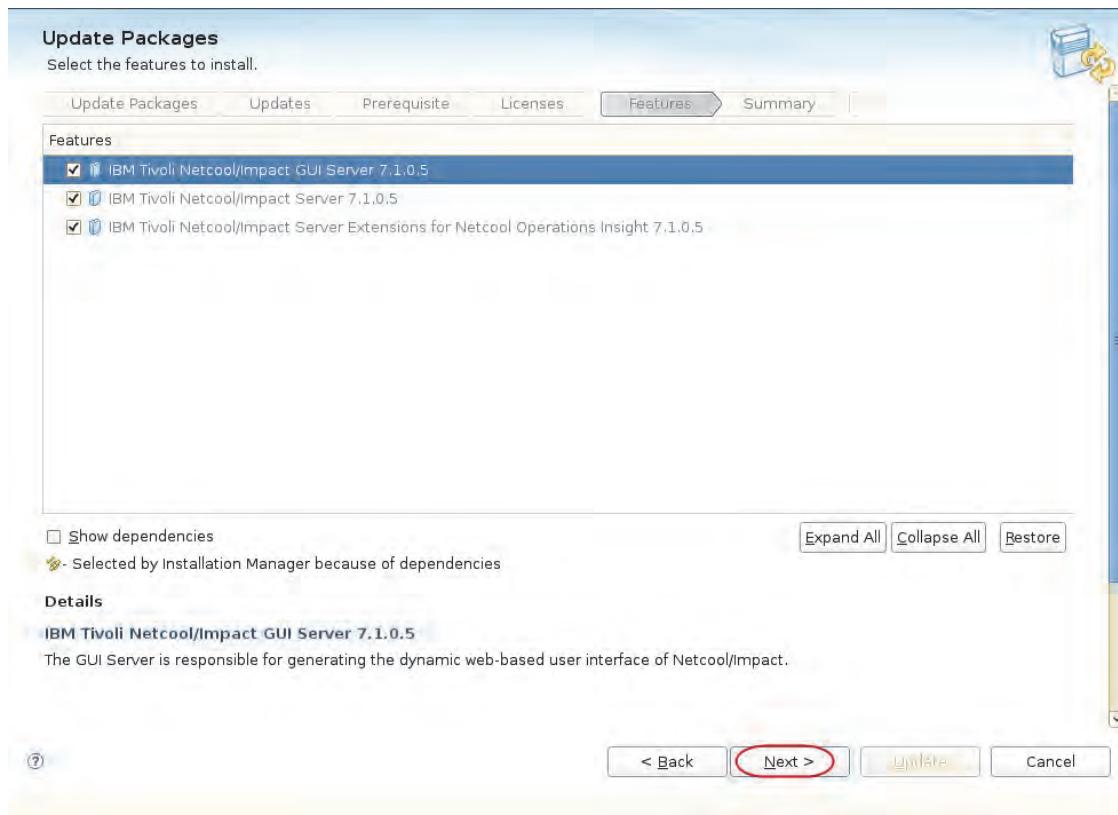
10. Click **Recheck Status** to verify prerequisites.



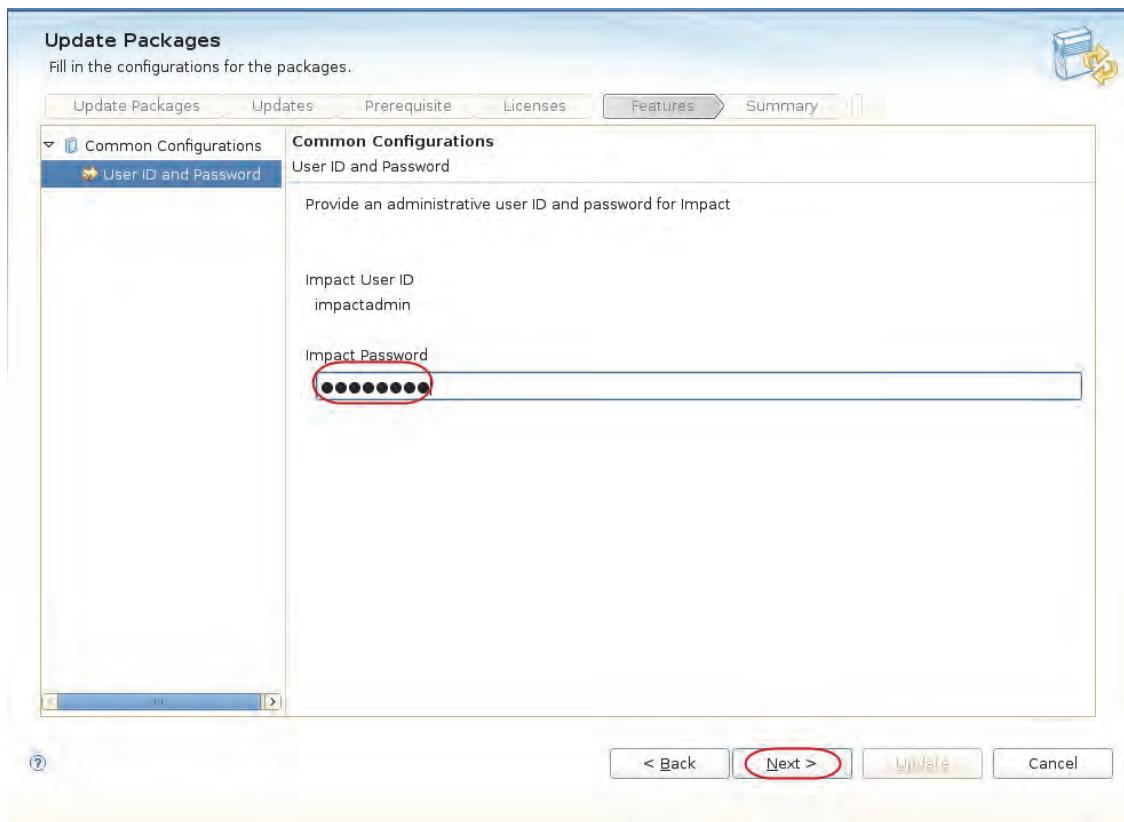
11. Select accept the terms in the license agreements and click **Next**.



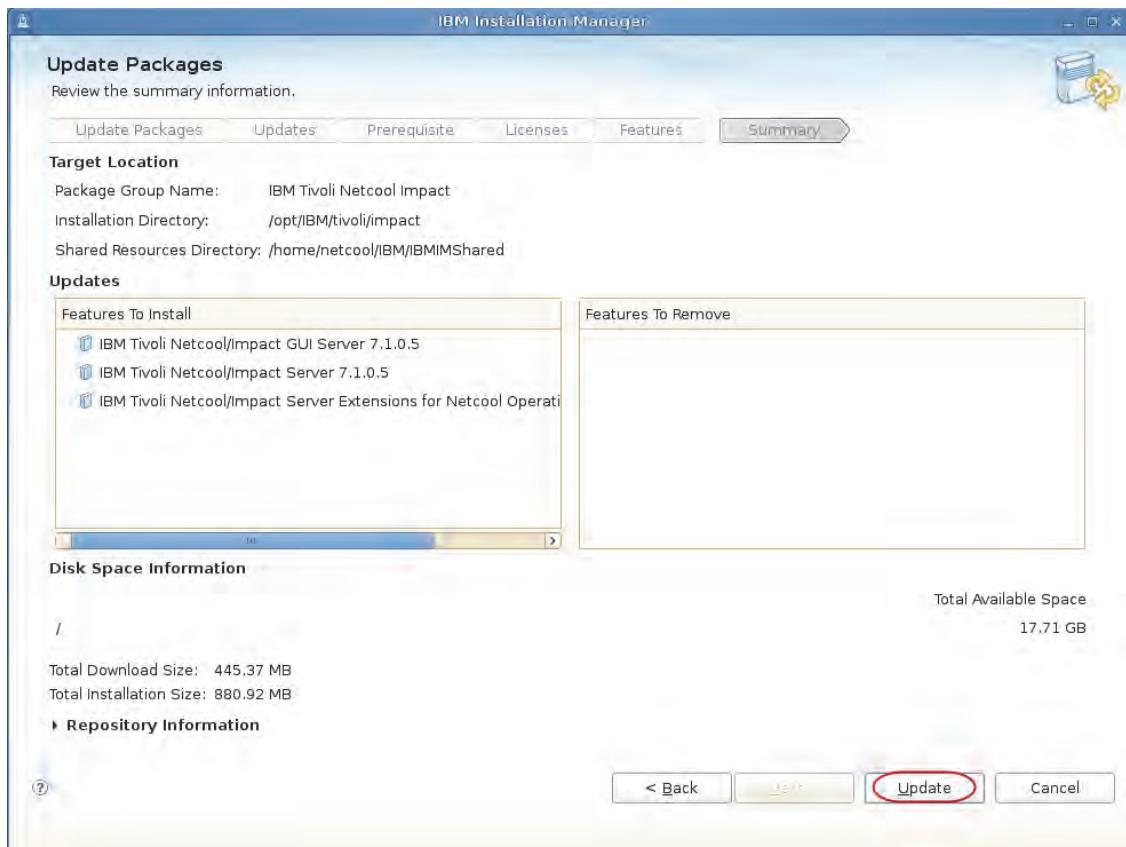
12. Verify that three features are selected. Click **Next**.



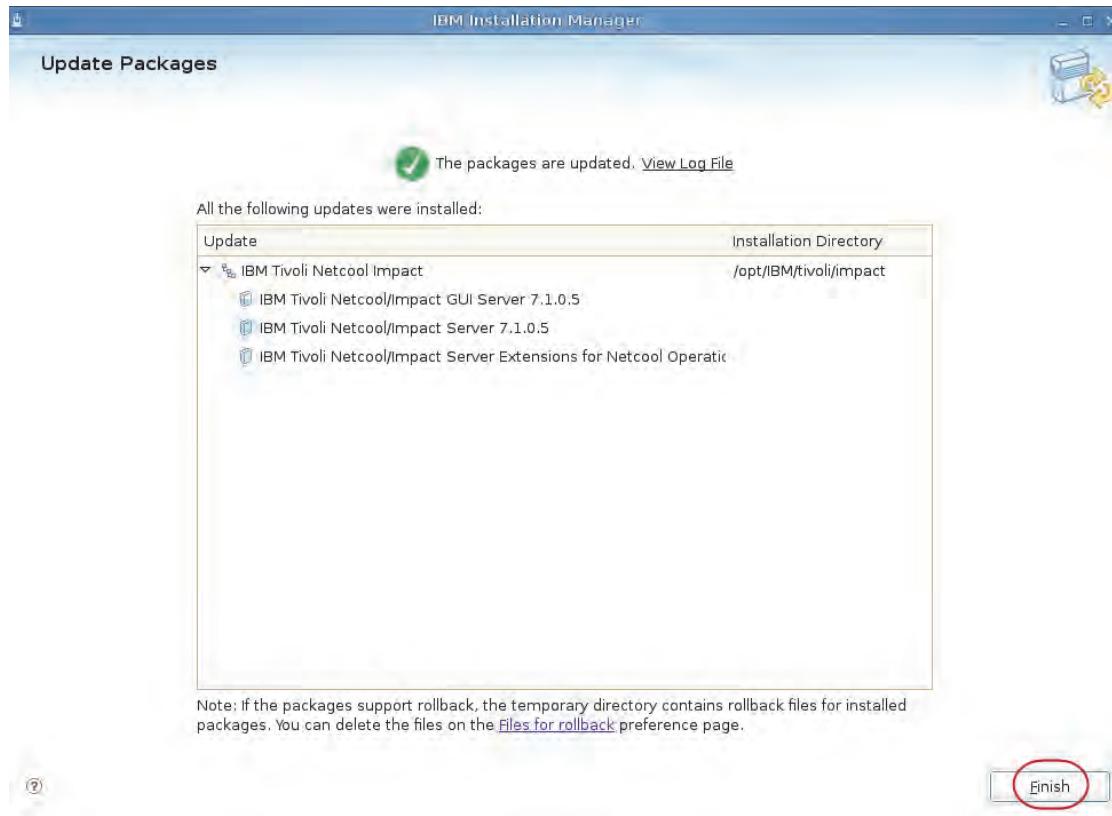
13. Type **object00** in the Impact Password field. Click **Next**.



14. Click **Update** to install fp5.



15. Verify that the update completed successfully. Click **Finish**.



## Exercise 4 Verifying Fix Pack installation

1. Select **Firefox** in the **Applications** tab.



2. Enter the default Netcool/Impact web console address:

<https://jazz01.tivoli.edu:16311/ibm/console/>

3. Log in to the Netcool/Impact console as the user **impactadmin**, with the password **object00**. Click **Go**.



4. Notice that the product version changed from 7.1.0.1 to 7.1.0.5.

The image is a screenshot of a Mozilla Firefox browser window displaying the IBM Netcool/Impact web console. The title bar says "IBM Netcool/Impact - Mozilla Firefox". The address bar shows the URL "https://jazzd1.tivoli.edu:16311/ibm/console/". The main content area shows the "Getting started" page for version 7.1.0.5. A red arrow points from the address bar to the product version number "7.1.0.5" in the header. The page includes a "Thank you for using IBM Tivoli Netcool/Impact" message and a brief description of the product's features. Below this is a section titled "Managing Netcool/Impact" with three icons: a blue square, a document icon, and a gear icon.

5. Exit Impact. Click the person icon in the lower left side of the console and select **Logout**.



6. Close the browser. Select the **File > Quit** from the menu.

# Exercise 5 Initializing the Derby database

Use the Derby database as a data source. The next few steps prepare the database for class exercises.

1. The initialization files, **Impact71DERBYSetup.sql** and **OVTABLESETUP.sql**, are in the **/labfiles** directory. Enter the following commands to move the files to a Netcool/Impact directory:

```
cd /labfiles
cp Impact71DERBYSetup.sql /opt/IBM/tivoli/impact/bin
cp OVTABLESETUP_DERBY.sql /opt/IBM/tivoli/impact/bin
```

2. Change directory to **/opt/IBM/tivoli/impact/bin** and enter the following command:

```
./ncl_db connect
```

3. At the prompt, type the following command:

```
connect 'jdbc:derby://jazz01.tivoli.edu:1527/ImpactDB;user=impact;password=derbypass';
```

```
netcool@jazz01:/opt/IBM/tivoli/impact/bin> ./ncl_db connect
ij version 10.8
ij> connect 'jdbc:derby://jazz01.tivoli.edu:1527/ImpactDB;user=impact;password=derbypass';
ij>
```

4. Run the script by entering the following command at the **ij>** prompt:

```
run 'Impact71DERBYSetup.sql';
```

5. Verify the script operations by entering the following command at the **ij>** prompt:

```
select * from inventory;
```

PARTID	STATUS	NODE  TYPE	INTERFACE  VERSION
DAL-99e309	1	gambit  Cisco	IOS 12.0
DAL-93488f	1	orac  Sun	1  Freeware
DAL-934r588f	1	wombat  Sun	1  Freeware
DAL-934r88f	1	muppet  Sun	1  Freeware
DAL-9345488f	1	moose  Sun	1  Freeware

6. Start the second script at the **ij>** prompt by entering the following command:

```
run 'OVTABLESETUP_DERBY.sql';
```

7. Verify the script operations by entering the following command at the **ij>** prompt:

```
select * from ASSET;
```

```
ij> select * from ASSET;
DEVICE
|OPHONE |CCPHONE |OWNER |OEMAIL |CUSTCONTACT |CCEMAIL |CUSTOMER
-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+
192.168.1.119 |ACME TELCOM
|Server Adminperson
|212-555-3331 |saadmin@vacmtel.com
|Device Ownerperson
|609-555-1234 |downer@acmtel.com

1 row selected
ij>
```

8. Close the Derby database command interpreter by entering the following command:

```
quit;
```

# Exercise 6 Modifying the OMNIbus Object server

1. Confirm that the OMNIbus ObjectServer server is started.

2. Change directory to the following location:

/opt/IBM/tivoli/netcool/omnibus/bin/

3. Enter either of the following commands:

./nco\_ping NCOMS

or

ps -ef | grep nco

4. If the OMNIbus server is *not* running, start OMNIbus with the following command:

/opt/IBM/tivoli/netcool/omnibus/bin/nco\_objserv NCOMS &

5. Use the **nco\_ping NCOMS** command to verify that the ObjectServer is running.

6. Verify that the **addfields.sql** script exists in the **/labfiles** directory. Enter the following command:

cat /labfiles/addfields.sql

```
netcool@jazz01:/opt/IBM/tivoli/impact/bin> cat /labfiles/addfields.sql
ALTER TABLE alerts.status ADD COLUMN PartNumber VARCHAR(64);
ALTER TABLE alerts.status ADD COLUMN Department VARCHAR(64);
ALTER TABLE alerts.status ADD COLUMN SLA VARCHAR(64);
ALTER TABLE alerts.status ADD COLUMN WorkingHours VARCHAR(64);
ALTER TABLE alerts.status ADD COLUMN ImpactFlag1 INTEGER;
ALTER TABLE alerts.status ADD COLUMN ImpactFlag2 INTEGER;
ALTER TABLE alerts.status ADD COLUMN ImpactFlag3 INTEGER;
ALTER TABLE alerts.status ADD COLUMN ImpactServiceFlag INTEGER;
ALTER TABLE alerts.status ADD COLUMN IMSent INTEGER;
go
```

7. Change directory to **/opt/IBM/tivoli/netcool/omnibus/bin**.

8. Start the **addfields.sql** script with the following command:

./nco\_sql -server NCOMS -user root -password 'object00' < /labfiles/addfields.sql

```
netcool@jazz01:/opt/IBM/tivoli/impact/bin> cd /opt/IBM/tivoli/netcool/omnibus/bi
n
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ./nco_sql -server NCOMS -use
r root -password 'object00' < /labfiles/addfields.sql
(0 rows affected)
```

The message **0 rows affected** indicates that the script commands were processed.



**Note:** If the nco\_sql command indicates errors in the file, use the command **dos2unix addfields.sql** to convert the text file to the proper format.

# Unit 2 The Netcool/Impact user interface exercises

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

## Exercise 1 Starting the Netcool/Impact components

1. To confirm that the OMNIbus server is running, change to the following directory:

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

2. Enter either of the following commands:

```
./nco_ping NCOMS
```

OR

```
ps -ef | grep nco
```

3. If the OMNIbus server is *not* started, enter the following command:

```
/opt/IBM/tivoli/netcool/omnibus/bin/nco_objserv -name NCOMS &
```

4. Use the **nco\_ping NCOMS** command to verify that the object server is running.

5. Verify the Netcool/Impact server, enter the following command:

```
/opt/IBM/tivoli/impact/bin/startImpactServer.sh
```

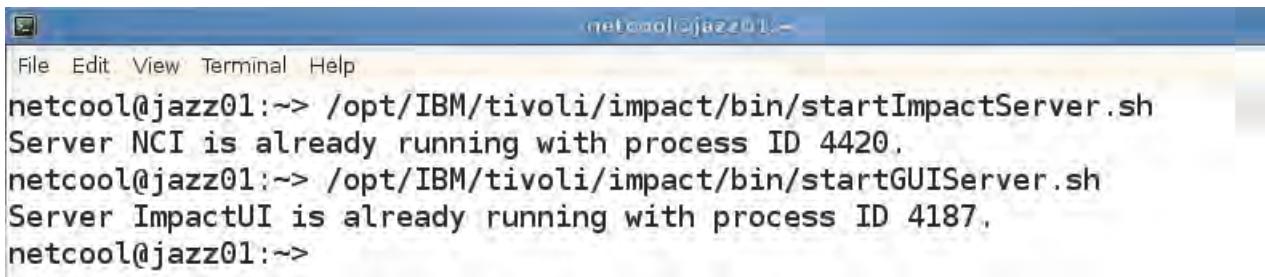
```
netcool@jazz01:/opt/IBM/tivoli/impact/bin> /opt/IBM/tivoli/impact/bin/startImpactServer.sh
Starting server NCI.
Server NCI started with process ID 11295.
Started Impact Server NCI
```

6. To start the Netcool/Impact GUI server, enter the following command:

```
/opt/IBM/tivoli/impact/bin/startGUIServer.sh
```

```
netcool@jazz01:/opt/IBM/tivoli/impact/bin> /opt/IBM/tivoli/impact/bin/startGUIServer.sh
Starting server ImpactUI.
Server ImpactUI started with process ID 11694.
Started GUI Server ImpactUI
```

If the servers are already running the commands will display the process ID similar to the following:



```
netcool@jazz01:~> /opt/IBM/tivoli/impact/bin/startImpactServer.sh
Server NCI is already running with process ID 4420.
netcool@jazz01:~> /opt/IBM/tivoli/impact/bin/startGUIServer.sh
Server ImpactUI is already running with process ID 4187.
netcool@jazz01:~>
```

## Exercise 2 Creating a project

1. Open the Firefox web browser to log into the Netcool/Impact console. Start the browser and click the home page icon in the upper right.

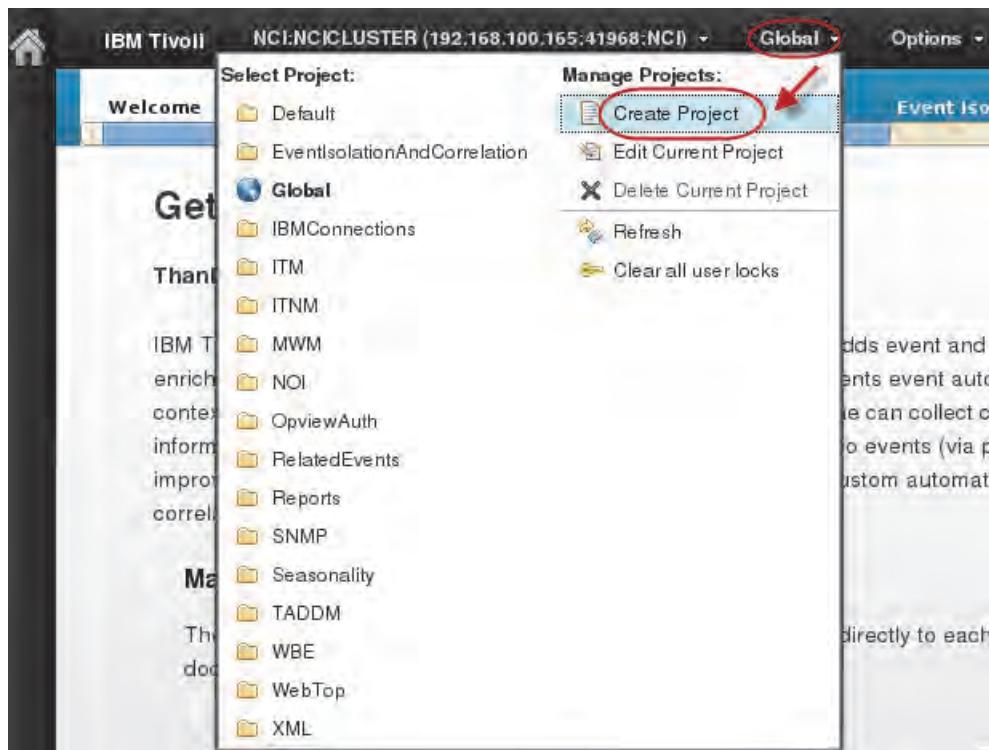


2. Enter **impactadmin** in the **User name** field, **object00** in the **Password** field, and click **Go**.



The image shows the IBM Netcool/Impact login page. It has a dark background with white text. There are two input fields: 'User name' containing 'impactadmin' and 'Password' containing 'object00'. Both fields are circled with a red line. Below the fields is a blue 'Go' button, which is also circled with a red line. At the bottom left, there is a copyright notice: '@ Copyright IBM Corp. 2005, 2015.'

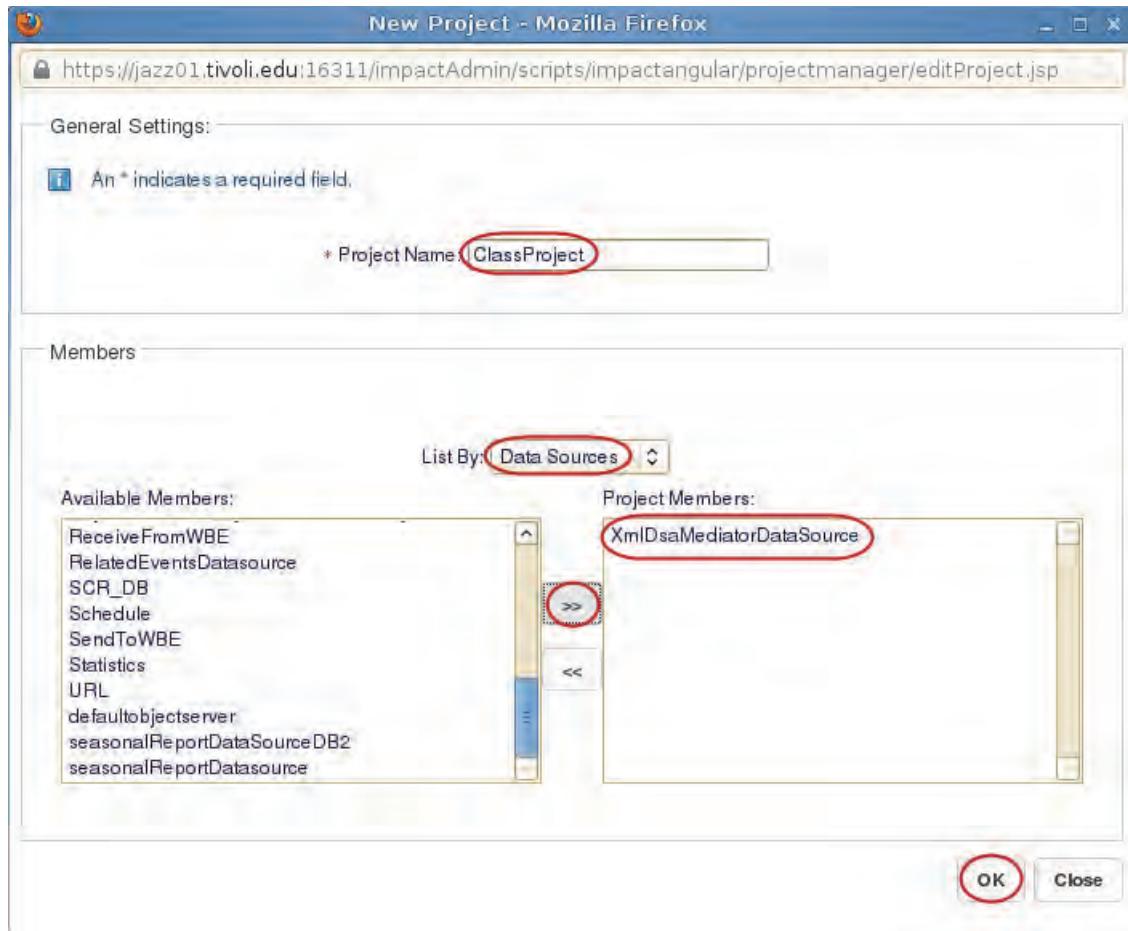
3. Create a Netcool/Impact project. Click the project list menu at the top of the console and click **Create Project** in the **Manage Projects** section.



A project configuration window opens.

4. Enter **ClassProject** in the **Project Name** field in the **General Settings** section.

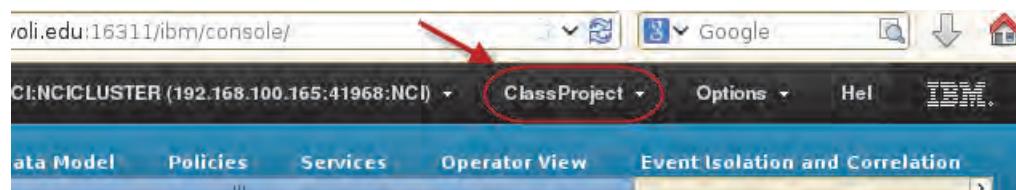
5. In the **Member Selection** section, complete the following steps:
  - a. Select **Data Sources** in the **List By** menu.
  - b. Scroll down in the **Global Repository** list, click **XmIDsaMediatorDataSource**, click **>>** to add it to the **Project Members** list and click **OK**.



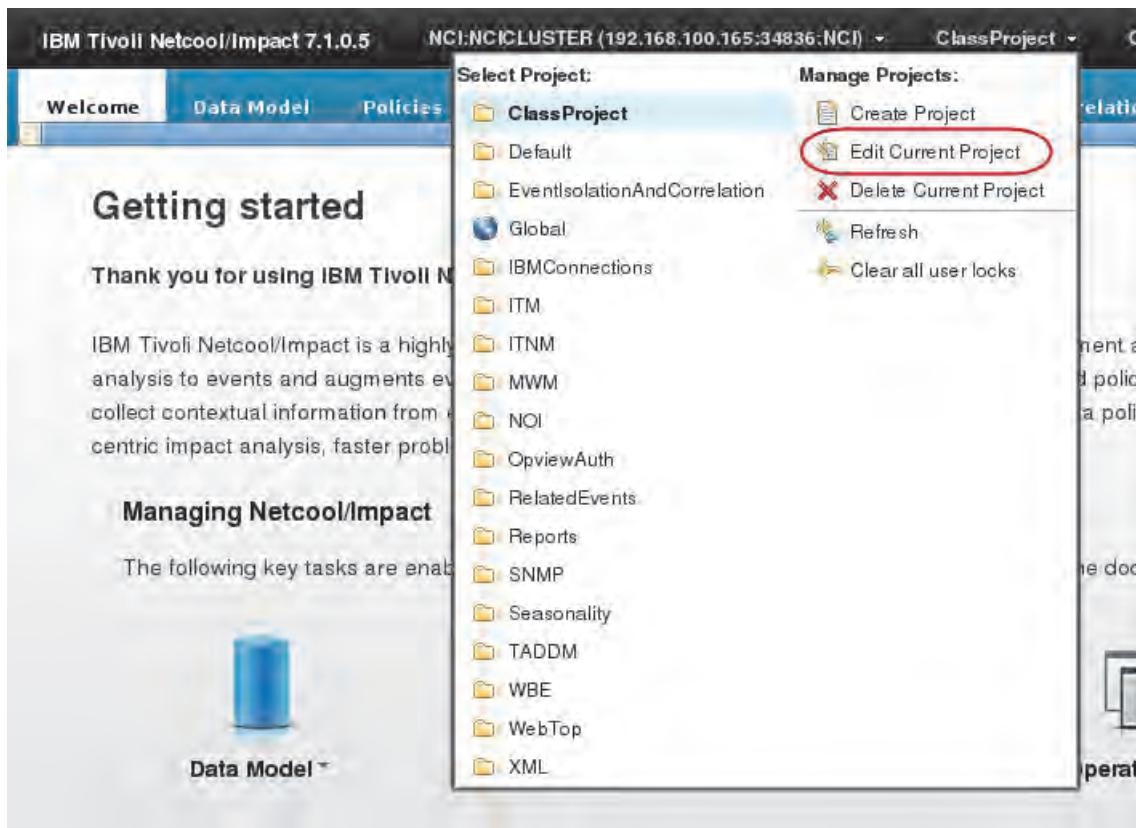
6. Click **OK** in the confirmation window.



**ClassProject** is shown as the current project in the project menu at the top of the console.

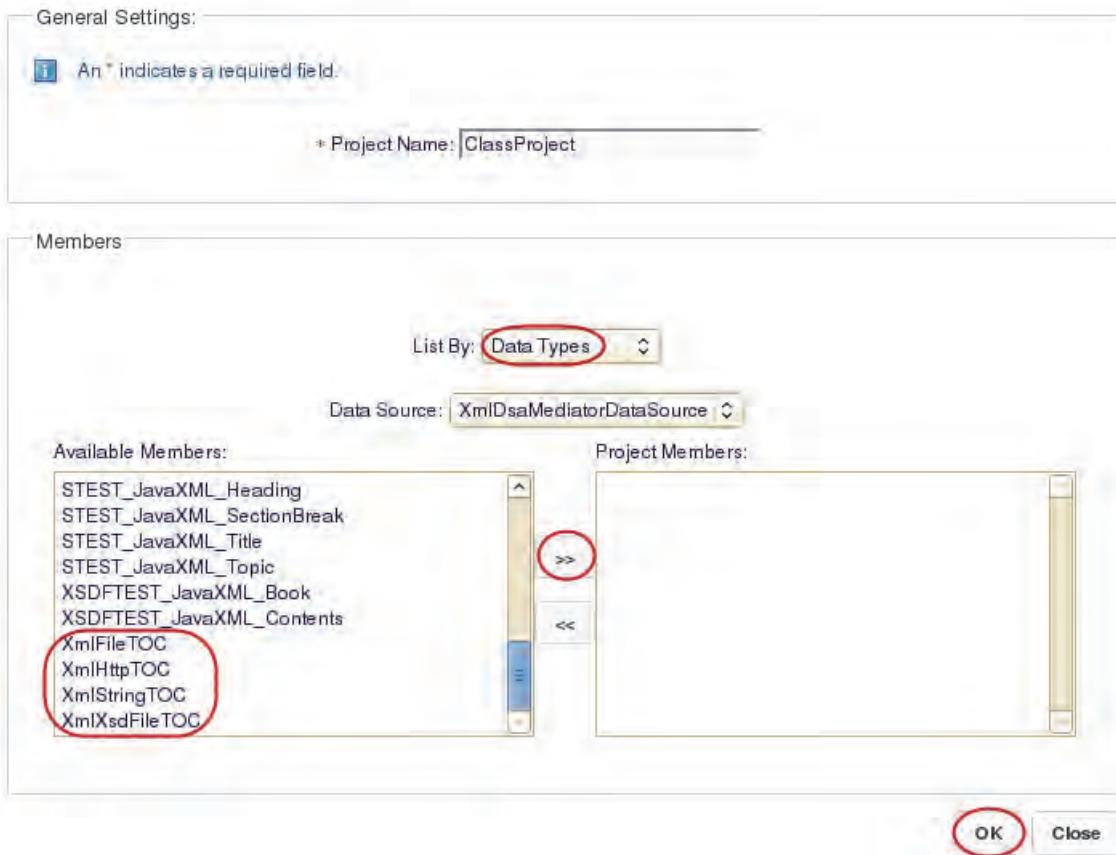


7. Edit the ClassProject configuration. Click the **ClassProject** menu and click **Edit Current Project** in the **Manage Projects** section.



8. Select **Data Types** in the **List By** menu. Because **XmIDsaMediatorDataSource** is the only data source in the project, it is automatically shown in the Data Source menu.

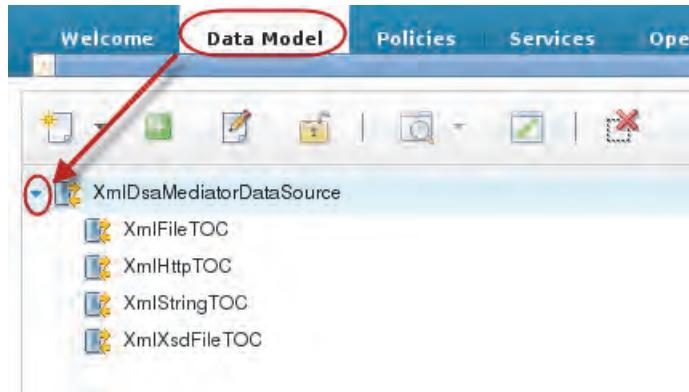
9. Add all of the available data types to the data source. Select all data **xml** data types in the **Available Members** list, click **>>** to add them to **Project Members**, and click **OK**.



10. Click the **Data Model** tab in the toolbar at the top of the console to show the list of data sources in the current project.

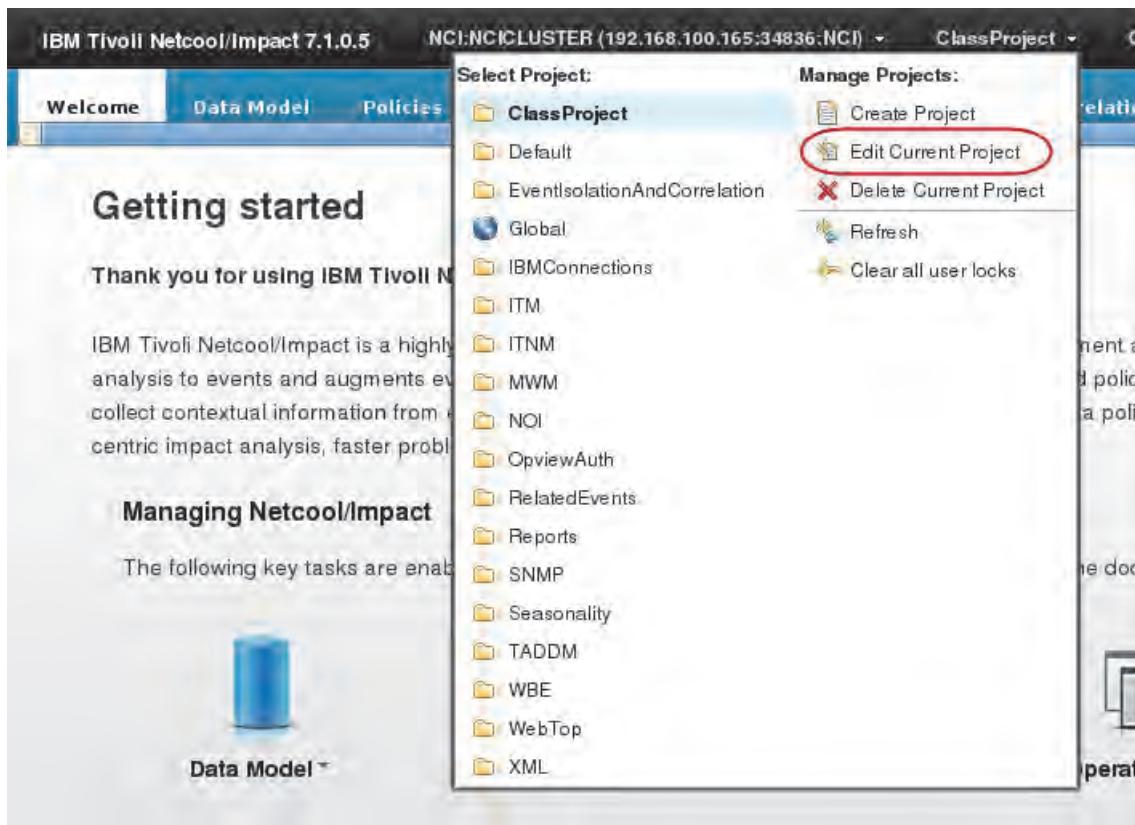


11. Click the arrow to the left of **XmIDsaMediatorDataSource** to show the list of configured data types.

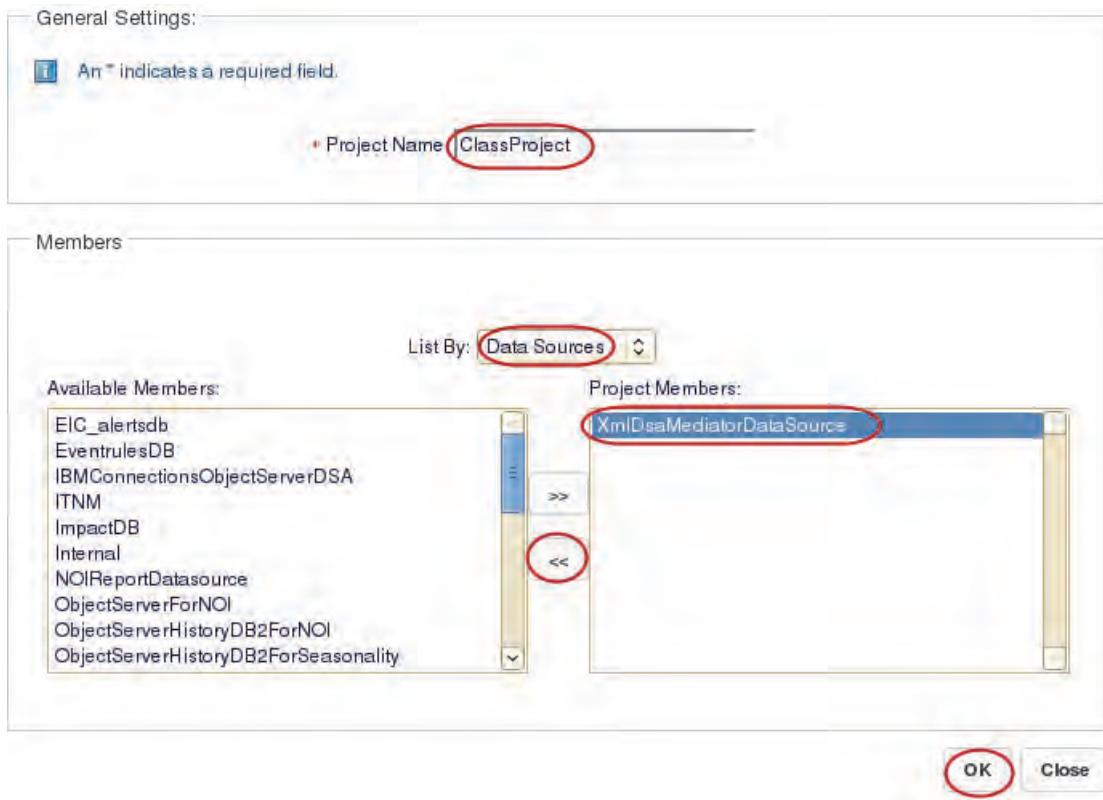


## Exercise 3 Removing project members

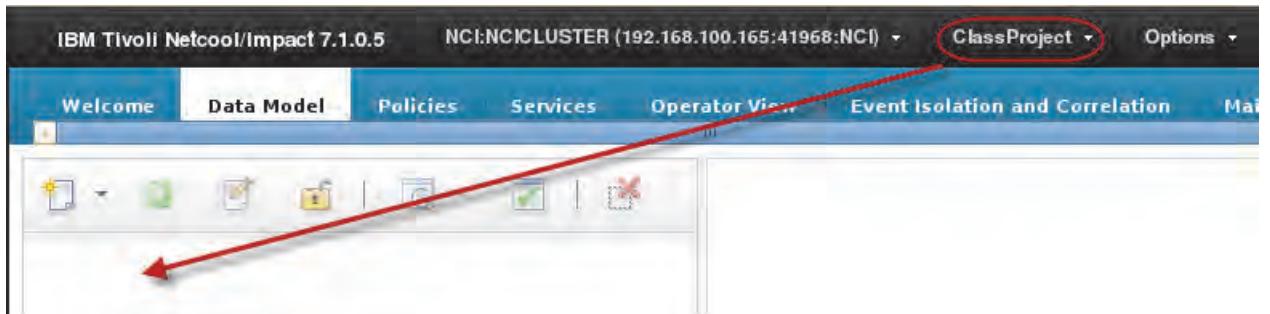
1. Edit the ClassProject project. Click **ClassProject** in the project menu and select **Edit Current Project**.



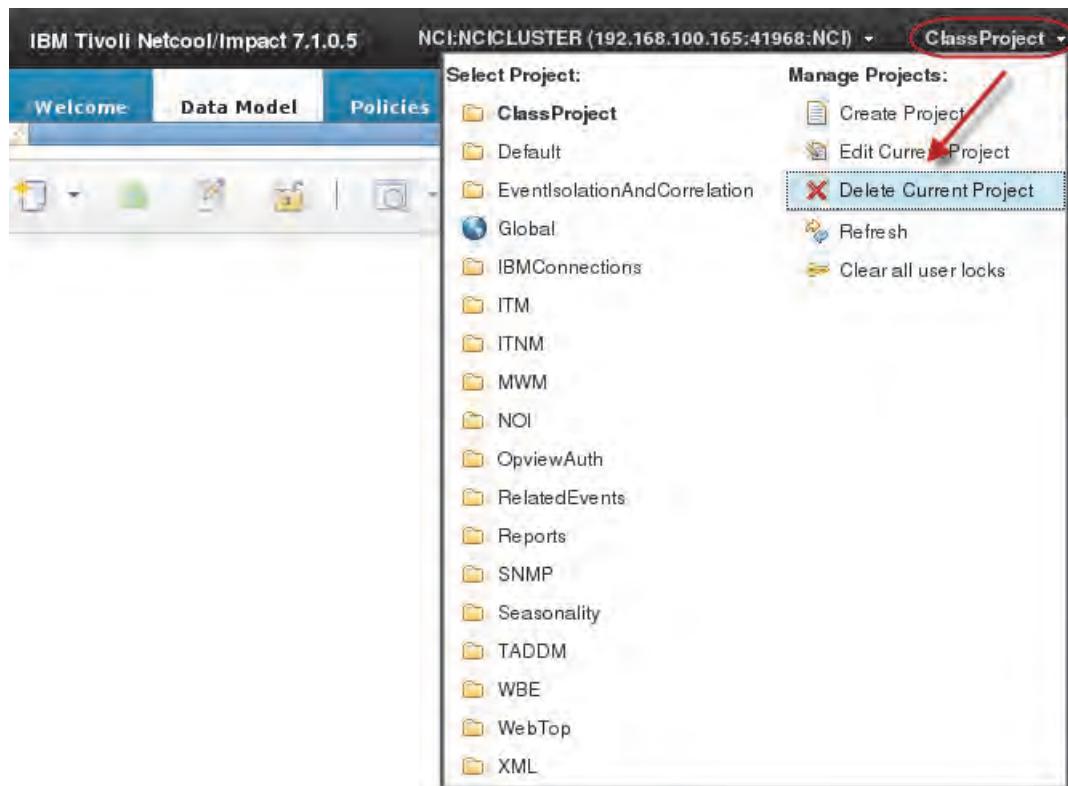
2. Remove the XmlDsaMediatorDataSource data source from the project.
  - a. Select **Data Sources** in the **List By** section.
  - b. Click **XmlDsaMediatorDataSource** in the **Project Members** list.
  - c. Click the double arrows (**<>**) and click **OK**.



The console indicates that there is no data to show in the **Data Model** tab.



3. Delete the ClassProject project. Click **ClassProject** in the project menu and select **Delete Current Project**.



4. Click **OK** in the confirmation window.



**ClassProject** is removed from the project menu and the current project reverts to **Global**.





# Unit 3 The Netcool/Impact data model exercises

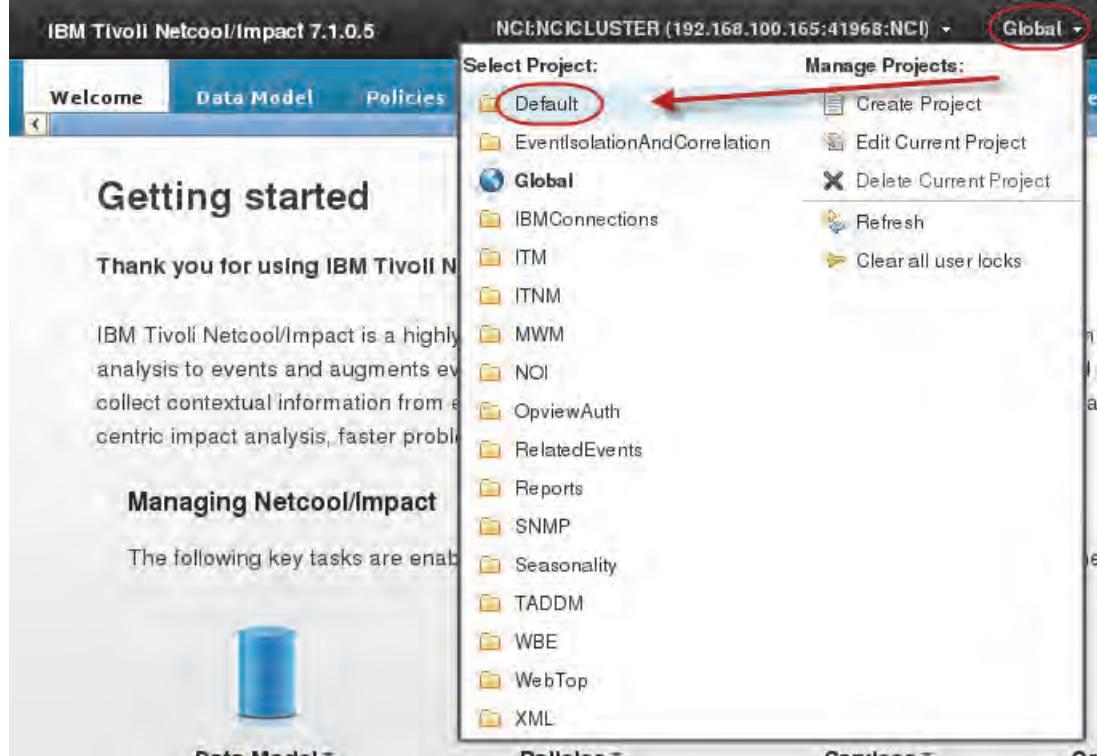
The focus of this unit is the Netcool/Impact Data Model.

## Exercise 1 Creating the HS\_TRAIN data source

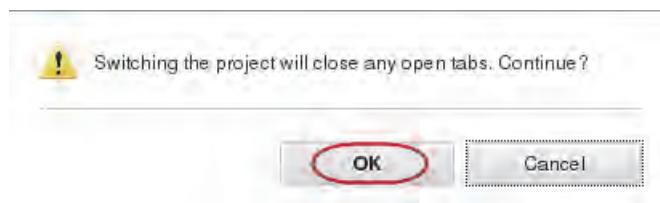
You must create a data source definition for the database that holds the sample data. The sample data was loaded into the Derby database during an earlier exercise.

1. Log on to the Netcool/Impact console as the user **impactadmin** and the password **object00**.
2. Select **Default** in the project menu.

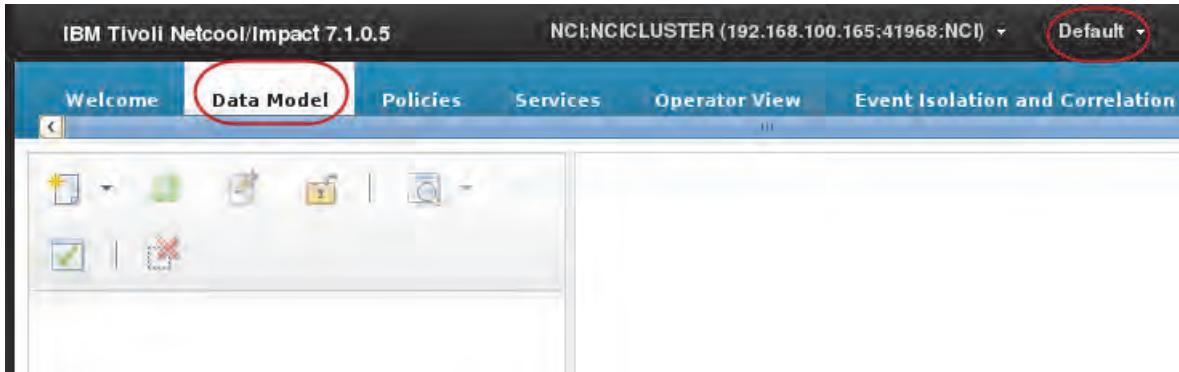
**Note:** You work in the **Default** project for the remainder of the class.



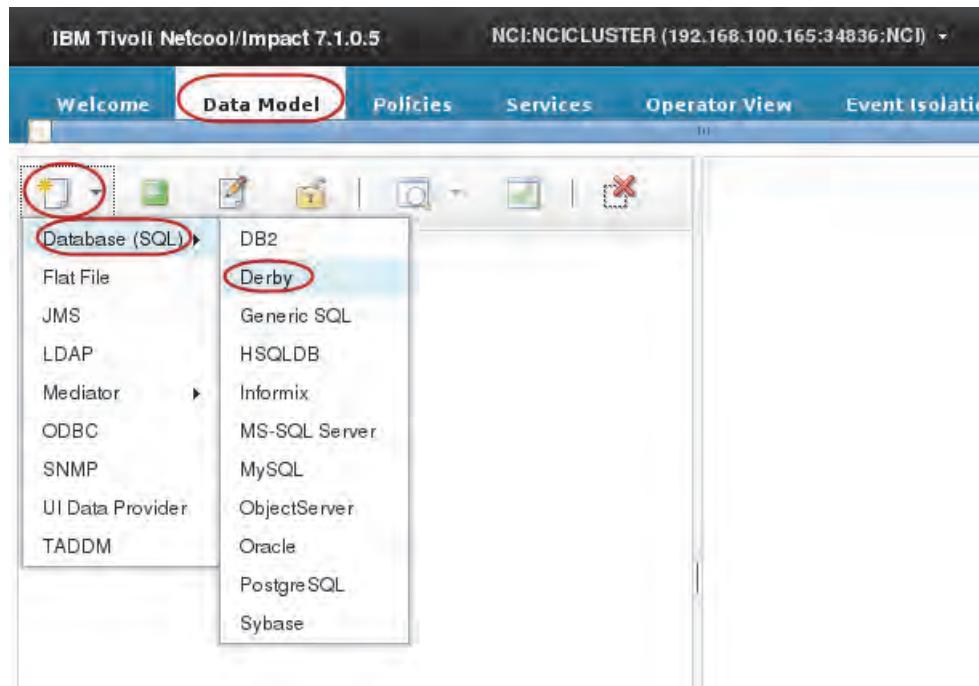
3. Click **OK** in the confirmation window.



4. Click **Data Model** to open the **Data Model** page.



5. Click the **New Data Source** icon and select **Database (SQL) > Derby**.

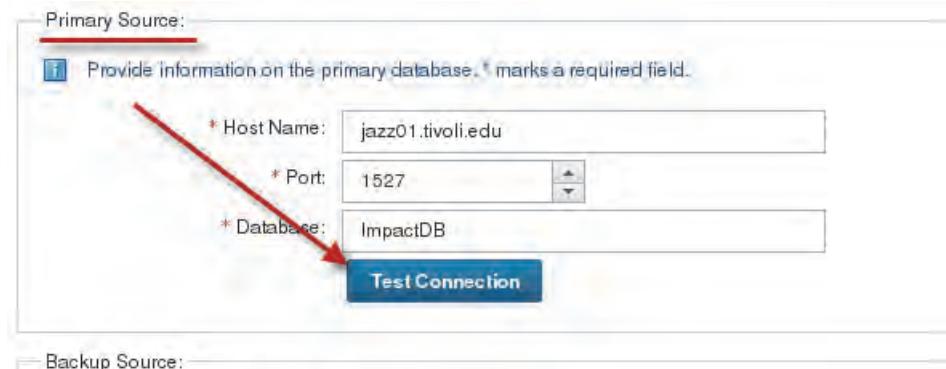


6. On the right side of the page, configure the **General Settings**, **Database Failure Policy**, and the **Primary Source** panels. Use the following information:
  - General Data
    - Data Source Name **HS\_TRAIN**
    - Username **impact**
    - Password **derbypass**
    - Maximum SQL Connection **5**
  - Database Failure Policy **Disable Backup**
  - Primary Source
    - Host Name **jazz01.tivoli.edu**
    - Port **1527**
    - Database **ImpactDB**

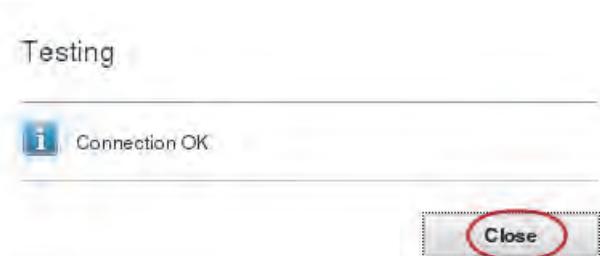
The screenshot shows the 'New Derby' configuration dialog with three main sections:

- General Settings:** Fields include Data Source Name (HS\_TRAIN), Username (impact), Password (redacted), and Maximum SQL Connection (5).
- Database Failure Policy:** Action selected is "Disable Backup".
- Primary Source:** Fields include Host Name (jazz01.tivoli.edu), Port (1527), Database (ImpactDB), and a highlighted "Test Connection" button.

7. In the Primary Source section, click **Test Connection**.



8. When the Information window opens, click **OK**.



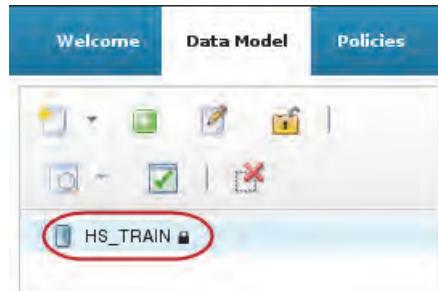
9. Click the **Save** icon to save the data source configuration.



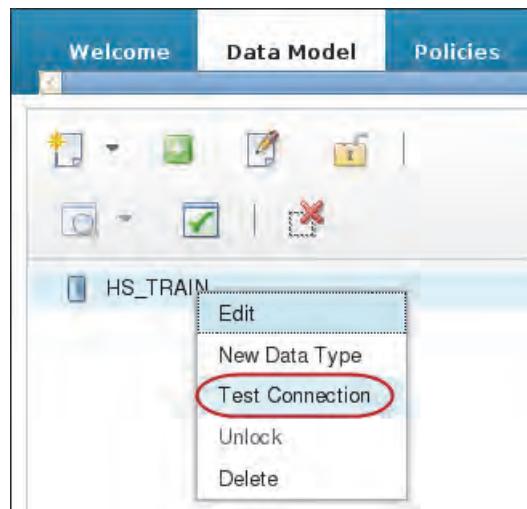
10. Click **Close** if the window does not clear.



You can edit the data source definition after you save it. While the data source is opened in edit mode, the GUI shows a lock symbol to the right of the **HS\_TRAIN** data source.

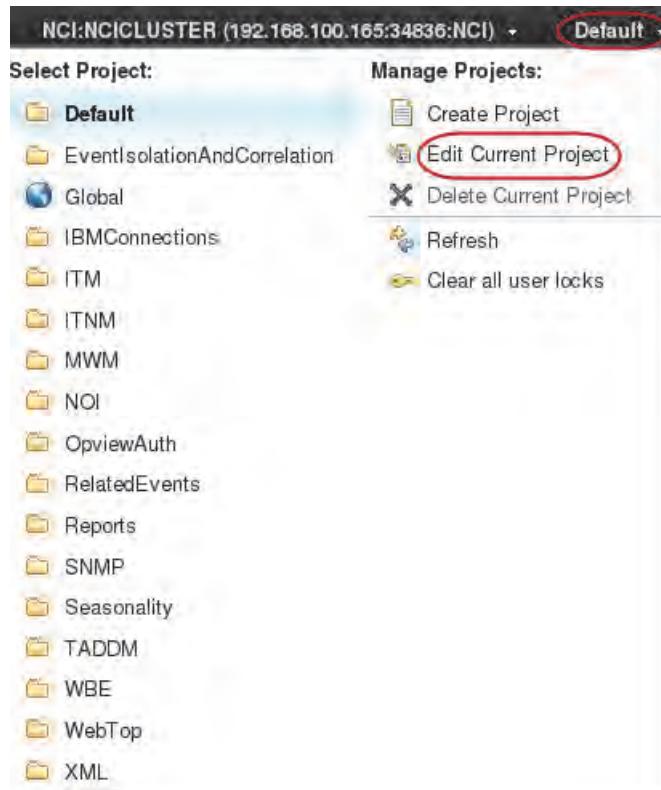


You can also test the connection by right-clicking the data source name and selecting **Test Connection**.



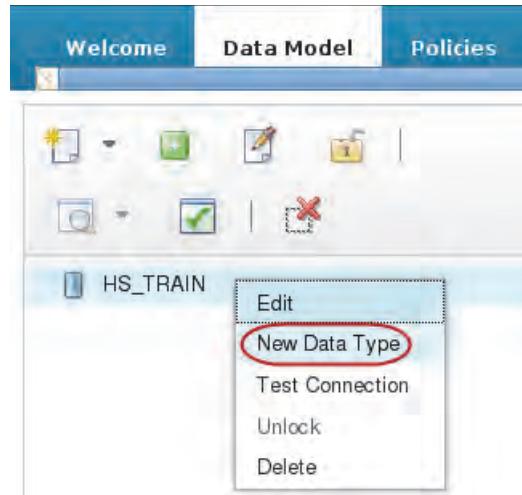


**Note:** If the **HS\_TRAIN** data source is not shown in the **Default** project, check the Global Repository by clicking the **Edit Project** icon. Add the **HS\_TRAIN** data source to the **Default** project.



## Exercise 2 Creating data types for the HS\_TRAIN data source

1. Right-click the HS\_TRAIN data source name, and select New Data Type.



On the right side of the browser window, the data type configuration panel opens.

2. Enter **HS\_Node** for the **Data Type Name**.
3. Select **IMPACT** from the **Base Table** list.
4. To the right of the **Base Table** list, select **NODE** from the list on the right.



**Important:** If an error occurs in this section clear the browser cache and try again.

New Data Type for HS\_TRAIN

SQL Data Type Config Page

Table Description      Dynamic Links      Cache Settings

General Settings:

Provide general information which describes the data type. An \* indicates required fields.

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

Tables: NODE

Refresh Fields

Show New / Deleted Fields

Delete Selection      New      Edit      Move Up      Move Down

5. Click **Refresh Fields** to display the table structure.
6. Select **NODE** as the **Key Field**. Double click **No** to display the checkbox. Click again to show the check-mark.

SQL Data Type Config Page

Table Description      Dynamic Links      Cache Settings

ID	Field Name	Format	Display Name	Description	Key Field
<input type="checkbox"/>	NODE	STRING	NODE	NODE	<input checked="" type="checkbox"/>

7. Click away from the row to show Yes. Click the Save icon.

The screenshot shows the 'SQL Data Type Config Page' for the 'NODE' table. The 'Table Description' tab is selected. The 'Tables' dropdown shows 'NODE'. Below it is a 'Refresh Fields' button, which is circled in red. A checkbox for 'Show New / Deleted Fields' is also present. The main area displays a table of fields:

ID	Field Name	Format	Display Name	Description	Key Field
NODE	NODE	STRING	NODE	NODE	Yes
IP	IP	STRING	IP	IP	No
NODEID	NODEID	INTEGER	NODEID	NODEID	No
CUSTID	CUSTID	INTEGER	CUSTID	CUSTID	No
DEPTID	DEPTID	INTEGER	DEPTID	DEPTID	No

8. Verify that **HS\_Node** is listed under the data source **HS\_TRAIN** in the data model.

The screenshot shows the 'Data Model' console. The 'HS\_TRAIN' data source is expanded, showing its contents. The 'HS\_Node' data type is highlighted and circled in red. The 'HS\_TRAIN' tab and its close symbol are also circled in red.

9. Close the open page tabs in the right side of the console. Click the close symbol (X) in the **HS\_TRAIN** and **HS\_Node** tabs in the right side of the page.

The screenshot shows the 'Operator View' console. The 'HS\_TRAIN' and 'HS\_Node' tabs are closed, indicated by their respective close symbols (X) circled in red.

10. Verify that the data type definition is functional by right-clicking **HS\_Node** and selecting **View Data Items**. Table values display in the right pane for the **HS\_Node** data type.

The screenshot shows the Netcool/Impact Data Model interface. The top navigation bar includes Welcome, Data Model (selected), Policies, Services, Operator View, Event Isolation and Correlation, and Maintenance. The left sidebar shows a tree structure with HS\_TRAIN selected, which contains HS\_Node. A context menu is open over HS\_Node, with the 'View Data Items' option highlighted and circled in red. The main pane displays the 'Data Type Name: HS\_Node' and a table titled 'Number of Objects: 43'. The table has columns: Select, NODE, IP, NODEID, CUSTID, DEPTID, SERVID, View Links, and Edit. The data rows are:

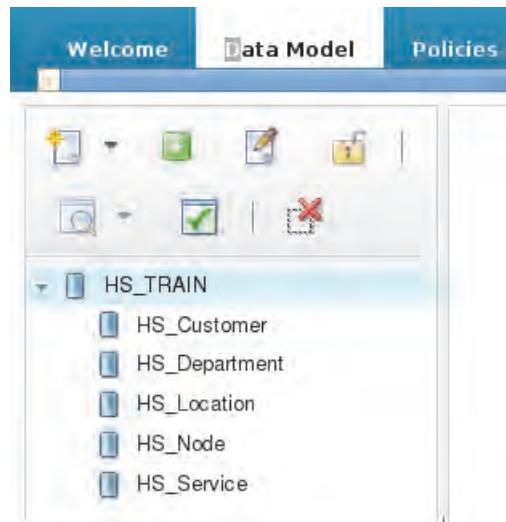
Select	NODE	IP	NODEID	CUSTID	DEPTID	SERVID	View Links	Edit
<input type="checkbox"/>	gambit	10.10.10.10	0	1	1	1		
<input type="checkbox"/>	wombat	10.10.10.1	1	2	2	2		
<input type="checkbox"/>	orac	10.10.10.2	2	3	3	3		
<input type="checkbox"/>	muppet	10.10.10.3	3	4	4	0		
<input type="checkbox"/>	moose	10.10.10.4	4	5	5	5		
<input type="checkbox"/>	hal	10.10.10.5	5	6	6	6		
<input type="checkbox"/>	vixen	10.10.10.6	6	7	7	7		

11. Create four additional data types for the **HS\_TRAIN** data source by repeating the previous steps for each new data type.

Use the data type names and associated keys from the following table.

Data Type Name	Base Table Description	Key
HS_Customer	IMPACT/CUSTOMER	CUSTOMERID
HS_Department	IMPACT/DEPARTMENT	DEPTID
HS_Location	IMPACT/LOCATION	LOCATIONID
HS_Service	IMPACT/SERVICE	SERVICEID

After completion of the data types, the **Data Sources and Types** panel looks like the following screen image.



The **HS\_Customer**, **HS\_Department**, **HS\_Location**, and **HS\_Service** tables definitions in the **HS\_TRAIN** data source are shown in the following screen captures.

The data type definition for the **HS\_Customer** table looks like this example:

This screenshot shows the 'SQL Data Type Config Page' for the 'HS\_Customer' table. It has tabs for 'Table Description', 'Dynamic Links', and 'Cache Settings'. The 'Table Description' tab is active. It shows the 'General Settings' section with fields for 'Data Type Name' (HS\_Customer), 'Data Source Name' (HS\_TRAIN), and checkboxes for 'Enabled' and 'Access the data through UI data'. The 'Dynamic Links' tab shows a configuration for the 'CUSTOMER' table in the 'IMPACT' schema. The 'Cache Settings' tab is also visible. Below these tabs is a table listing fields: ID, Field Name, Format, Display Name, Description, and Key Field. The fields listed are CUSTOMER (String, Customer, Customer, No), CUSTOMERID (Integer, CustomerID, CustomerID, Yes), and CONTACTNAME (String, ContactName, ContactName, No).

The data type definition For the **HS\_Department** table looks like this example:

**General Settings:**

- \* Data Type Name: HS\_Department
- Data Source Name: HS\_TRAIN
- Enabled
- Access the data through UML

**Table Description:**

Configure the database table and fields.

ID	Field Name	Format	Display Name	Description	Key Field
DEPARTMENT	DEPARTMENT	STRING	DEPARTMENT	DEPARTMENT	No
DEPTID	DEPTID	INTEGER	DEPTID	DEPTID	Yes

The data type definition for the **HS\_Location** table looks like this example:

**General Settings:**

- \* Data Type Name: HS\_Location
- \* Data Source Name: HS\_TRAIN
- Enabled
- Access the data through UML

**Table Description:**

Configure the database table and fields.

ID	Field Name	Format	Display Name	Description	Key Field
LOCATIONID	LOCATIONID	INTEGER	LOCATIONID	LOCATIONID	Yes
LOCATION	LOCATION	STRING	LOCATION	LOCATION	No

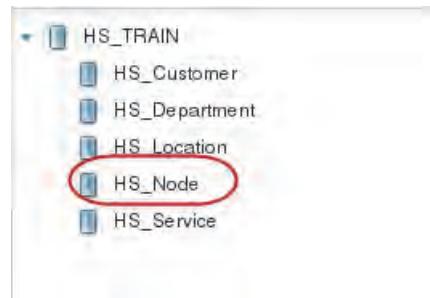
The data type definition for the **HS\_Service** table looks like this example:

The screenshot shows the 'SQL Data Type Config Page' for the 'HS\_Location' data type. It has tabs for 'Table Description', 'Dynamic Links', and 'Cache Settings'. The 'Table Description' tab is selected. Under 'General Settings', the 'Data Type Name' is set to 'HS\_Location' and the 'Data Source Name' is 'HS\_TRAIN'. The 'Enabled' checkbox is checked. In the 'Table Description' section, the schema is 'IMPACT' and the table is 'SERVICE'. A 'Refresh Fields' button is available. Below these sections is a table with columns: ID, Field Name, Format, Display Name, Description, and Key Field. The table contains three rows: SERVICEID (Field Name: SERVICEID, Format: INTEGER, Display Name: SERVICEID, Description: SERVICEID, Key Field: Yes), SERVICE (Field Name: SERVICE, Format: STRING, Display Name: SERVICE, Description: SERVICE, Key Field: No).

## Exercise 3 Creating dynamic links

In this exercise, you create four dynamic links. The first link connects the source table **HS\_Node** to the target table **HS\_Customer**, using **CUSTID** as the foreign key.

1. Double-click **HS\_Node** to edit the data item.



2. Click the **Dynamic Links** tab.

The screenshot shows the 'SQL Data Type Config Page' for 'HS\_Node'. The 'Dynamic Links' tab is highlighted with a red circle. Below it, under 'General Settings', there are fields for 'Data Type Name' (HS\_Node) and 'Data Source Name' (HS\_TRAIN), both marked with asterisks indicating they are required. A checked checkbox labeled 'Enabled' is also present.

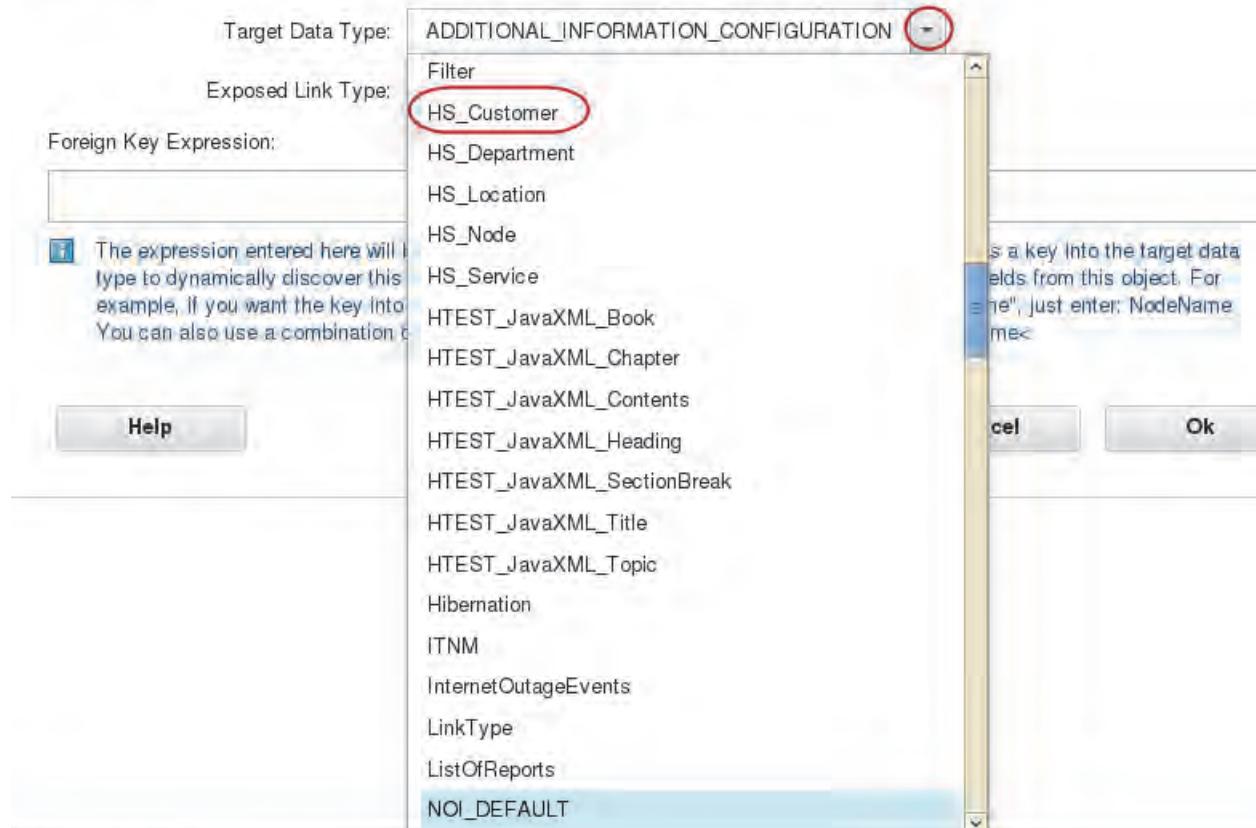
3. Scroll to the **Link By Key** section. Click **New** to create the first entry in the table.

The screenshot shows the 'SQL Data Type Config Page' for 'HS\_Node'. The 'Dynamic Links' tab is selected. In the 'Link By Key' section, a 'New' button is circled with a red circle. A vertical scroll bar on the right side of the page is also circled with a red arrow pointing downwards.

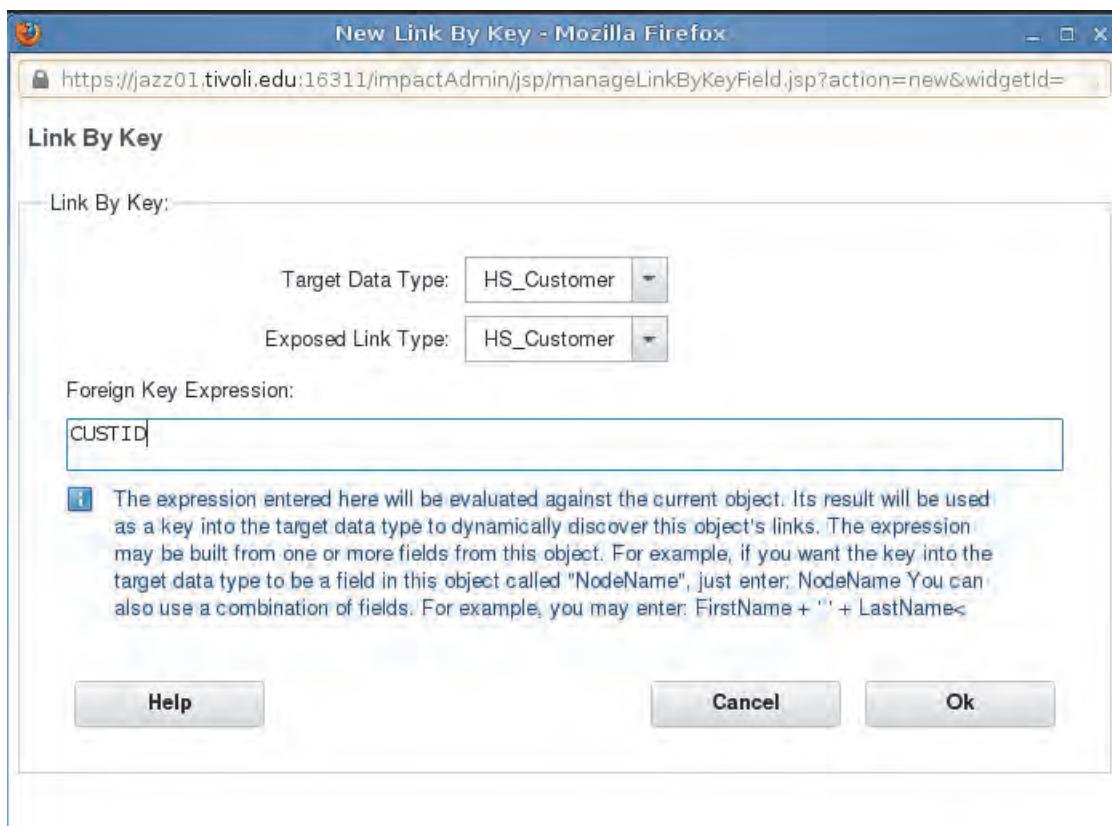
4. Select **HS\_Customer** in the **Target Data Type** menu.

**Link By Key**

—Link By Key:



5. Type **CUSTID** in the **Foreign key expression** field. Click **OK** to save the link description.

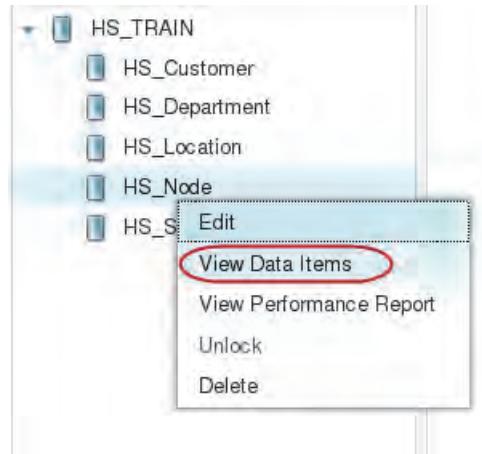


6. Click the **Save** icon.

The screenshot shows the 'SQL Data Type Config Page' for 'HS\_Node'. The 'Dynamic Links' tab is selected. A red arrow points to the 'Save' icon in the toolbar. The 'Link By Key' section has a tooltip explaining dynamic links. The 'Dynamic Links' table shows one entry:

Target Data Type	Value	Exposed Link Type
HS_Customer	CUSTID	HS_Customer

7. Verify that the link works by right-clicking the HS\_Node data type name and selecting **View Data Items**.



8. Click the magnifying glass icon next to the **gambit** entry to view the linked data item information.

A screenshot of the Netcool/Impact Data Model view. The top navigation bar includes Welcome, Data Model (selected), Policies, Services, Operator View, Event Isolation and Correlation. The left sidebar shows a tree view of data types: HS\_TRAIN, HS\_Customer, HS\_Department, HS\_Location, HS\_Node (selected), and HS\_Service. The main area displays the "Data Items: HS\_Node" view. It shows the "Data Type Name: HS\_Node" and "Number of Objects: 43". There is a "Filter Retrieved Data Items:" input field and a table below. The table has columns: Select, NODE, IP, NODEID, CUSTID, DEPTID, SERVID, and View Links. Three rows of data are shown:

Select	NODE	IP	NODEID	CUSTID	DEPTID	SERVID	View Links
<input type="checkbox"/>	gambit	10.10.10.10	0	1	1	1	
<input type="checkbox"/>	wombat	10.10.10.1	1	2	2	2	
<input type="checkbox"/>	orac	10.10.10.2	2	3	3	3	

The resulting screen should look like the following example.

The screenshot shows a software interface titled "Data Items: HS\_Node". On the left, there's a list of items: "gambit(HS\_Node)" and "BT(HS\_Customer)". The "BT(HS\_Customer)" item is highlighted with a red oval. On the right, there's a "Data Item Info (View Mode)" panel displaying the following data:

CUSTOMER	BT
CUSTOMERID	1
CONTACTNAME	Fred Bloggs
CONTACTNUMBER	152
SERVICELEVEL	95%
WORKINGHOURS	12:00-15:00
LOCID	1

9. Repeat the steps in this exercise for the remaining link table entries.

**Table 1 Link information**

Link Source Table	Link Target Table	Foreign Key
HS_Node	HS_Department	DEPTID
HS_Node	HS_Service	SERVID
HS_Customer	HS_Location	LOCID

The following graphic shows the resulting link definitions for **HS\_Node**.

Target Data Type	Value	Exposed Link Type
HS_Customer	CUSTID	HS_Customer
HS_Department	DEPTID	HS_Department
HS_Service	SERVID	HS_Service

10. Click the link icon to show the **HS\_Node** link data for the **gambit** entry.

Select	NODE	IP	NODEID	CUSTID	DEPTID	SERVID	View Links
<input type="checkbox"/>	gambit	10.10.10.10	0	1	1	1	
<input type="checkbox"/>	wombat	10.10.10.1	1	2	2	2	
<input type="checkbox"/>	orac	10.10.10.2	2	3	3	3	

The following graphic shows the resulting link for **HS\_Customer** after the dynamic link to **HS\_Location** has been created.

The screenshot shows the 'SQL Data Type Config Page' for the 'HS\_Customer' data type. The 'Dynamic Links' tab is selected. A note says: 'Use the Dynamic Links By Key table to manage dynamic links. A dynamic link where the relationship between two data types is specified by a foreign key expression.' Below is a table:

Target Data Type	Value	Exposed Link Type
HS_Location	LOCID	HS_Location

11. Display data for the **HS\_Node** data item. Click the link icon to show the **HS\_Customer** link data for the **gambit** entry. Expand **HS\_Customer** to see the link to the **HS\_Location** table.

The screenshot shows the 'Data Items: HS\_Node' page. Under 'View Links', it says: 'Click or Press Enter on a link to view its details.' A list of links is shown:

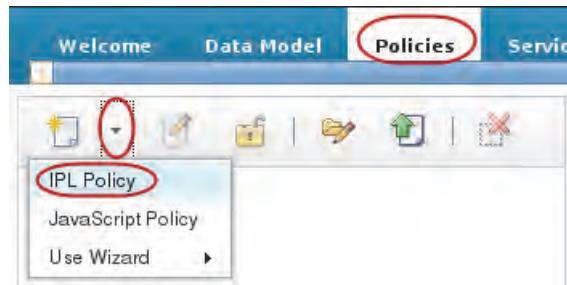
- gambit(HS\_Node)
- BT(HS\_Customer)
- 1(HS\_Location)
- Technical Services - US(HS\_Department)
- 1(HS\_Service)

# Unit 4 Policies exercises

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

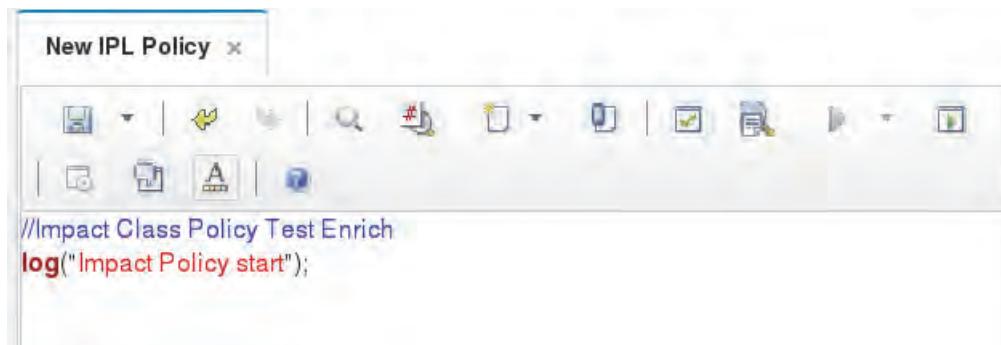
## Exercise 1 Creating a policy

1. Click **Policies** to open the **Policies** tab. Click the **New Policy** icon. Select **IPL Policy** in the list.



2. Modify the policy as shown in the screen capture.

```
log ("Impact Policy start");
```



```
//Impact Class Policy Test Enrich
log("Impact Policy start");
```

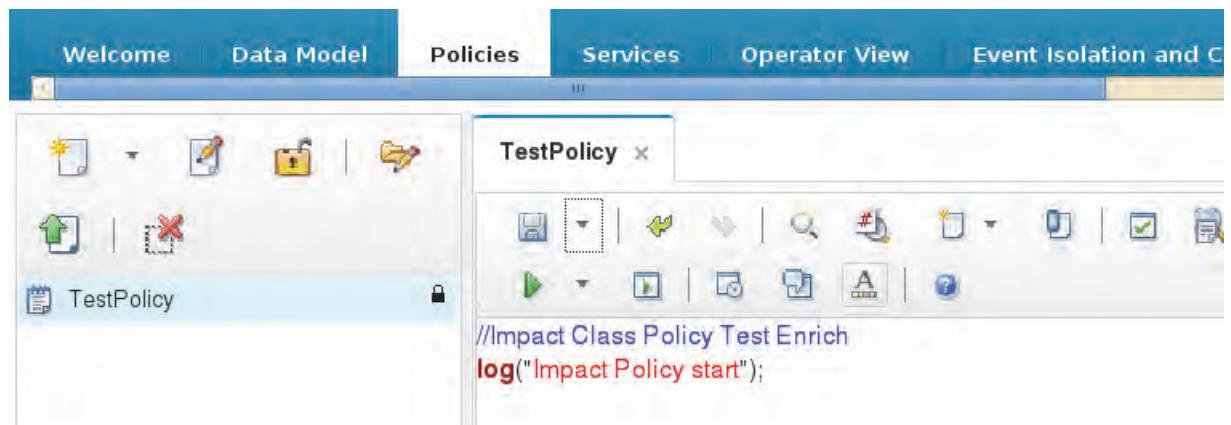
3. Click the arrow next to the **Save** icon and select **Save As** in the list.



4. Type the **TestPolicy** in the **Policy Name** field. Click **Save As**.



The new policy is displayed in the Policies panel.

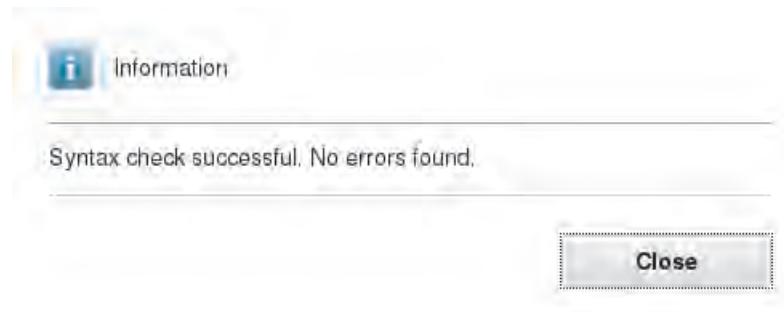


## Exercise 2 Creating a context policy

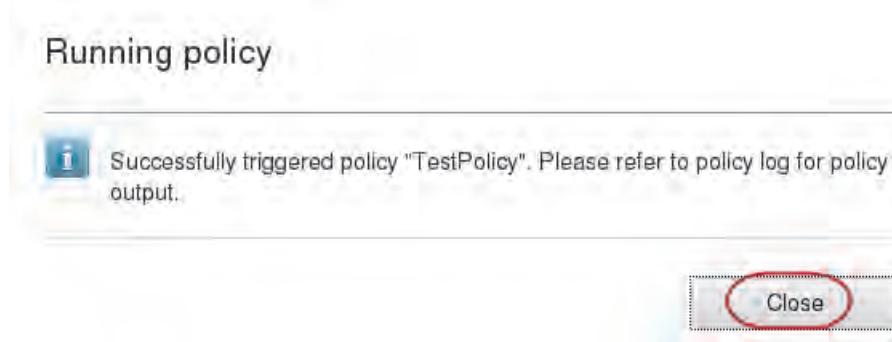
1. Edit the **TestPolicy** just created.
2. Type the following policy into the policy editor after the log statement:

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

3. Click the **Check Syntax** icon to check for policy errors.
4. Click **OK** to close the window.

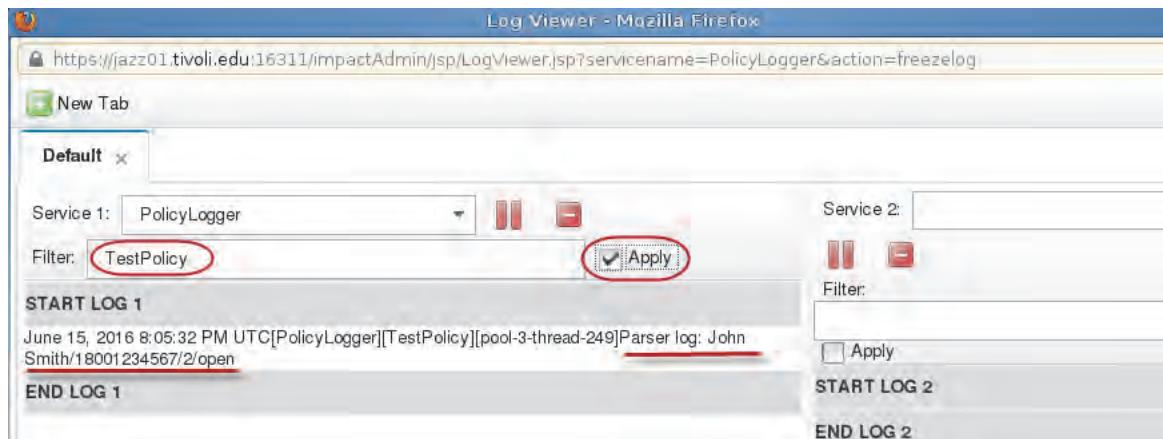


5. Click the **Save** icon .
6. Click the right-pointing arrow to test the policy. Click **OK** to close the window.



7. Click the **View Policy Log** icon to view the policy results.

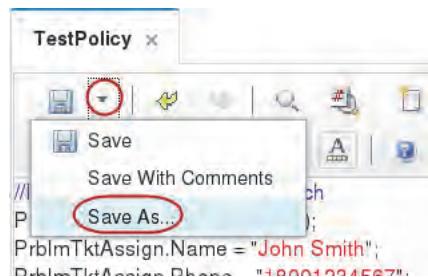
8. Create a Log filter.
  - Type **TestPolicy** in the **Filter** field.
  - Check **Apply** to isolate the results.
  - Click the minus sign “-” in the red box to clear the screen.
  - Leave the **Log Viewer** window open.
  - Execute **TestPolicy** one more time from the **Policy** panel.
  - The resulting **Log Viewer** output should look similar to the following example.



**Important:** You have to set the filter before you run the policy.

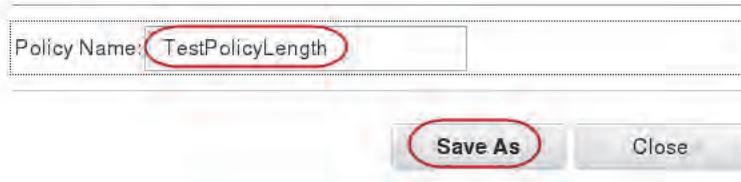
## Exercise 3 Using the length function

1. Open **TestPolicy** in the policy editor.
2. Click the arrow next to the **Save** icon and select **Save As** from the list.

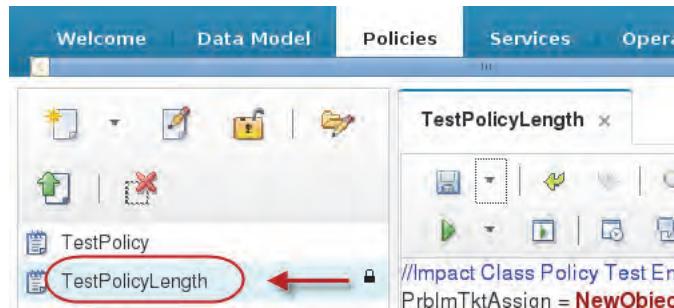


3. Type **TestPolicyLength** in the **Policy Name** field and click **OK**.

Save As



4. Verify that the new policy is displayed in Policies list as **TestPolicyLength**.



5. Modify the **TestPolicyLength** policy. Delete the existing lines of code, and add the following lines, also shown in the screen capture. Click the **Save** icon.

```
//Length Test
log("Impact Policy Start");
DataType="HS_Location";
Filter=("City='New York'");
CountOnly=False;
MyNodes=GetByFilter(DataType,Filter,CountOnly);
NumNodes=Length(MyNodes);
Log("Length Test Policy" + "," + NumNodes);
```

**Note:** The quotes display correctly when you type in the policy editor. Cutting and pasting from a pdf document may not provide the desired results.

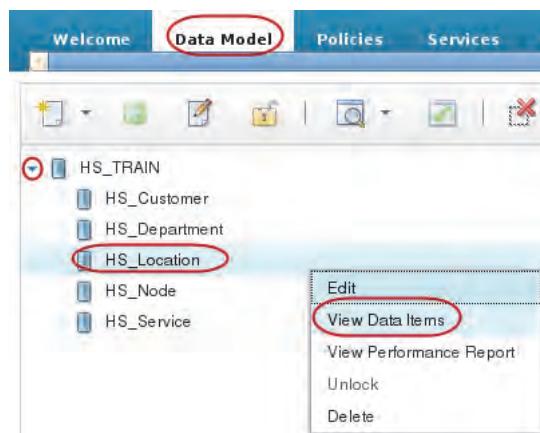
6. Click the icon to turn on syntax highlighting in the policy.



In the policy editor blue, green and red colors make it easier to follow the policy flow.

```
//Length Test
log("Impact Policy Start");
Data Type="HS_Location";
Filter="("City='New York')";
CountOnly=False;
MyNodes=GetByFilter(Data Type,Filter,CountOnly);
NumNodes=Length(MyNodes);
Log("Length Test Policy" + ":" + NumNodes);
```

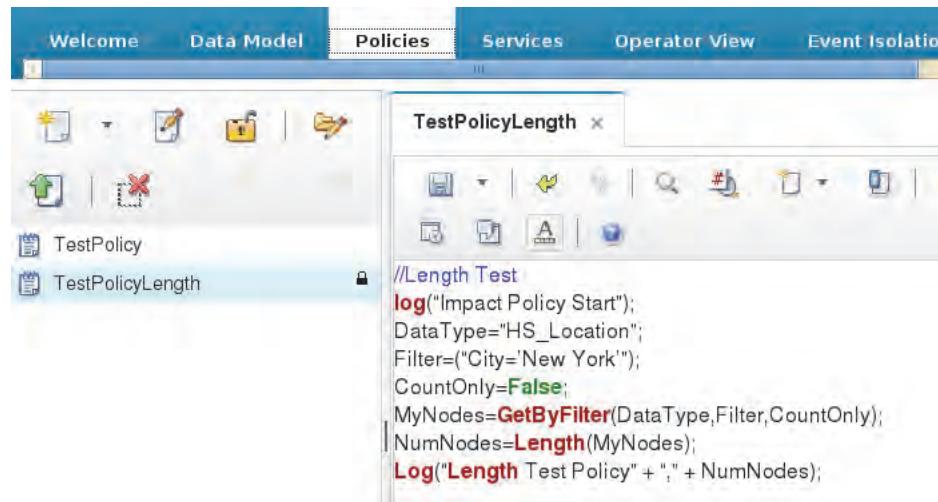
- Verify the filter used in the **GetByFilter** expression. Click **Data Model** to open the **Data Model** tab. Expand **HS\_TRAIN**. Right-click **HS\_Location** and select **View Data Items**.



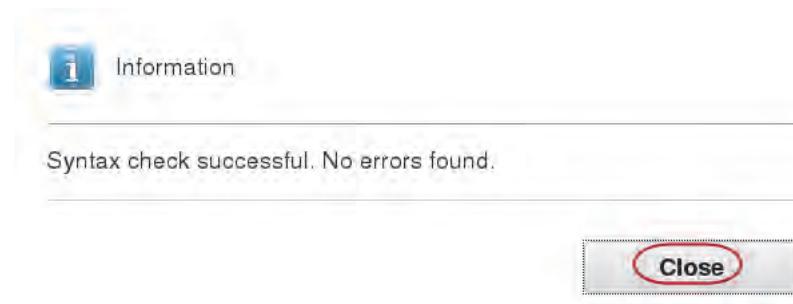
- Enter the **Filter** expression as shown in the screen capture.

LOCATIONID	LOCATION	CITY	COUNTRY	AREACODE
1	US:New York:Wall Street	New York	US	212

The resulting policy window looks similar to the screen capture.

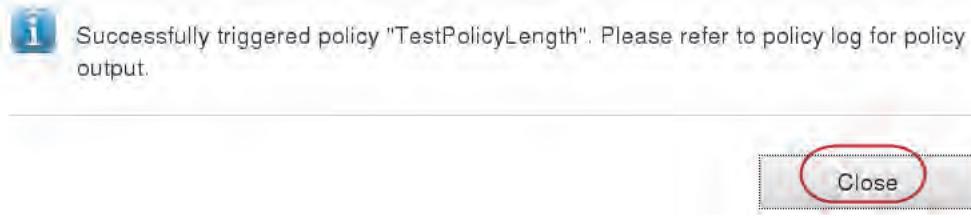


9. Click the **Check Syntax** icon to check for policy errors. Perform a syntax check and save the policy.



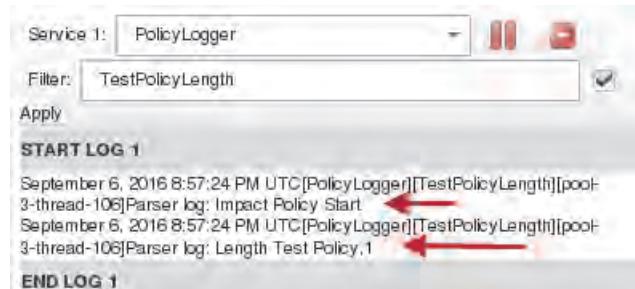
10. Click the right-pointing arrow to test the policy. Click **OK** to close the window.

### Running policy



11. Click the **Policy Log** icon to view the policy results.

**PolicyLogger** results for this policy look similar to the following screen capture.



12. Modify the **TestPolicyLength** policy to test a filter for the **Service** table.

Use the following information to build the filter expression:

**Data Type:** HS\_Service    **Filter:** Service='Network Management'

If the **HS\_Service** table returns data using the filter expression, you can be confident when you add the filter expression to the policy.

- Test the filter expression using the **Data Model** for the **HS\_Service** table.
- Modify the policy to add the filter.
- Run the policy, then view the log to see the results.
- The results should look similar to the following example.

The screenshot displays two main windows:

- Data Items: HS\_Service**: Shows a table with one object named "Network Management". A red circle highlights the "Filter" button, which is set to "Service='Network Management'".
- TestPolicyLength**: Shows the policy code with annotations:
  - //Length Test
  - log("Impact Policy Start");
  - DataType="HS\_Service"; (highlighted by a red arrow)
  - Filter=("Service='Network Management'"); (highlighted by a red arrow)
  - CountOnly=False;
  - MyNodes=GetByFilter(DataType,Filter,CountOnly);
  - NumNodes=Length(MyNodes);
  - Log("Length Test Policy" + "," + NumNodes);
- PolicyLogger** interface at the bottom right showing log entries for the policy run.

13. Modify the policy again to test a filter for the **HS\_Department** table. Run the policy, and view the log to see the results.

Use the following information to build the filter expression:

DataType: HS\_Department Filter:DeptID=1

The results should show as follows:

The screenshot shows the IBM Impact interface. At the top, there's a toolbar with various icons. Below it is a navigation bar with 'Data Items: HS\_Department' and a 'Filter' dropdown set to 'DeptID=1'. The main area displays a table with one row: 'Technical Services - US' with contact name 'Tim Heywood' and phone number '0181875-9500'. To the right of the table is a 'TestPolicyLength' editor window containing the following code:

```
//Length Test
log("Impact Policy Start");
DataType="HS_Department";
Filter="DeptID=1";
CountOnly=False;
MyNodes=GetByFilter(DataType,Filter,CountOnly);
NumNodes=Length(MyNodes);
Log("Length Test Policy" + " " + NumNodes);
```

Two red arrows point from the text 'DeptID=1' in the code to the 'Filter' field in the navigation bar above. To the right of the editor is a 'Default' panel with a service configuration and a log window titled 'START LOG 1' showing the execution details.

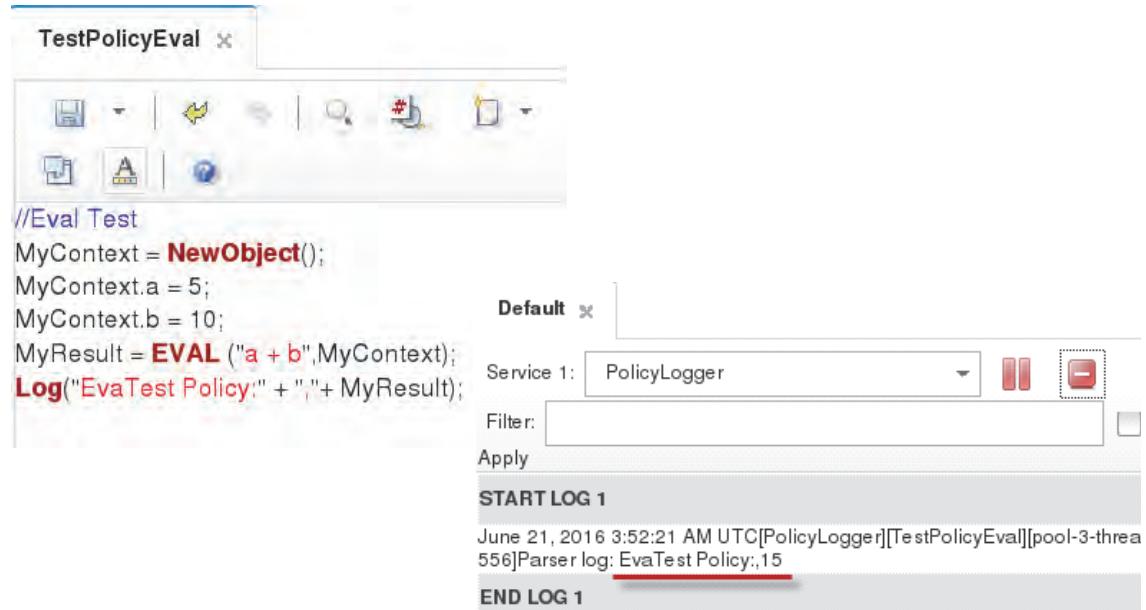
# Exercise 4 Using the Eval function

1. Create the **TestPolicyEval** policy using the steps described in the previous exercises. Enter the policy data as displayed.

```
//Eval Test
MyContext = NewObject();
MyContext.a = 5;
MyContext.b = 10;
MyResult = EVAL ("a + b",MyContext);
Log("EvaTest Policy:" + ","+ MyResult);
```

2. Test the **TestPolicyEval** policy and verify the results.

The result looks similar to the following screen capture.



# Exercise 5 Using the extract function

1. Create the **TestPolicyExtract** policy using the steps from the previous exercises. Enter the policy data as shown in the following screen capture.

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

2. Test the **TestPolicyExtract** policy and verify the results.

The result looks similar to the following screen capture.

The screenshot shows the Policy Studio interface with the following details:

- Title Bar:** TestPolicyExtract
- Toolbar:** Includes icons for Save, Undo, Redo, Find, Filter, and Log.
- Policy Editor:** Displays the C# code for the policy:

```
//Extract Test
MyString = "This is a test";
MyWord=Extract(MyString, 1, " ");
Log(MyWord);
MyString = "This|is|a|test.";
MyWord = Extract (MyString,3,"|");
Log(MyWord);
```
- Default Log View:** Shows the configuration for the log:
  - Service 1: PolicyLogger
  - Filter: (empty)
  - Buttons: Stop, Start, Refresh, Apply
- Log Output:** Displays the log entries:

```
START LOG 1
June 21, 2016 4:14:07 AM UTC[PolicyLogger][TestPolicyExtract][pool-3-thread-569]Parser log: is
June 21, 2016 4:14:07 AM UTC[PolicyLogger][TestPolicyExtract][pool-3-thread-569]Parser log: test.
END LOG 1
```



---

# **Unit 5 Services exercises**

This unit has no student exercises.



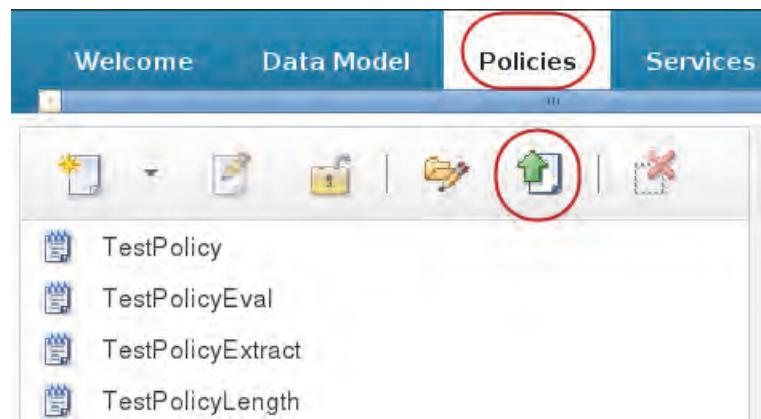
# Unit 6 The Enrichment policy exercises

This unit focuses on the Enrichment Policy.

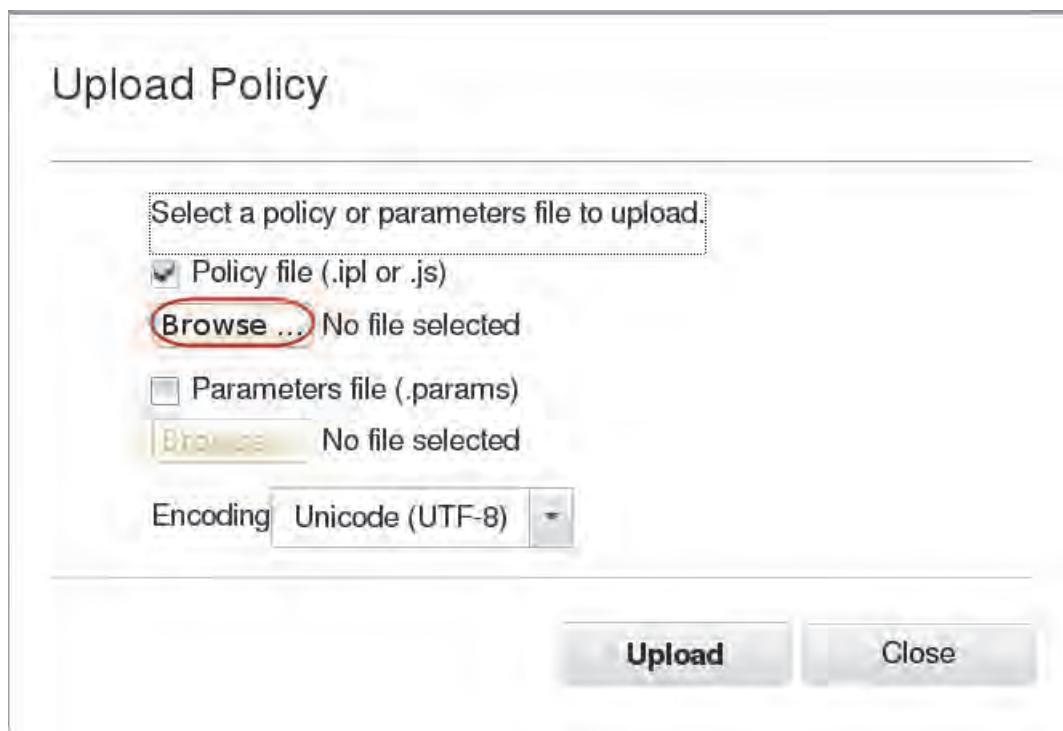
## Exercise 1 Importing a policy with the policy uploader

In this exercise, you use the policy uploader to import a policy. The policy illustrates basic event enrichment by using the Impact Programming Language functions **GetByLinks** and **GetByKey**.

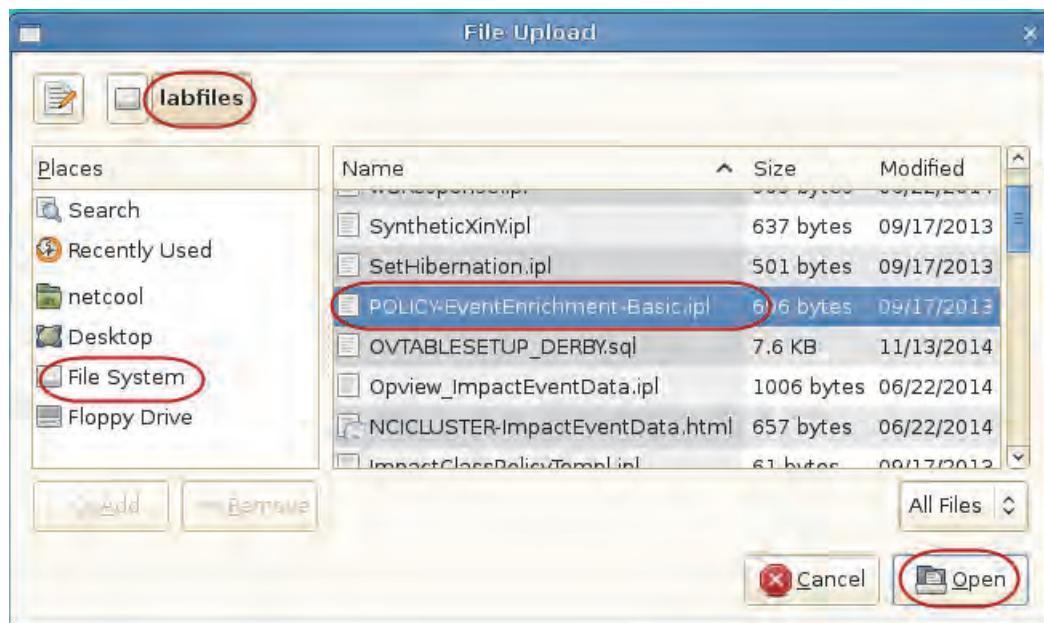
1. In the navigation tree, click **Policies** to open the **Policies** tab.
2. Click the **Upload a Policy File** icon.



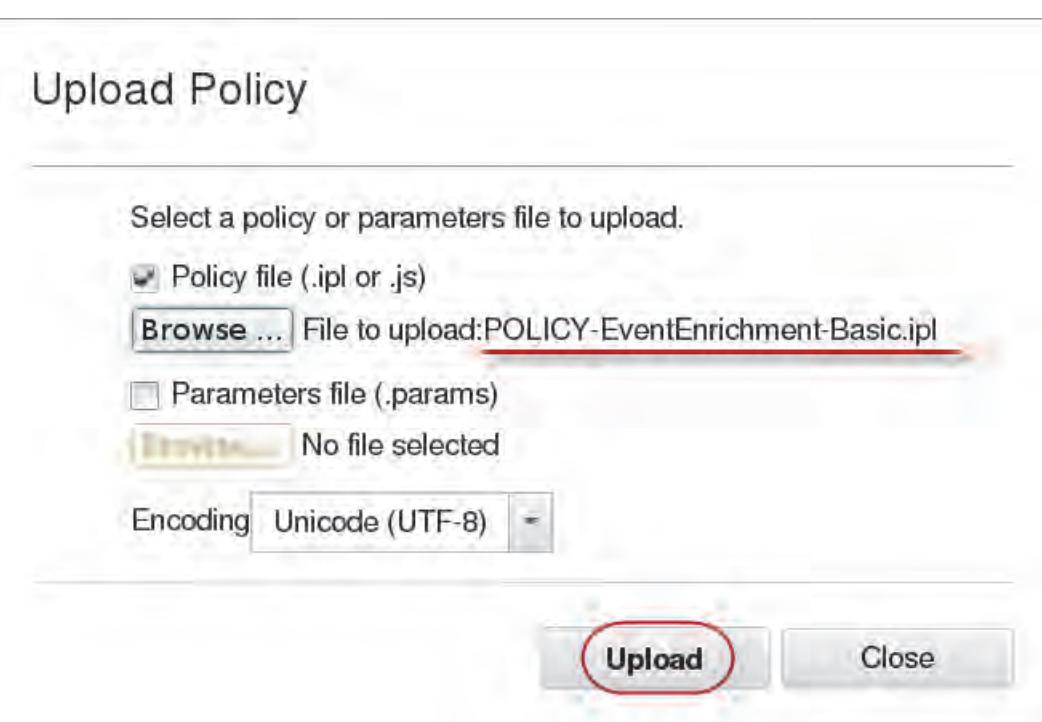
3. Click **Browse** to navigate to the **/labfiles** directory.



4. Highlight the **POLICY-Event\_Enrichment-Basic.ipl** policy. Click **Open** to load the policy and close the window.



5. Click **Upload** to import the selected policy.



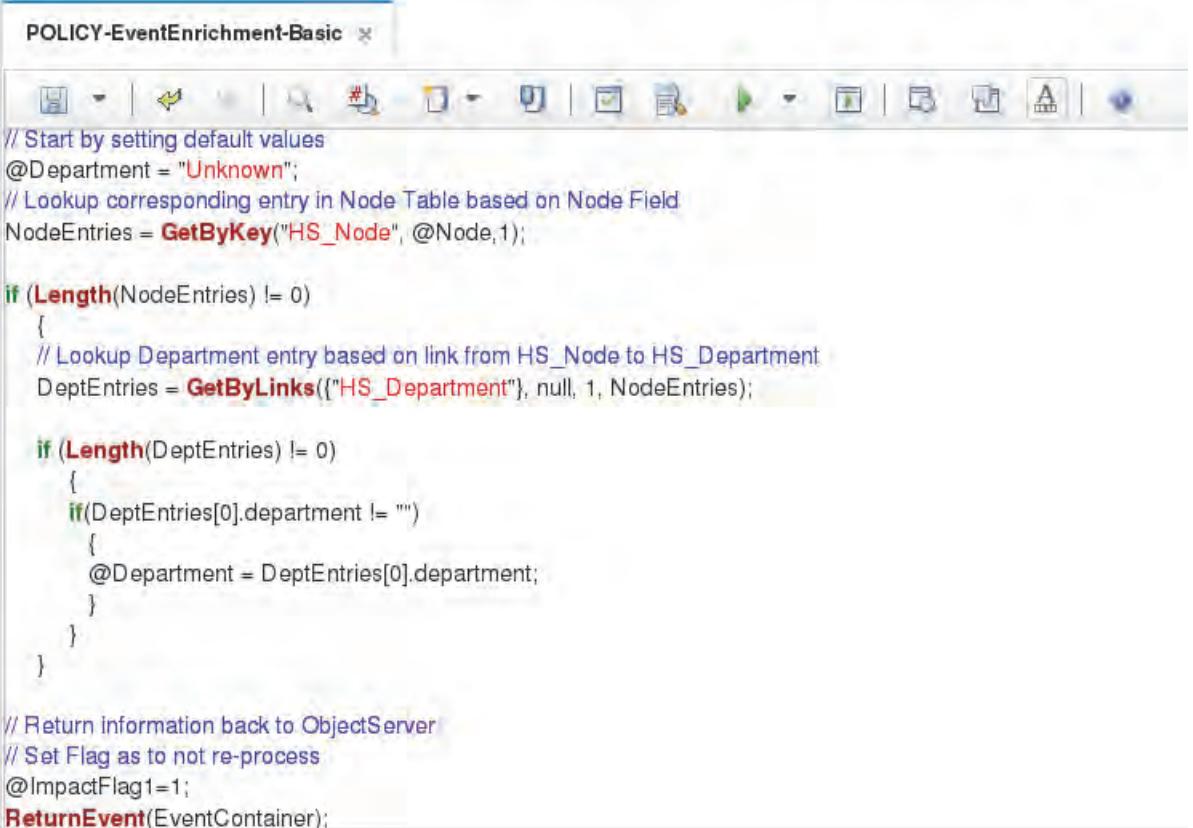
The policy is added to the **Policies** list as shown in the following display.

The screenshot shows a software interface with a blue header bar containing tabs for 'Welcome', 'Data Model', 'Policies' (which is highlighted), and 'Services'. Below the header is a toolbar with several icons. The main area is a list titled 'Policies' with the following items:

Policy Name
POLICY-EventEnrichment-Basic
TestPolicy
TestPolicyEval
TestPolicyExtract
TestPolicyLength

A red arrow points to the 'POLICY-EventEnrichment-Basic' entry in the list.

6. Right-click the **POLICY-EventEnrichment-Basic** policy. Select **EDIT** from the list to view the policy contents. Click **Save**.



```
// 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;
        }
    }
}

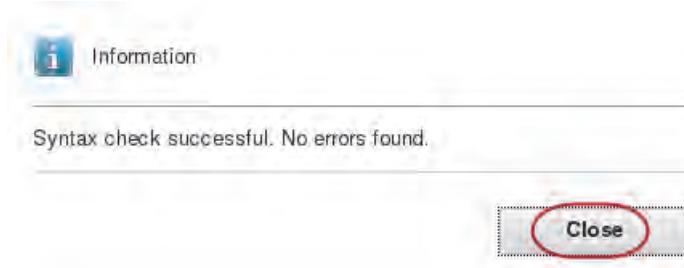
// Return information back to ObjectServer
// Set Flag as to not re-process
@ImpactFlag1=1;
ReturnEvent(EventContainer);
```

7. Determine the best location for each log statement. Edit the policy **POLICY-EventEnrichment-Basic** and add each of the following log statements.

```
// Start by setting default values
log("****Policy Start****");
log("Serial from the Event Container=====>"+@Serial);
log("Department set in the policy=====>"+@Department);
log("Node from the Event Container=====> "+@Node);
log("NodeEntries: Result of Lookup in HS_Node=====> "+NodeEntries);
log("Dept Entries: Result of lookup in HS_Department =====>"+DeptEntries);
log("Department Output to be sent to OMNIbus in the
EventContainer=====>"+@Department);
```

8. Click the check syntax icon to verify there are no errors. 

9. Click **OK** to close the window. Save the policy.



The resulting policy should look similar to the following display.

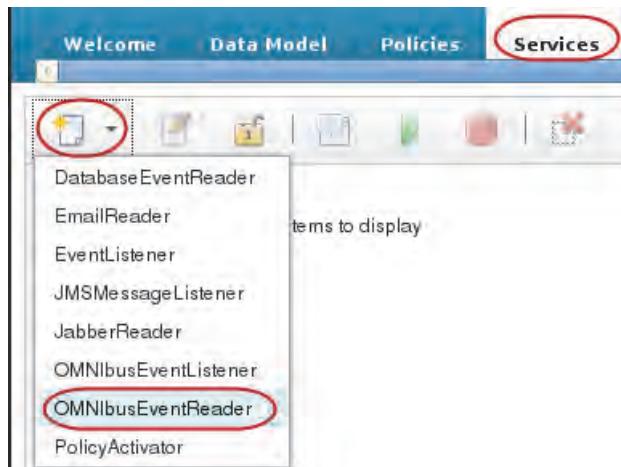
The screenshot shows the IBM Policy Studio interface with the title bar 'POLICY-EventEnrichment-Basic'. The main area displays the following C#-like policy code:

```
// 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);
    if (Length(DeptEntries) != 0)
    {
        if (DeptEntries[0].department != "")
        {
            @Department = DeptEntries[0].department;
        }
    }
}
log("Department Output to be sent to OMNIbus in the EventContainer=====>" + @Department);
// Return information back to ObjectServer
// Set Flag as to not re-process
@ImpactFlag1=1;
ReturnEvent(EventContainer);
```

# Exercise 2 Configuring event mapping in the OMNIbus Event Reader service

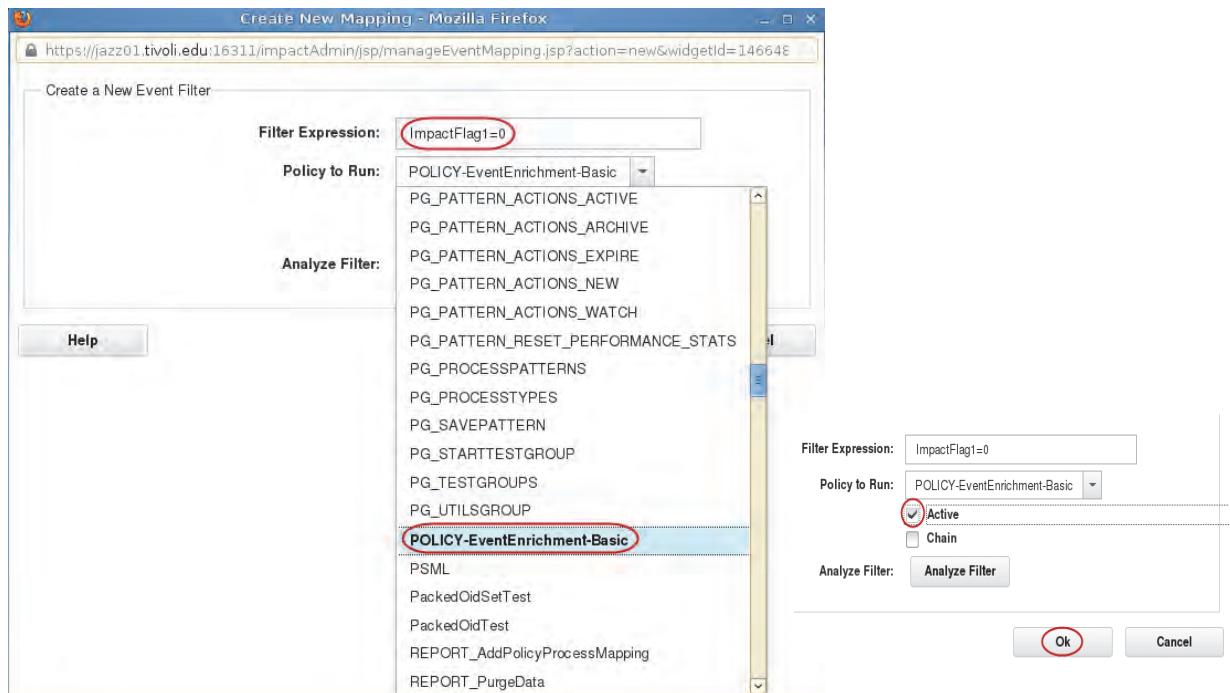
1. Click Services to open the **Services** tab. Click the **Create New Service** icon. Select **OMNIbusEventReader** in the list.



2. Click the **Event Mapping** tab. Scroll down to the **Event Mapping Table**. Click **New** to create a new mapping for the policy.

Restriction Filter	Policy Name

3. In the **Edit Event Filter** window, perform the following steps:
  - a. In the **Filter Expression** field, type **ImpactFlag1=0**.
  - b. Select **POLICY-EventEnrichment-Basic** from the **Policy to Run** list.
  - c. Select the **Active** check box. Click **OK** to close the window.



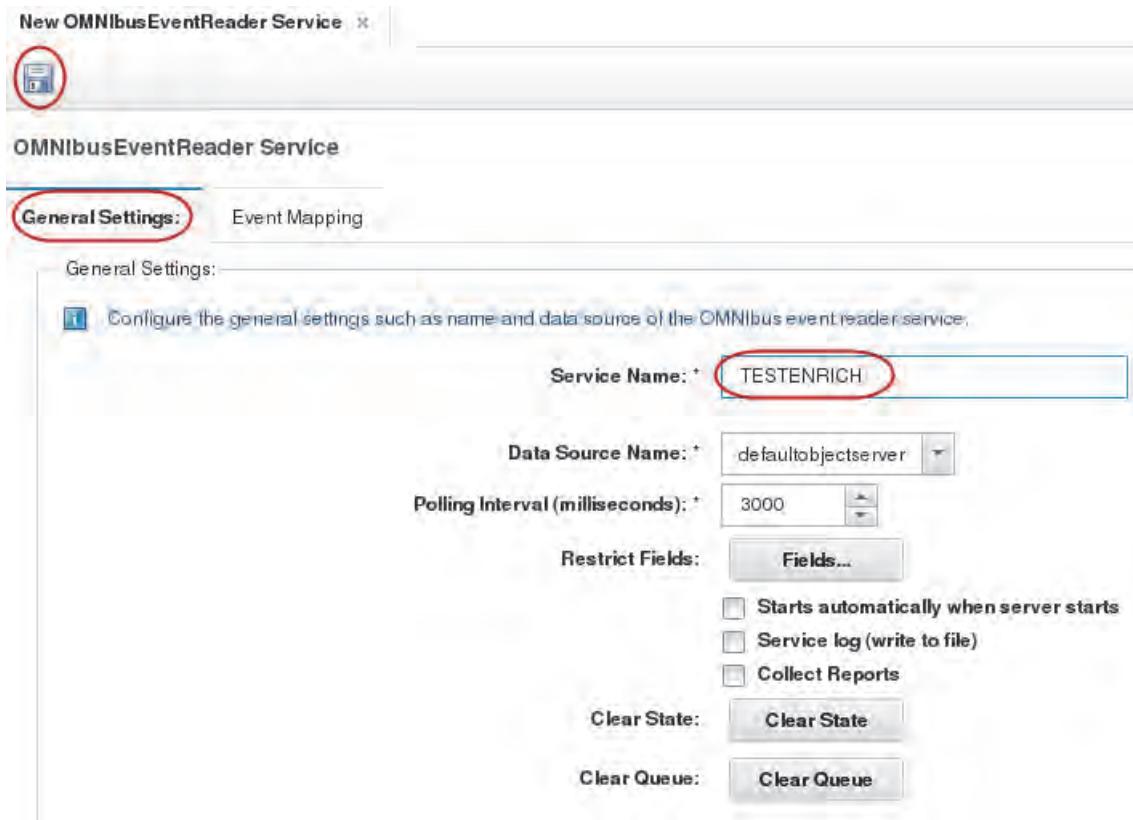
The policy shows in the mapping table as shown in the following screen capture. When this service starts, the **POLICY-EventEnrichment-Basic** policy runs. The filter selects events to pass to the policy that have a zero value in the alerts.status field; **ImpactFlag1=0**.

The screenshot shows the 'Event Mapping' tab of the OMNIbusEventReader Service configuration. The table has two columns: 'Restriction Filter' and 'Policy Name'. A single row is present, representing the filter 'ImpactFlag1=0' which maps to the policy 'POLICY-EventEnrichment-Basic'. The policy is marked as 'Active'.

Restriction Filter	Policy Name	Active
ImpactFlag1=0	POLICY-EventEnrichment-Basic	Yes

Below the table, there is a 'Quick Add' section where a new row can be added. The columns in this section are 'e', 'Active', and 'Chain'. The first column has the value 'enrichment-Basic', the second column has 'Yes', and the third column has 'No'.

4. Click the **General Settings** tab. Type **TESTENRICH** as the service name. Click the **Save** icon.



The new service shows in the **Services** list as shown in the following screen capture.



- Double-click the GNOME Terminal icon on the desktop.



- To generate OMNIbus events, start the *Simnet Probe* by typing the following commands from the command line as shown:

```
cd /opt/IBM/tivoli/netcool/omnibus/probes  
.nco_p_simnet &
```

A screenshot of a terminal window titled "netcool@jazz01:~>". The window shows the command "cd /opt/IBM/tivoli/netcool/omnibus/probes" being entered. Below it, the command ".nco\_p\_simnet &" is shown, with the "&" part circled in red. The output shows "[1] 13249" followed by the prompt "netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/probes>".

- Start the **TESTENRICH** service by clicking the **Start Service** icon.



The check mark indicates the service is running.



8. Click the icon to view the Service Log . Verify messages in the Service Log output.

A screenshot of the Service Log viewer. The title bar says 'Default'. Below it, 'Service 1:' is set to 'TESTENRICH'. There are 'Filter:' and 'Apply' buttons. The main area is a scrollable log window titled 'START LOG 1'. It contains the following log entries:

```
June 21, 2016 5:18:44 AM UTC[TESTENRICH]Starting the service TESTENRICH
June 21, 2016 5:18:44 AM UTC[TESTENRICH]Registering TESTENRICH as a source of events in the QueueManager
June 21, 2016 5:18:44 AM UTC[TESTENRICH]Initializing
June 21, 2016 5:18:44 AM UTC[TESTENRICH]Initializing the Event Feed Connection..
June 21, 2016 5:18:44 AM UTC[TESTENRICH]Connected to the Database
June 21, 2016 5:18:44 AM UTC[TESTENRICH]Connected to EventFeed.
June 21, 2016 5:18:44 AM UTC[TESTENRICH]Connected to Event Source
June 21, 2016 5:18:44 AM UTC[TESTENRICH]Running.
```

# Exercise 3 Verifying the results of the POLICY-Event\_Enrichment-Basic policy

1. Edit the **POLICY-Event\_Enrichment-Basic** policy window. Click the **Policy Log** icon to view results of the log statements.



```
[MessageProcessor-Dog#3]Parser log: ***Policy Start***  

[MessageProcessor-Dog#3]Parser log: Serial from the Event Container=====>1991  

[MessageProcessor-Dog#3]Parser log: Department set in the policy=====>Unknown  

[MessageProcessor-Dog#3]Parser log: Node from the Event Container=====> jazz01.tivoli.edu  

[MessageProcessor-Dog#1]Parser log: ***Policy Start***  

[MessageProcessor-Dog#1]Parser log: Serial from the Event Container=====>1992  

[MessageProcessor-Dog#1]Parser log: Department set in the policy=====>Unknown  

[MessageProcessor-Dog#1]Parser log: Node from the Event Container=====> jazz01.tivoli.edu  

[MessageProcessor-Dog#1]Parser log: NodeEntries: Result of Lookup in HS_Node=====> {}  

[MessageProcessor-Dog#3]Parser log: NodeEntries: Result of Lookup in HS_Node=====> {}  

[MessageProcessor-Dog#3]Parser log: Department Output to be sent to OMNIbus in the EventContainer=====>Unknown  

[MessageProcessor-Dog#3]Parser log: Department Output to be sent to OMNIbus in the EventContainer=====>Unknown  

[MessageProcessor-Dog#2]Parser log: **Policy Start**  

[MessageProcessor-Dog#2]Parser log: Serial from the Event Container=====>1993  

[MessageProcessor-Dog#2]Parser log: Department set in the policy=====>Unknown  

[MessageProcessor-Dog#2]Parser log: Node from the Event Container=====> link6  

[MessageProcessor-Dog#2]Parser log: NodeEntries: Result of Lookup in HS_Node=====> {link6}  

[MessageProcessor-Dog#2]Parser log: Dept Entries: Result of lookup in HS_Department =====>{Sales}  

[MessageProcessor-Dog#2]Parser log: Department Output to be sent to OMNIbus in the EventContainer=====>Sales
```

2. Double-click the **GNOME Terminal** icon on the desktop.



3. Run the script to verify the updates to the OMNIbus **alerts.status** table using the following commands:

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

./nco_sql -server NCOMS -user root -password object00 -i /labfiles/OMNIbusQuery
```

The **Node** and **Department** entries in OMNIbus should look similar to the following display.

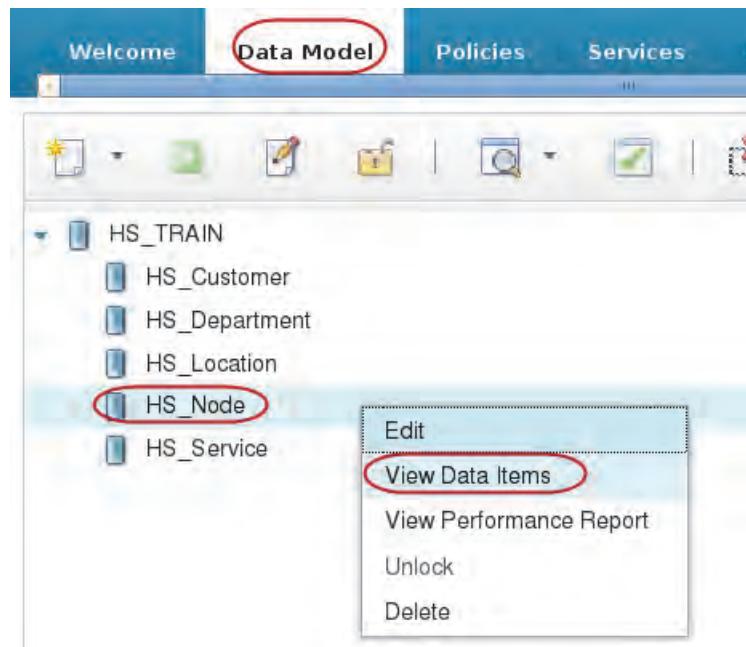


**Note:** The number of rows affected varies depending on how long the probe has been running.

```
netcool@jazz01:~> cd /opt/IBM/tivoli/netcool/omnibus/bin
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ./nco_sql -server
NCOMS -user root -password object00 -i /labfiles/OMNIbusQuery
Node
    Department
    -----
    link6
        Sales
    link5
        Technical Services - EMEA
    link1
        Training Department

(3 rows affected)
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin>
```

4. Compare the results to the table data. Click the **Data Model** tab. Right-click **HS\_Node** and select **View Data Items**.

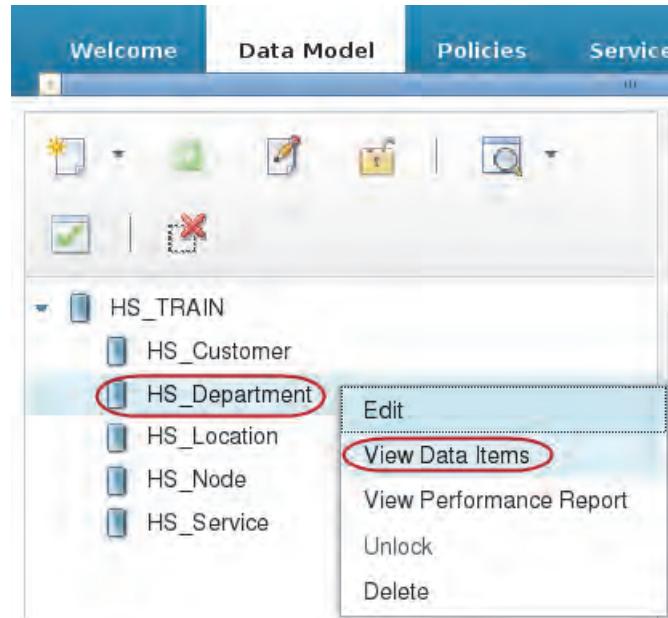


The **HS\_Node** entries are shown similar to the display.

ID	Name	IP Address	5	6	7	8	9	10	Actions
	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			
	angel	10.10.10.8	8	0	9	2			
	link1	10.10.10.9	9	0	10	0			
	link2	10.10.10.10	10	9	0	4			
	link3	10.10.10.11	11	10	0	5			
	link4	10.10.10.12	12	1	1	0			
	link5	10.10.10.13	13	2	2	7			
	link5	10.10.10.14	14	3	3	1			
	link6	10.10.10.15	15	4	4	2			
	batman	10.10.10.16	16	4	4	0			

Recall the links between tables created earlier. The **HS\_Node** table uses **DEPTID** to link to the **HS\_Department** table.

5. Compare the results to the table data. Click the **Data Model** tab. Right-click **HS\_Department** and select **View Data Items**.



The **HS\_Department** entries are shown similar to the display.

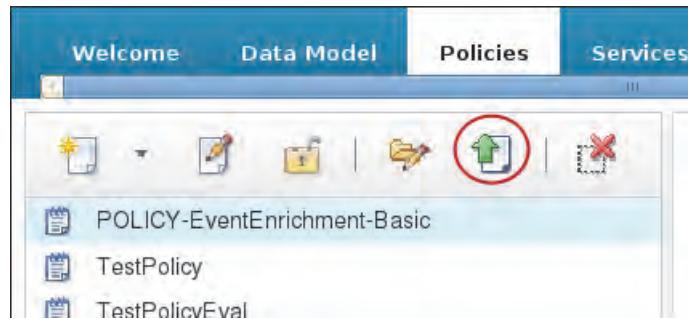
**Data Items: HS\_Department**

Select	DEPARTMENT	DEPTID	CONTACTNAME	CONTACTNUMBER	View Links	Edit
<input type="checkbox"/>	Technical Services - US	1	Tim Heywood	0181875-9500		
<input type="checkbox"/>	Technical Services - EMEA	2	Laurent Marchal	0181875-9500		
<input type="checkbox"/>	Finance	3	Anthony Isaacs	0181875-9500		
<input type="checkbox"/>	Sales	4	Mike Donohue	0181875-9500		
<input type="checkbox"/>	Marketing	5	Evan Birkhead	0181875-9500		
<input type="checkbox"/>	Technical Support	6	Mike Foster	0181875-9500		
<input type="checkbox"/>	Development	7	Adam Kerrison	0181875-9500		
<input type="checkbox"/>	Maintenance	8	Jim Bleecker	0181875-9500		
<input type="checkbox"/>	Product Management	9	Peter Shearan	0181875-9500		
<input type="checkbox"/>	Training Department	10	Sarah Cambridge	0181875-9500		

# Exercise 4 Adding functions to the enhanced event enrichment policy

In this exercise, you add parsing and enrichment functions to the Event Enrichment policy created in the previous exercise. You populate the additional OMNIbus alert.status fields **Location**, **SLA**, **WorkingHours**, and **Customer**. You implement **Severity** and **Event Suppression**.

1. Click the **Upload a Policy File** icon.



2. Click **Browse** to navigate to the **/labfiles** directory.

Upload Policy

Select a policy or parameters file to upload

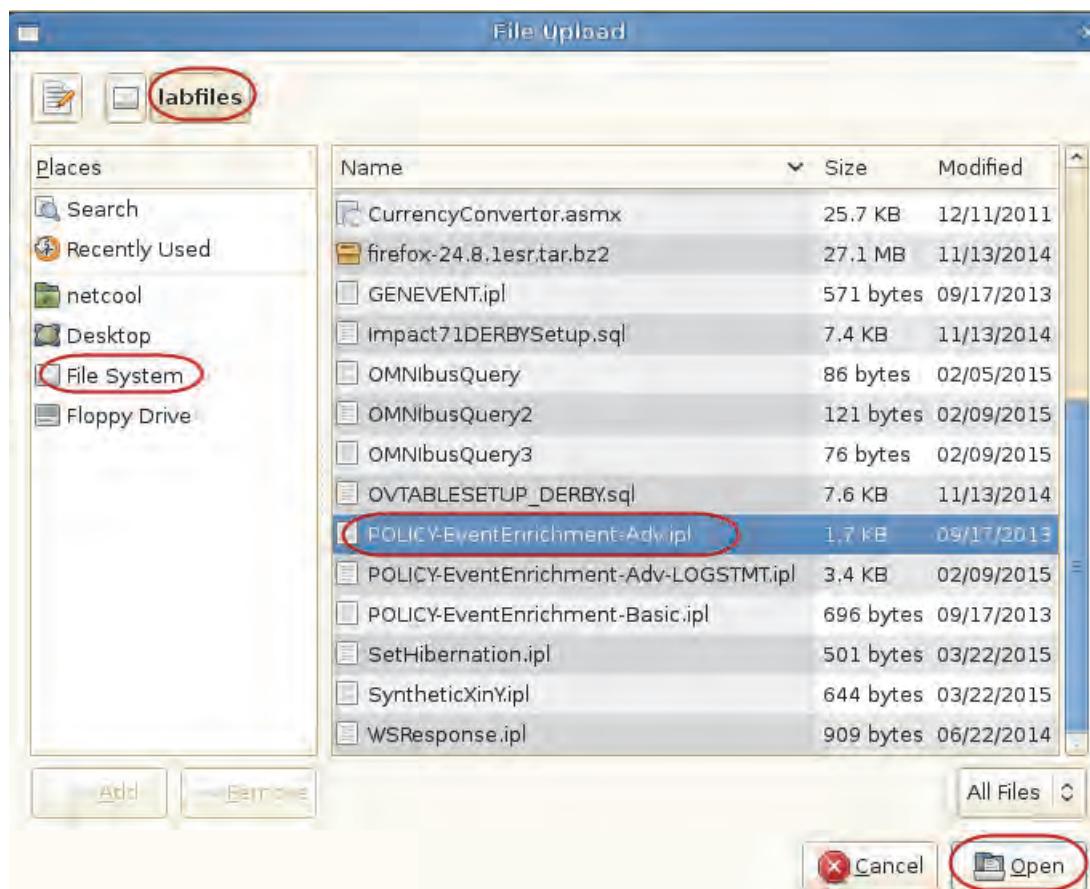
Policy file (.ipl or .js)  
**Browse...** | No file selected

Parameters file (.params)  
No file selected

Encoding: Unicode (UTF-8)

**Upload**    **Close**

3. Click File System>labfiles and select the **POLICY-EventEnrichment-Adv** policy. Click Open.



4. Click **Upload** to import the policy.

## Upload Policy

Select a policy or parameters file to upload.

Policy file (.ipl or .js)

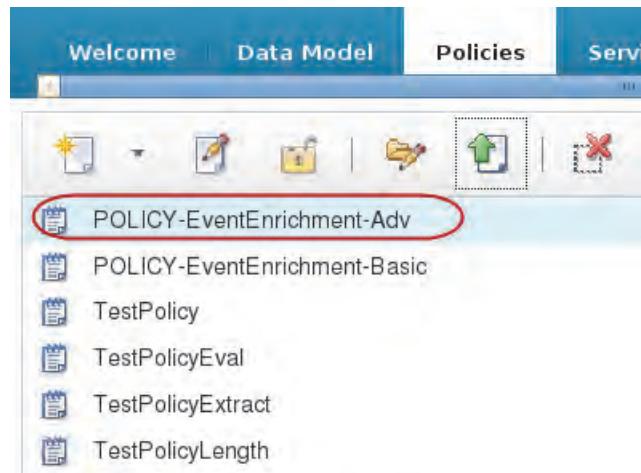
No file selected

Parameters file (.params)

No file selected

Encoding

The policy is added to the **Policies** list as shown in the following screen capture.



5. Modify the policy and add **Log** statements to verify the policy flow. See the examples in the following screen capture.



**Note:** Pay close attention the bracket ({} ) placements. Be sure to check syntax.

- a. Log statements: Policy Start, Input Key, GetByKey HS\_Node

```
POLICY-EventEnrichment-Adv

Log("**** Event Enrichment Adv Policy Start ****");
//Set policy defaults
@Department="Unknown";
@Location = "Unknown";
@SLA = "Unknown";
@WorkingHours = "Unknown";
@Customer = "Unknown";

//Input key from Event Container: Node
Log("== Display Event Container Search Key from OMNIbus ===");
Log("== Node ==>" + @Node);

//Get By Key HS_Node
NodeEntries = GetByKey("HS_Node", @Node, 1);
Log("== Result from the HS_Node table ==>" + NodeEntries);
```

Three red arrows point to specific Log statements in the policy logic:

- An arrow points to the first Log statement: `Log("**** Event Enrichment Adv Policy Start ****");`
- An arrow points to the second Log statement: `Log("== Display Event Container Search Key from OMNIbus ===");`
- An arrow points to the third Log statement: `Log("== Node ==>" + @Node);`

Figure 1 Policy logic screen 1

b. Log statements: GetByLinks HS\_Department and HS\_Customer, Process Department

```
//Get By Links
if (Length(NodeEntries) > 0) {
    @Service = NodeEntries[0].Service;
    Log("==> @Service value set from HS_Node ==>" + @Service); ←
    DeptEntries = GetByLinks({"HS_Department"}, null, 1, NodeEntries);
    Log("==> Result from HS_Department search ==>" + DeptEntries); ←
    CustomerEntries = GetByLinks({"HS_Customer"}, null, 1, NodeEntries);
    Log("==> Result from HS_Customer search ==>" + CustomerEntries); ←
}

//Process Department
if (Length(DeptEntries) > 0) {
    if (DeptEntries[0].department != "") {
        @Department = DeptEntries[0].department;
        Log("==> @Department value set from link ==>" + @Department); ←
    }
}
```

Figure 2 Policy logic screen 2

c. Log Statements: Process Customer

```
//Process Customer
if (Length(CustomerEntries) > 0) {
    @Customer = CustomerEntries[0].customer;
    Log("==> @Customer value set from link ==>" + @Customer); ←
    @SLA = CustomerEntries[0].servicelevel;
    Log("==> @SLA value set from link ==>" + @SLA); ←
    SLAValue=extract(@SLA, "[0-9]+\%");
    Log("==> SLAValue from extract ==>" + SLAValue); ←
    if (SLAValue != "") {
        if (int(SLAValue) <= 95) {
            @Severity=4;
        }
        else {
            @Severity=5;
        }
        Log("==> @Severity value calculated ==>" + @Severity); ←
    }
}
```

Figure 3 Policy logic screen 3

## d. Log statements: Working Hours Calculation

```

//Set working hours only if CustomerEntries is grater than 0
@WorkingHours = CustomerEntries[0].workinghours;
Log("==> @WorkingHours set from HS_Customer ==>" + @WorkingHours); ←
If(@WorkingHours != "24 Hours") {
    WHTime = @WorkingHours;
    Log("==> WHTime value calculated ==>" + WHTime); ←
    StartMinute = int(retract(WHTime,".*:(0-9)+:-"));
    Log("==> StartMinute ==>" + StartMinute); ←
    StartTime = int(retract(WHTime,"([0-9]+):*-"))*60 + StartMinute;
    Log("==> StartTime value calculated ==>" + StartTime); ←
    EndMinute = int(retract(WHTime,"-.*:(0-9)+"));
    Log("==> EndMinute value calculated ==>" + EndMinute); ←
    EndTime = int(retract(WHTime,"-([0-9]+):")*60 + EndMinute);
    CurrentMinute = int(localtime(@LastOccurrence,"mm"));
    Log("==> CurrentMinute value calculated ==>" + CurrentMinute); ←
    CurrentTime = int(localtime(@LastOccurrence,"HH"))*60 + CurrentMinute;
    Log("==> CurrentTime value calculated ==>" + CurrentTime); ←
If((CurrentTime < StartTime) or (CurrentTime > EndTime)) {
    @SuppressEscl = 4;
}
Log("==> @SuppressEscl value set ==>" + @SuppressEscl); ←
}

```

Figure 4 Policy logic screen 4

## e. Log statements: GetByLinks HS\_Location, set ImpactFlag2, Return Event

```

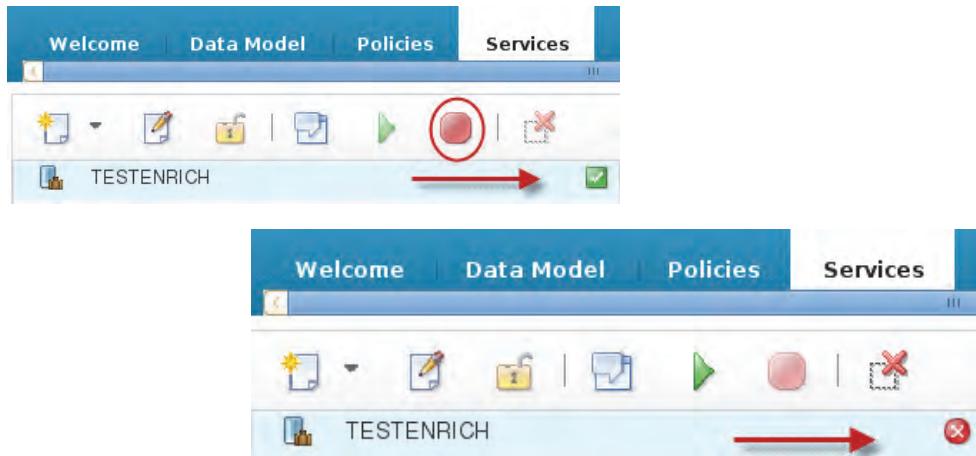
//Set Location only if CustomerEntries is greater than 0
//Get By Links
LocationEntries = GetByLinks({"HS_Location"}, null, 1, CustomerEntries);
Log("==> LocationEntries from HS_Location ==>" + LocationEntries); ←
If(Length(LocationEntries) > 0) {
    temploc= LocationEntries[0].location;
    Log("==> temploc value calculated ==>" + temploc); ←
    location=retract(temploc,".*:(.)*" + " " + retract(temploc,".*:(.)*$"));
    @Location = location;
    Log("==> @Location value calculated ==>" + @Location); ←
}
//end bracket below for CustomerEntries check
}
//Set Impact reprocess flag
@ImpactFlag2 = 1;
ReturnEvent(EventContainer);

```

Figure 5 Policy logic screen 5

# Exercise 5 Testing the enhanced enrichment policy

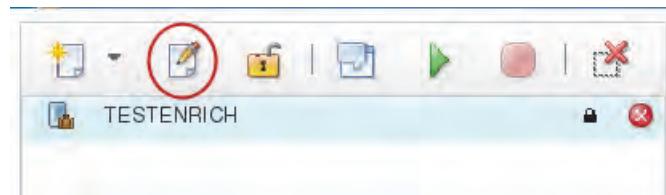
1. Stop the **TESTENRICH** service by clicking the red **Stop Service** button. The green check box changes to a red circle with an x when the server stops.



2. Click the Service Log icon to verify the server has stopped.   
The shutdown messages should look similar to the following display.

```
June 21, 2016 2:57:20 PM UTC[TESTENRICH]Attempting to
shutdown...
June 21, 2016 2:57:20 PM UTC[TESTENRICH]Unregistered
TESTENRICH from the RoundRobinEventQueueManager
June 21, 2016 2:57:20 PM UTC[TESTENRICH]Shutdown
completed.
June 21, 2016 2:57:20 PM UTC[TESTENRICH]TESTENRICH
stopped
END LOG 1
```

3. Edit the **TESTENRICH** service by clicking the pencil icon.



4. Click the **Event Mapping** tab. Scroll down to the **Event Mapping Table**. Select **Policy-EventEnrichment-Basic**. Click **Edit**.

The screenshot shows the 'OMNibusEventReader Service' interface with the 'Event Mapping' tab selected. Below it is a table titled 'Event Mapping Table'. A tooltip at the top right says: '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.' The table has columns: Delete Selection, New, Edit (circled in red), Move Up, Move Down, and Quick Add. There are two rows in the table:

Restriction Filter	Policy Name	Active	Chain
<input checked="" type="checkbox"/> ImpactFlag1=0	POLICY-EventEnrichment-Basic	Yes	No

5. Change the Filter Expression to use **Impact Flag2= 0**. Select **POLICY-EventEnrichment-Adv** from the **Policy to Run** drop down list. Verify that the **Active** checkbox is checked. Click **OK**.

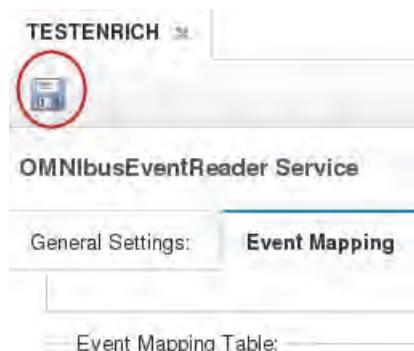
The screenshot shows the 'Create a New Event Filter' dialog box. It contains the following fields:

- Filter Expression:** ImpactFlag2=0 (circled in red)
- Policy to Run:** POLICY-EventEnrichment-Adv (circled in red)
- Active:**
- Chain:**
- Analyze Filter:** Analyze Filter button
- Buttons:** Help, Ok (circled in red), Cancel

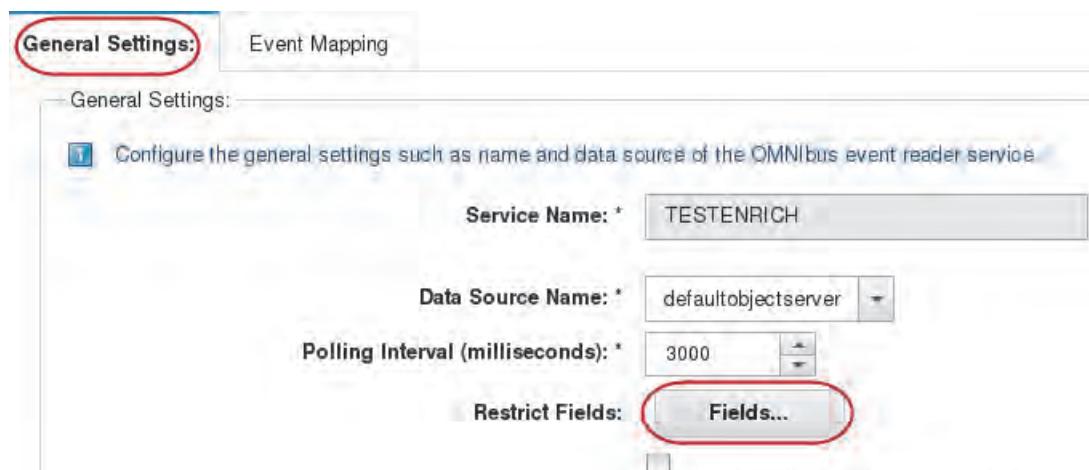
The policy shows in the mapping table as shown in the following screen capture. When this service starts, the **POLICY-EventEnrichment-Adv** policy runs. The filter selects events to pass to the policy that have a zero value in the alerts.status field; **ImpactFlag2=0**.

Event Mapping Table:			
Delete Selection    New    Edit    Move Up    Move Down    Quick Add			
Restriction Filter	Policy Name	Active	Chain
ImpactFlag2=0	POLICY-EventEnrichment-Adv	Yes	No

6. Save the changes. Click the Save icon.



7. On the **General Settings** tab, click **Fields**.

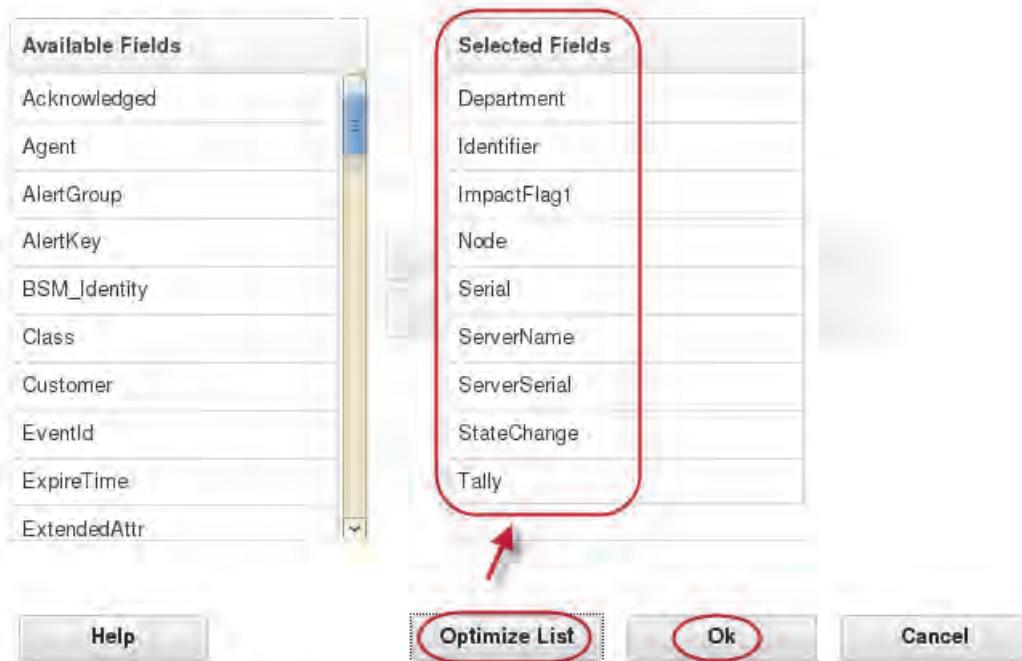


8. Click **Optimize List** to select the OMNIbus fields needed for this service.

### Select Omnibus Event Reader Fields

You can improve reader performance by selecting only the fields that you need to access in your policy.

To select which fields you want to retrieve from the ObjectServer, use the Add/Remove buttons to select fields, and the Move Up/Move Down buttons to change the selection order.



9. **Active Policies** displays the new policy **POLICY-EventEnrichment-Adv**. Click **Close**.

Information

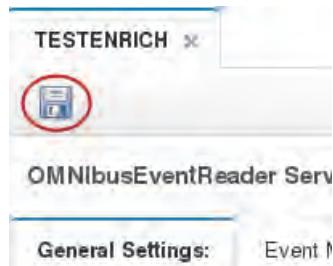
Active Policies:

- POLICY-EventEnrichment-Adv { *ImpactFlag1 Node Department* }

The fields in brackets are available to use in the listed active policies

**Close**

- Save the changes. Click the Save icon.



## Exercise 6 Verifying the results of the POLICY-EventEnrichment-Adv policy

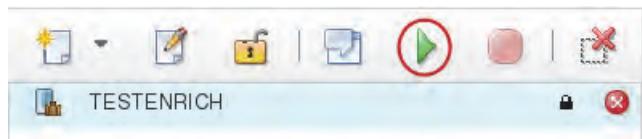


**Important:** Review the policy logic using log statements in the Policy Log and check the table values in OMNIbus.

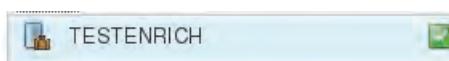
For example, verify that the following statements are true:

If the event **Severity** = 4, the **SLA** value <= to 95, and  
If the event **Severity** = 5, the **SLA** value is > 95.

- Start the **TESTENRICH** service by clicking the **Start Service** icon.



The started service will show a green checkbox.



- Open the **POLICY-Event\_Enrichment-Adv** policy window. Click the **Policy Log** icon to view results of the log statements.



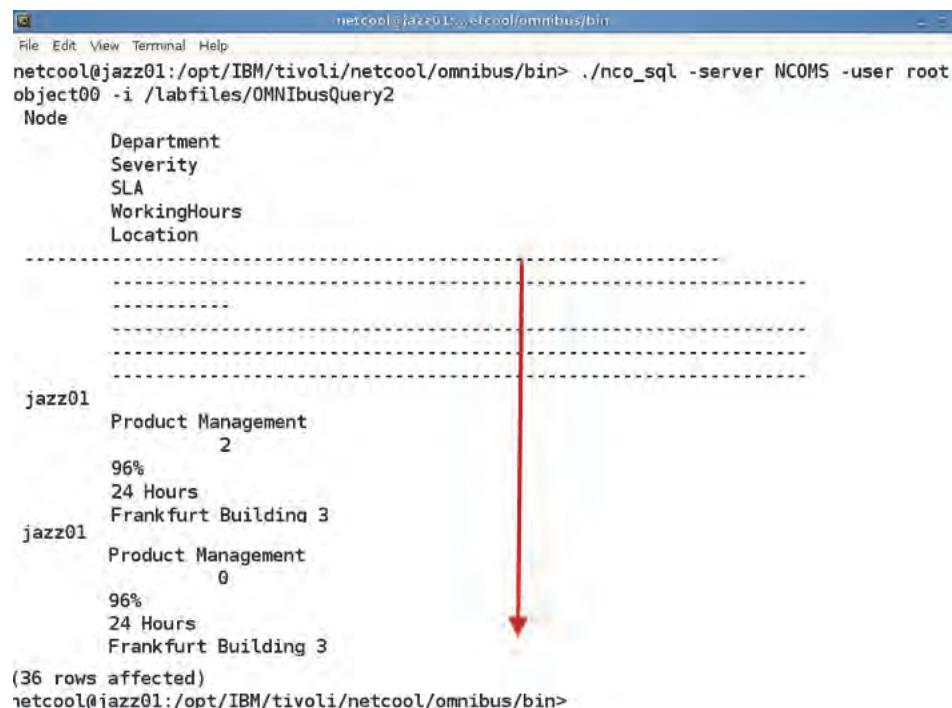
The log statements should look similar to the following screen capture.

```
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === Result from HS_Customer search ===>NULL
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === @Department value set from link ===>Sales
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === @Customer value set from link ===>Microsoft
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === @SLA value set from link ===>93%
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === SLAValue from extract ===>93
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === @Severity value calculated ===>4
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === @WorkingHours set from HS_Customer ===>9:00-17:30
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === WHTime value calculated ===>9:00-17:30
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === StartMinute ===>0
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === StartTime value calculated ===>540
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === EndMinute value calculated ===>30
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === CurrentMinute value calculated ===>8
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === CurrentTime value calculated ===>1028
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]Parser log: === @SuppressEscl value set ===>0
PolicyLogger][POLICY-EventEnrichment-Adv-LOGSTMT][MessageProcessor-Dog#8]
```

- Run the script to verify the updates to the OMNIbus **alerts.status** table using the following commands:

```
cd /opt/IBM/tivoli/netcool/omnibus/bin
./nco_sql -server NC0MS -user root -password object00 -i /labfiles/OMNIbusQuery2
```

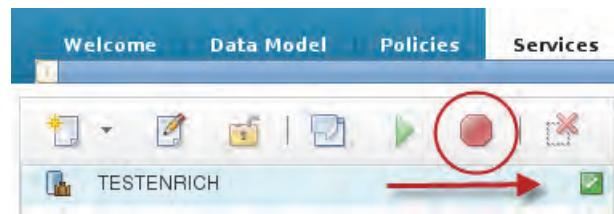
The **Node**, **Department**, **Severity**, **SLA**, **WorkingHours**, and **Location** entries in OMNIbus should look similar to the following screen capture.



```
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ./nco_sql -server NC0MS -user root
object00 -i /labfiles/OMNIbusQuery2
      Node
        Department
        Severity
        SLA
        WorkingHours
        Location
        -----
        -----
        -----
        -----
        jazz01
          Product Management
            2
            96%
            24 Hours
            Frankfurt Building 3
        jazz01
          Product Management
            0
            96%
            24 Hours
            Frankfurt Building 3
      (36 rows affected)
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin>
```

**Note:** The number of rows affected varies depending on how long the probe has been running. Also, if there are problems generating data using the restriction filter delete the TESTENRICH service and create TESTENRICH2 with the same parameters.

4. Stop the **TESTENRICH** service.



---

# Unit 7 Controlling policy execution sequence exercises

This unit focuses on event locking and policy chaining.

## Exercise 1 Observing chaining and thread locking

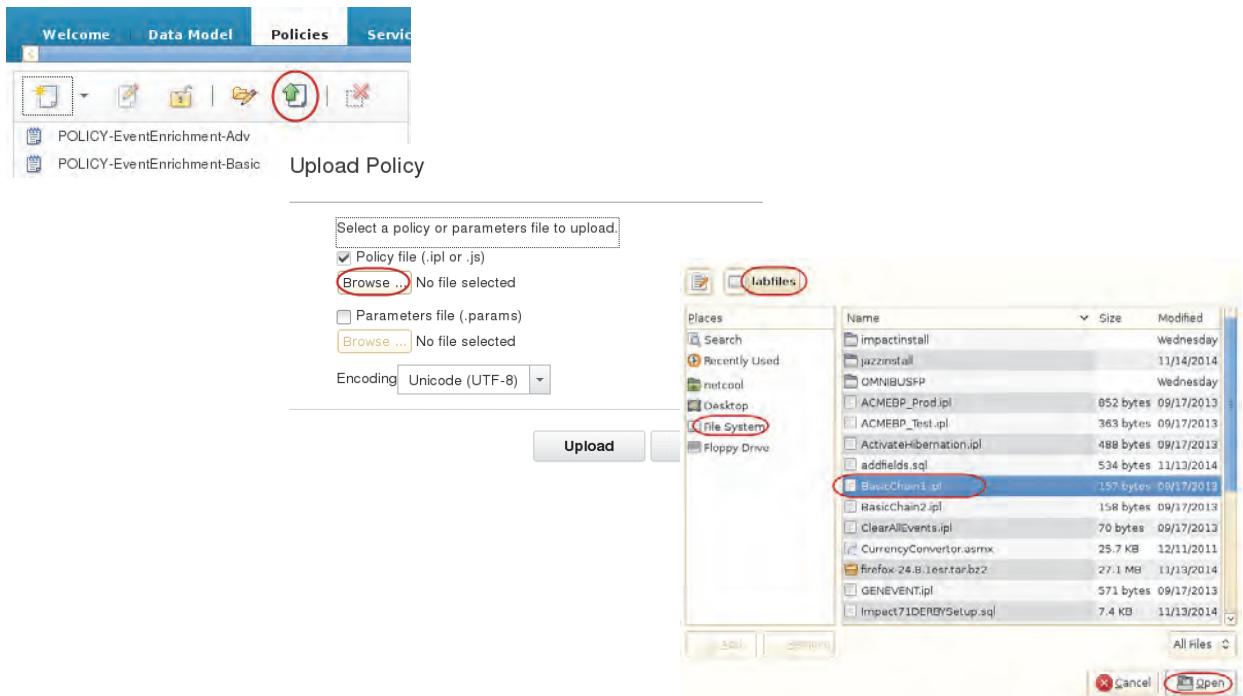
In this exercise, you observe the behavior of policies when implementing thread processing control and policy chaining. You import policies and then configure TESTENRICH for two new event mappings. The first event mapping runs **BasicChain1**. The second event mapping runs **BasicChain2**.

1. Stop the Simnet Probe by issuing the following commands to locate the process ID and stop the process.

```
ps -ef | grep nco  
kill <process id>
```

```
netcool@jazz01:~/Desktop> ps -ef | grep nco  
netcool 4256 1 0 Jun20 ? 00:06:15 /opt/IBM/tivoli/netcool/omnibus/  
platform/linux2x86/bin64/nco_objserv  
netcool 4315 1 1 Jun20 ? 00:38:21 /opt/IBM/tivoli/impact/sdk/jre/b  
in/java -javaagent:/opt/IBM/tivoli/impact/wlp/bin/tools/ws-javaagent.jar -Djava.  
awt.headless=true -XX:MaxPermSize=256m -Xms512M -Xmx1200M -Dclient.encoding.over  
ride=UTF-8 -Dhttps.protocols=SSL_TLSv2 -jar /opt/IBM/tivoli/impact/wlp/bin/tools  
/ws-server.jar ImpactUI  
netcool 4557 1 4 Jun20 ? 01:33:03 /opt/IBM/tivoli/impact/sdk/jre/b  
in/java -javaagent:/opt/IBM/tivoli/impact/wlp/bin/tools/ws-javaagent.jar -Djava.  
awt.headless=true -XX:MaxPermSize=256m -Xms512M -Xmx1200M -Dclient.encoding.over  
ride=UTF-8 -Dhttps.protocols=SSL_TLSv2 -jar /opt/IBM/tivoli/impact/wlp/bin/tools  
/ws-server.jar NCI  
netcool 15769 15659 0 05:18 pts/0 00:00:42 /opt/IBM/tivoli/netcool/omnibus/  
platform/linux2x86/probes64/nco_p_simnet  
netcool 19553 19536 0 18:00 pts/1 00:00:00 grep nco  
netcool@jazz01:~/Desktop> kill -9 15769
```

2. Click the **Upload Policy** icon. Click **Browse** to locate **BasicChain1.ipl** in the **labfiles** directory. Click **Open**.

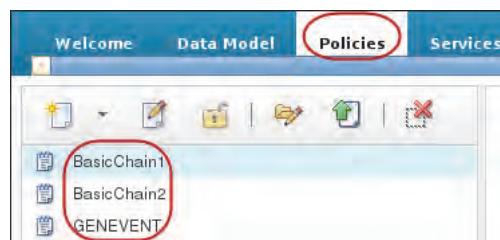


3. Click **Upload** to add the policy to the Policies list.

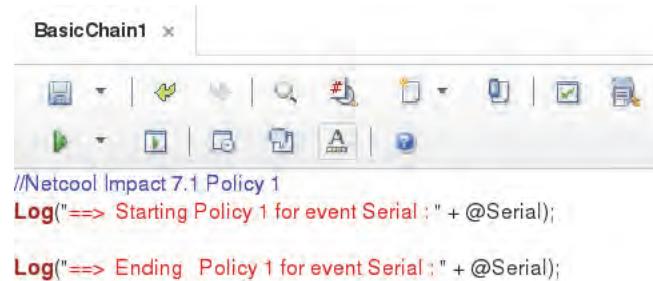


4. Repeat the steps for **BasicChain2.ipl**, and **Genevent.ipl**.

The policies are added to the Policies list.



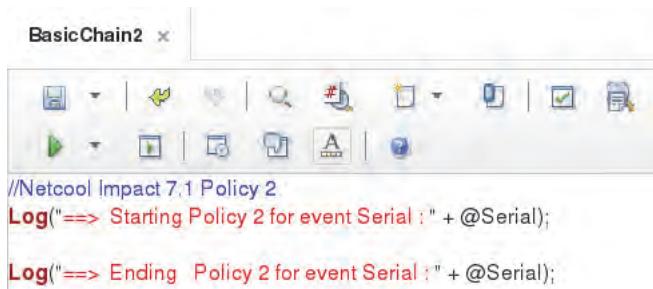
BasicChain1.ipl



```
//Netcool Impact 7.1 Policy 1
Log("==> Starting Policy 1 for event Serial :" + @Serial);

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

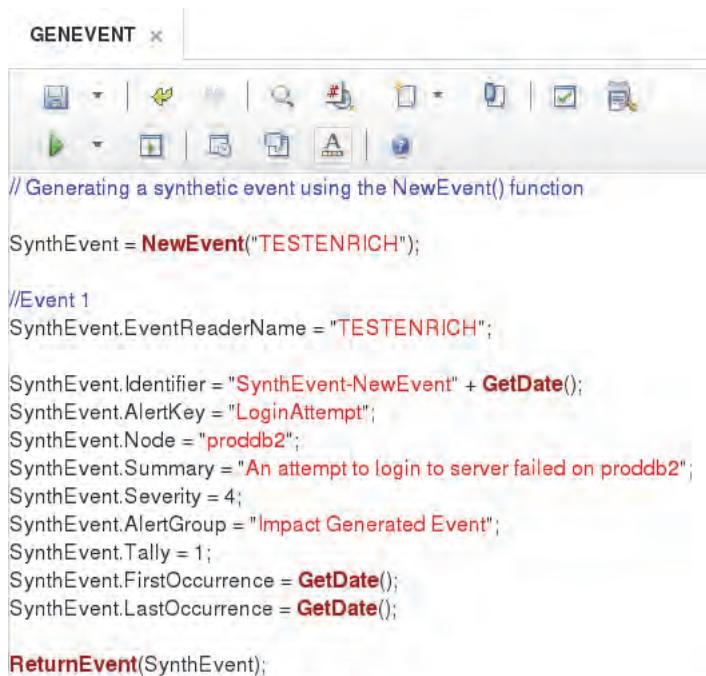
BasicChain2.ipl



```
//Netcool Impact 7.1 Policy 2
Log("==> Starting Policy 2 for event Serial :" + @Serial);

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

GENEVENT.ipl



```
// Generating a synthetic event using the NewEvent() function

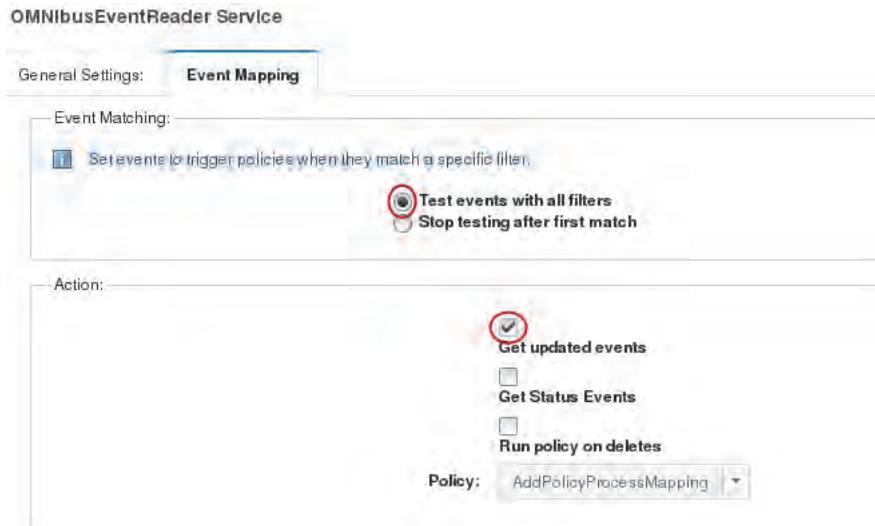
SynthEvent = NewEvent("TESTENRICH");

//Event 1
SynthEvent.EventReaderName = "TESTENRICH";

SynthEvent.Identifier = "SynthEvent-NewEvent" + GetDate();
SynthEvent.AlertKey = "LoginAttempt";
SynthEvent.Node = "proddb2";
SynthEvent.Summary = "An attempt to login to server failed on proddb2";
SynthEvent.Severity = 4;
SynthEvent.AlertGroup = "Impact Generated Event";
SynthEvent.Tally = 1;
SynthEvent.FirstOccurrence = GetDate();
SynthEvent.LastOccurrence = GetDate();

ReturnEvent(SynthEvent);
```

5. Modify the TESTENRICH service and create new mappings for the uploaded policies. Delete any existing policies in the mapping table.



6. Scroll down to the **Event Mapping Table** and click the **New** button.

The screenshot shows the 'Event Mapping Table' section. A red arrow points from the previous 'Event Mapping' configuration area down to the table. The table has columns for 'Restriction Filter', 'Policy Name', 'Active', and 'Chain'. The status bar at the bottom of the table area says 'No items to display'. The 'New' button in the toolbar is circled in red.

**Note:** Be sure that **Event Locking** is not selected in the Event Locking section.

The **Edit Event Filter** window opens.

- a. Leave the **Filter Expression** field blank.
- b. From the **Policy to Run** list, select **BasicChain1**.
- c. Check the **Active** box.

- d. Click **OK**.



7. Click the **New** button.

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.				
Delete Selection	New	Edit	Move Up	Move Down
Restriction Filter	Policy Name	Active	Chain	
<input type="checkbox"/>	BasicChain1	Yes	No	

The **Edit Event Filter** window opens.

- a. Leave the **Filter Expression** field blank.
- b. From the **Policy to Run** list, select **BasicChain2**.
- c. Check the **Active** box.
- d. Click **OK**.



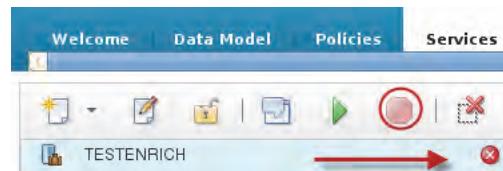
The resulting mappings are listed as shown in the following example.

Restriction Filter	Policy Name	Active	Chain
<input type="checkbox"/>	BasicChain1	Yes	No
<input type="checkbox"/>	BasicChain2	Yes	No

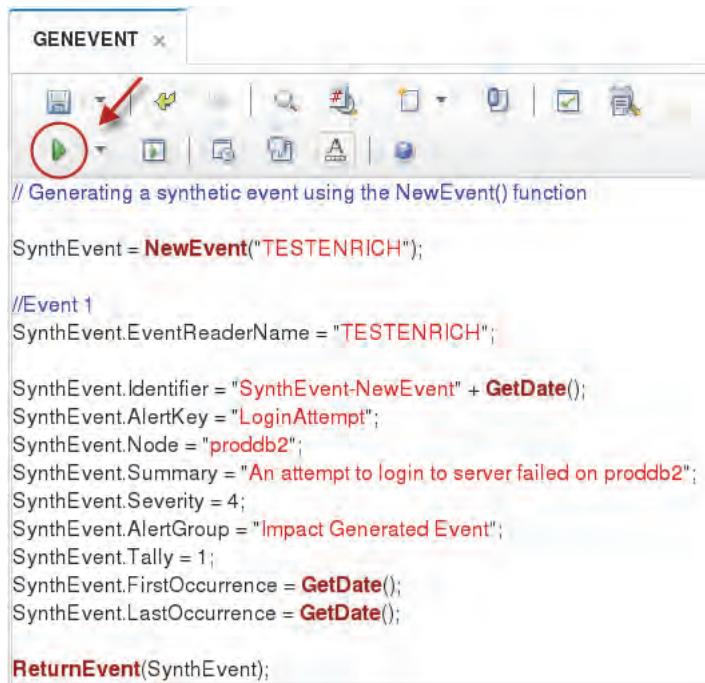
8. Save the changes. Click the **Save** icon.



9. Stop the **TESTENRICH** service.



10. Select the **GenEvent** policy in the **Policies** list. Click the green arrow to execute the policy.



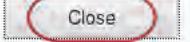
```
GENEVENT
// Generating a synthetic event using the NewEvent() function
SynthEvent = NewEvent("TESTENRICH");
//Event 1
SynthEvent.EventReaderName = "TESTENRICH";
SynthEvent.Identifier = "SynthEvent-NewEvent" + GetDate();
SynthEvent.AlertKey = "LoginAttempt";
SynthEvent.Node = "proddb2";
SynthEvent.Summary = "An attempt to login to server failed on proddb2";
SynthEvent.Severity = 4;
SynthEvent.AlertGroup = "Impact Generated Event";
SynthEvent.Tally = 1;
SynthEvent.FirstOccurrence = GetDate();
SynthEvent.LastOccurrence = GetDate();

ReturnEvent(SynthEvent);
```

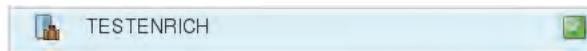
11. Click **Close**.

Running policy

 Successfully triggered policy "GENEVENT". Please refer to policy log for policy output.

 Close

12. Start the **TESTENRICH** service.



13. View the output in the policy logger For the GENEVENT policy.



Notice how Policy 2 starts before Policy 1 is finished with the event.

```
Parser log: ==> Starting Policy 1 for event Serial : 16
Parser log: ==> Starting Policy 1 for event Serial : 182
Parser log: ==> Starting Policy 2 for event Serial : 182
Parser log: ==> Ending  Policy 1 for event Serial : 182
Parser log: ==> Ending  Policy 1 for event Serial : 16
Parser log: ==> Ending  Policy 2 for event Serial : 182
```

14. Modify the **TESTENRICH** service and select the **Chain** check box next to each policy as shown in the example:

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.

<input type="checkbox"/> Delete Selection	New	Edit	Move Up	Move Down	Quick Add
Restriction Filter	Policy Name	Active	Chain		
<input type="checkbox"/>	BasicChain1	Yes	<input checked="" type="checkbox"/> Yes		
<input type="checkbox"/>	BasicChain2	Yes	<input checked="" type="checkbox"/> Yes		

15. Save the changes. Stop the **TESTENRICH** service. Start the service. Clear the policy log. Execute the **GenEvent** policy in the policy editor. View the policy log.

```
|Parser log: ==> Starting Policy 1 for event Serial : 430
|Parser log: ==> Ending Policy 1 for event Serial : 430
|Parser log: ==> Starting Policy 2 for event Serial : 3
|Parser log: ==> Ending Policy 2 for event Serial : 3
|Parser log: ==> Starting Policy 2 for event Serial : 430
|Parser log: ==> Ending Policy 2 for event Serial : 430
```

This time Policy 1 ends before Policy 2 starts for the same serial.

## Exercise 2 Configuring thread locking

Using event locking, the multi-threaded Event Reader sorts incoming events based on alert fields. It then processes them one category at a time in the order sent by the Event Reader.

1. Configure event locking on the **TESTENRICH Event Mapping Configuration** window for **BasicChain1** and **BasicChain2**. Select the **Enable** check box and, for this lab, type **Node** in the **Expression** field.

Event Locking:

Enable   
Expression: Node

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.

<input type="checkbox"/> Delete Selection	New	Edit	Move Up	Move Down	Quick Add
Restriction Filter	Policy Name	Active	Chain		
<input type="checkbox"/>	BasicChain1	Yes	<input checked="" type="checkbox"/> No		
<input type="checkbox"/>	BasicChain2	Yes	<input checked="" type="checkbox"/> No		

2. Restart the **TESTENRICH** service. Execute the **GenEvent** policy two or three times. View the policy log.

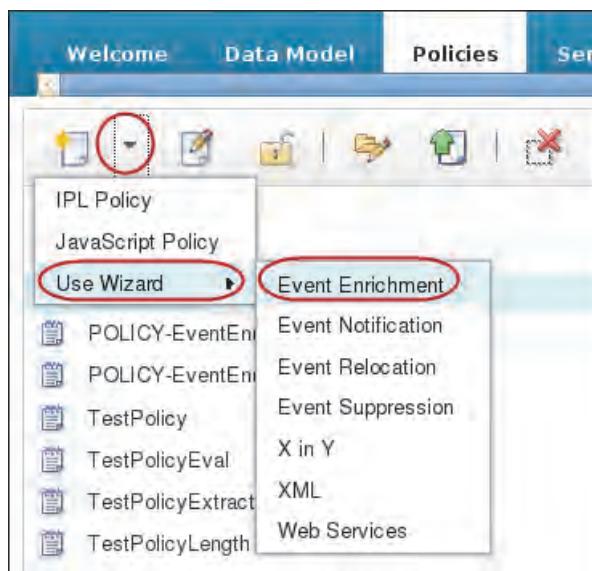


# Unit 8 Policy wizards exercises

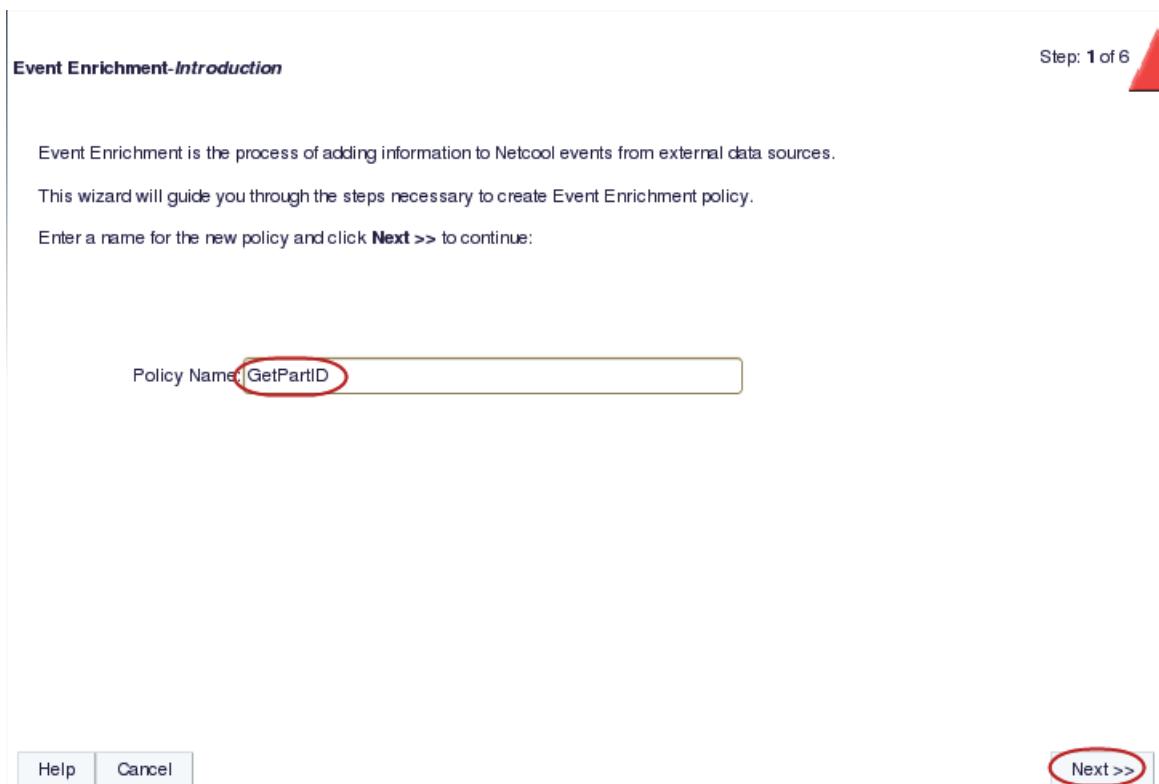
In this unit, you work with the policy wizard.

## Exercise 1 Creating an enrichment policy using the wizard

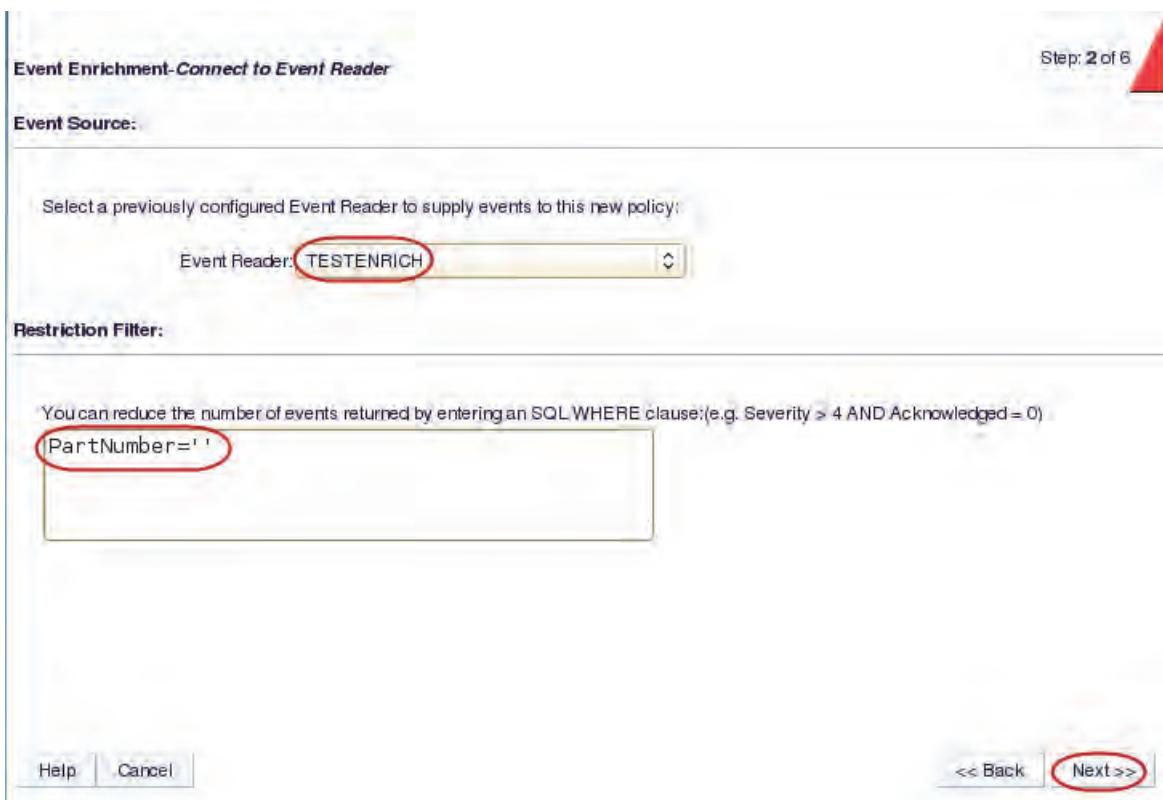
1. Click **Policies** to open the **Policies** tab.
2. Click the **New Policy** icon. Select **Use Wizard > Event Enrichment** in the list.



3. Type **GetPartID** in the **Policy Name** field. Click **Next**.



4. Select **TESTENRICH** as the Event Reader. Type **PartNumber= "** (two single quotation marks) in the **Restriction Filter** field. The restriction clause is set against fields in the Event Reader. Click **Next**.



5. Create a new data type by performing the following steps:
- Type **HS\_Inventory** in the **Data Type** field.
  - Select **HS\_TRAIN** from the **Data Source** list.

- c. Click **Next**.

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 >>

6. In the **Data Source Base Table** section, select **IMPACT** and **INVENTORY** in the **Base Table** lists to specify the table name that is being accessed from the **HS\_TRAIN** source.
- Click **Refresh**.
  - In the **Database Filter Builder** section, specify the following values for the database query.

Field	Value
Field	NODE
Comparator	=
Value	Node

- Click **Next**.

These settings specify the field in the table that is matched against the field in the ObjectServer. In this case, **NODE** from the HS\_Inventory table matches against **Node** from the ObjectServer.

**Event Enrichment-Identify Information**

**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 should be enriched. Choosing a value from the Field dropdown will cause the Comparator and Value dropdowns to dynamically update based on your choice.

Choose match criteria:

Match all     Match any

Field	Comparator	Value
NODE	=	Node

Add Conditions View SQL Clear All

Help Cancel << Back Next >>

If no fields are displayed in the list, most likely the ObjectServer source used for the Event Reader is not available, or it is configured improperly.

7. Select **PartNumber** from the **Event Fields** list. Click **Refresh**. Select **PARTID** in the **Available Data Source Fields** list, and click **Add**. PARTID moves to the **Selected Data Source Fields** list. Click **Apply**. Click **Next**.

**Event Enrichment-Enrich Event**

**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	<input type="button" value="Add »"/>	Selected Data Source Fields:
INTERFACE		<b>PARTID</b>
STATUS	<input type="button" value="Remove"/>	
TYPE		
VERSION		

Apply

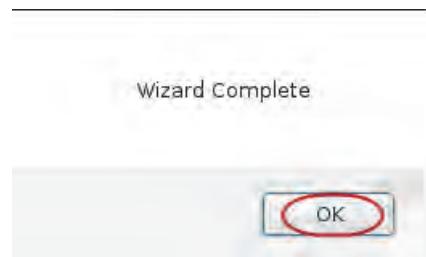
Help Cancel << Back **Next >>**

Note: PARTID is from the inventory table.

8. When the Summary and Finish window opens, click **Apply**. Click **Finish** to create the policy.

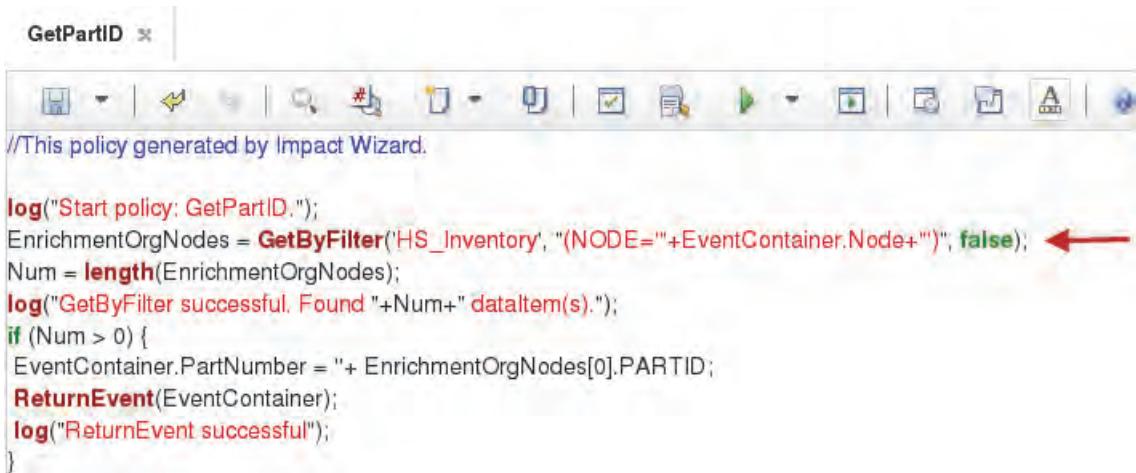


9. Click **OK** to close the window.



The new **GetPartID** policy opens in the policy window.

10. Click the **Save** icon.

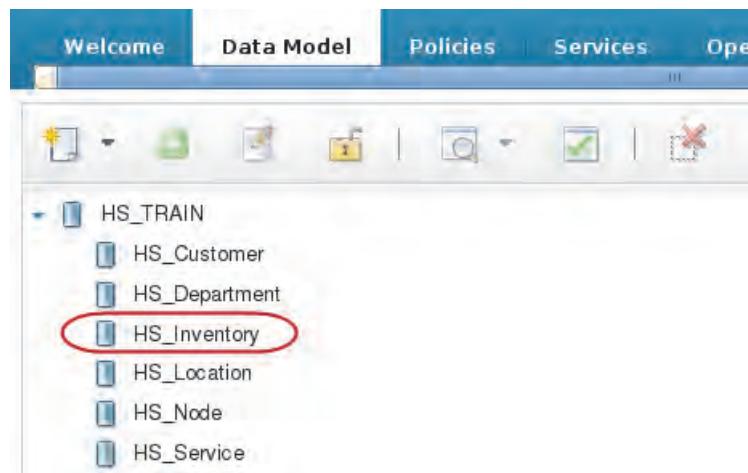


```
GetPartID x

//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");
}
```

11. Click **Data Model** and expand the Data Source **HS\_TRAIN**.

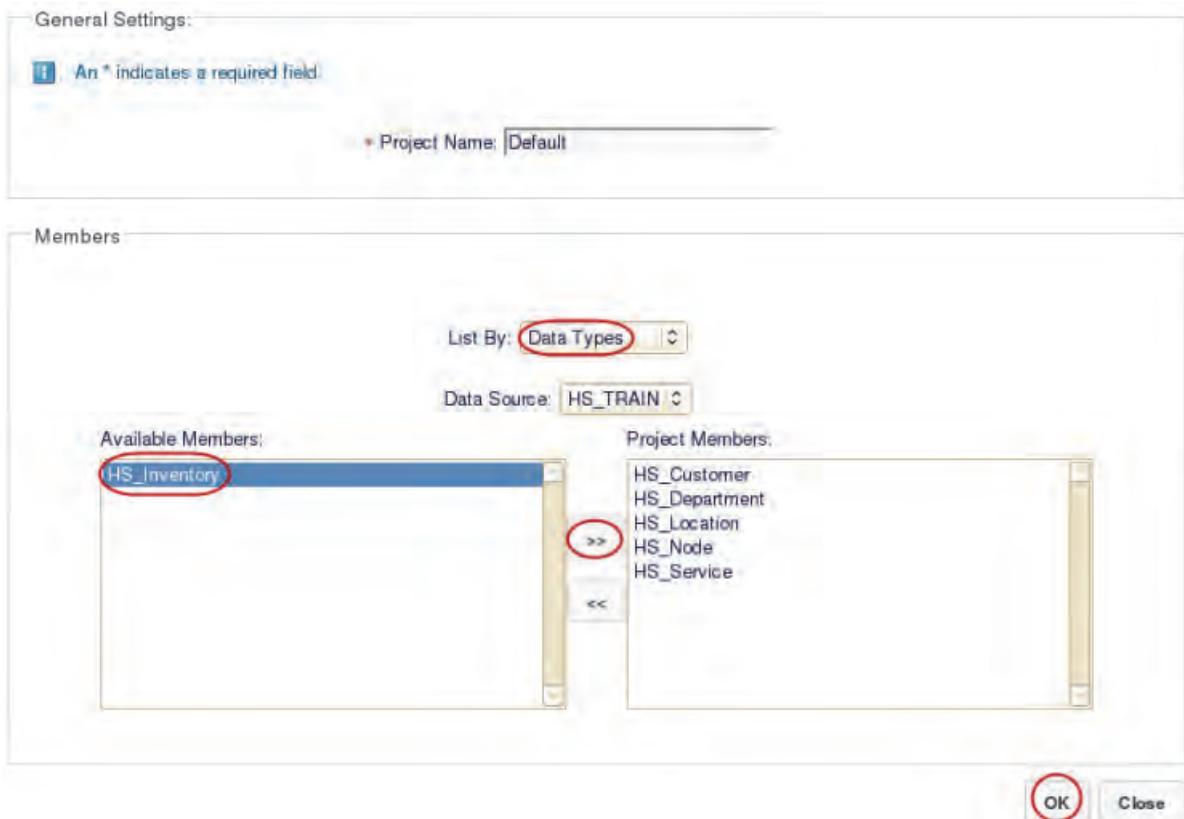


If **HS\_Inventory** is not listed, add it from the Global Repository.

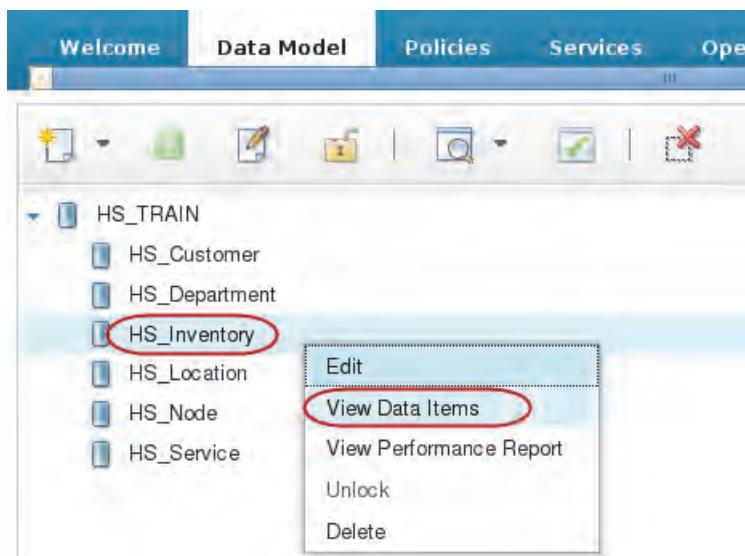
i. Edit the Default project.



- ii. List By **Data Types**. Select **HS\_Inventory** and add it to the **Default** project. Click **OK**.



12. Right-click **HS\_Inventory** and select **Edit** to view the table description. Right-click **HS\_Inventory** and select **View Data Items** to see field values.



The table should look similar to the following screen capture.

Items: HS\_Inventory ✎

Filter:

Data Type Name: HS\_Inventory

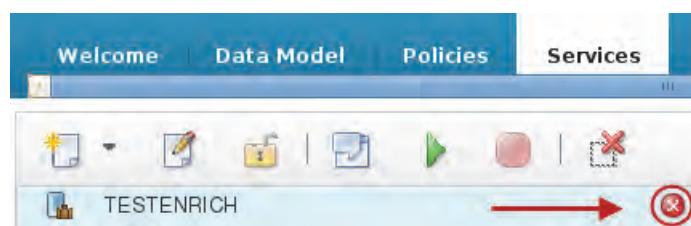
Number of Objects: 13

Filter Retrieved Data Items:

Select	PARTID	NODE	INTERFACE	STATUS	TYPE	VERSION	View Links	Edit
<input type="checkbox"/>	DAL-99e309	gambit		1	Cisco	IOS 12.0		
<input type="checkbox"/>	DAL-93488f	orac	1	1	Sun	Freeware		
<input type="checkbox"/>	DAL-934r588f	wombat	1	1	Sun	Freeware		
<input type="checkbox"/>	DAL-934r88f	muppet	1	1	Sun	Freeware		
<input type="checkbox"/>	DAL-9345488f	moose	1	1	Sun	Freeware		
<input type="checkbox"/>	SFO-93488DW	link1	1	1	Sun	Freeware		
<input type="checkbox"/>	SFO-93488g	link2	1	1	Sun	Freeware		

Notice the values for **PARTID** and **NODE**.

13. Verify that the **TESTENRICH** service is stopped.



14. Edit the service and make sure that there is only one policy **GetPartID**, defined to the service.  
Save any changes.

The screenshot shows the 'TESTENRICH' service configuration window. The 'Event Mapping' tab is selected. Below it is a table titled 'Event Mapping Table' with the following data:

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

A red circle highlights the row where 'PartNumber=""' is set as the restriction filter and 'GetPartID' is the policy name.

15. Double-click the **Gnome Terminal** icon on the desktop.



16. Start the Simnet Probe using the following commands:

```
cd /opt/IBM/tivoli/netcool/omnibus/probes/  
.nco_p_simnet &
```



Note: The TESTENRICH service was configured by the wizard to include the GetPartID policy and the policy filter.

17. Start the TESTENRICH service.



## 18. View the policy log.

```
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: GetByFilter successful. Found 1 dataItem(s)
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: ReturnEvent successful
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: Start policy: GetPartID.
[PolicyLogger][GetPartID][MessageProcessor-Dog#2]Parser log: GetByFilter successful. Found 0 dataItem(s).
[PolicyLogger][GetPartID][MessageProcessor-Dog#2]Parser log: Start policy: GetPartID.
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: GetByFilter successful. Found 0 dataItem(s).
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: Start policy: GetPartID.
[PolicyLogger][GetPartID][MessageProcessor-Dog#2]Parser log: GetByFilter successful. Found 0 dataItem(s).
[PolicyLogger][GetPartID][MessageProcessor-Dog#2]Parser log: Start policy: GetPartID.
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: GetByFilter successful. Found 0 dataItem(s).
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: Start policy: GetPartID.
[PolicyLogger][GetPartID][MessageProcessor-Dog#2]Parser log: GetByFilter successful. Found 0 dataItem(s).
[PolicyLogger][GetPartID][MessageProcessor-Dog#2]Parser log: Start policy: GetPartID.
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: GetByFilter successful. Found 0 dataItem(s).
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: Start policy: GetPartID.
[PolicyLogger][GetPartID][MessageProcessor-Dog#2]Parser log: GetByFilter successful. Found 0 dataItem(s).
[PolicyLogger][GetPartID][MessageProcessor-Dog#2]Parser log: Start policy: GetPartID.
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: GetByFilter successful. Found 1 dataItem(s).
[PolicyLogger][GetPartID][MessageProcessor-Dog#1]Parser log: ReturnEvent successful
```

19. Type the following commands to verify **Node** and **PartNumber** updates in OMNIbus.

```
cd /opt/IBM/tivoli/netcool/omnibus/bin/
./nco_sql -server NCMS -user root -password object00 -i
/labfiles/OMNIbusQuery3
```

```
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ./nco_sql -server NCMS -use
r root -password object00 -i /labfiles/OMNIbusQuery3
Node          PartNumber
-----
link3          LON-93488ef
link2          SF0-93488g
link3          LON-93488ef
link2          SF0-93488g
link1          SF0-93488DW
link5          LGA-393488f
link4          LON-943488f
link5          LGA-393488f
link4          LON-943488f
link1          SF0-93488DW

(10 rows affected)
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin>
```

# Unit 9 Notification policies exercises

In this unit, you configure a notification policy.

## Exercise 1 Notification

This exercise uses the Netcool/Impact email service to send a notification. Validate the send and receive email function on your machine using the **mail** command as follows:

1. To log in as the **root** user, type **su - root** from the command prompt.

2. Type **object00** when prompted for the password.

3. At the command prompt, type the following text:

**mail netcool@jazz01.tivoli.edu**

4. When prompted for a subject, type the following text:

**Network Status**

5. Press Enter. Type the following text:

**The server in the security room is down.**

6. To send the message, type a period (.) on a line by itself and press the Enter key. Type **exit** to switch back to the **netcool** user.

```
Directory: /home/netcool/Desktop
Tue Jun 21 21:11:07 UTC 2016
netcool@jazz01:~/Desktop> su - root
Password:
jazz01:- # mail netcool@jazz01.tivoli.edu
Subject: Network Status
The server in the security room is down.

.
EOT
jazz01:- # exit
logout
You have mail in /var/spool/mail/netcool
netcool@jazz01:~/Desktop> mail
Heirloom mailx version 12.2 01/07/07.  Type ? for help.
"/var/spool/mail/netcool": 1 message 1 new
>N 1 root@jazz01.tivoli Tue Jun 21 21:17 18/647 Network Status
?
```

7. To view messages, at the command prompt type **mail**. Hit the enter key again to display the details.

```
?  
Message 1:  
From root@jazz01.tivoli.edu Tue Jun 21 21:17:56 2016  
X-Original-To: netcool@jazz01.tivoli.edu  
Delivered-To: netcool@jazz01.tivoli.edu  
Date: Tue, 21 Jun 2016 21:17:56 +0000  
To: netcool@jazz01.tivoli.edu  
Subject: Network Status ←  
User-Agent: Heirloom mailx 12.2 01/07/07  
MIME-Version: 1.0  
Content-Type: text/plain; charset=us-ascii  
Content-Transfer-Encoding: 7bit  
From: root@jazz01.tivoli.edu (root)  
  
The server in the security room is down. ←
```

?

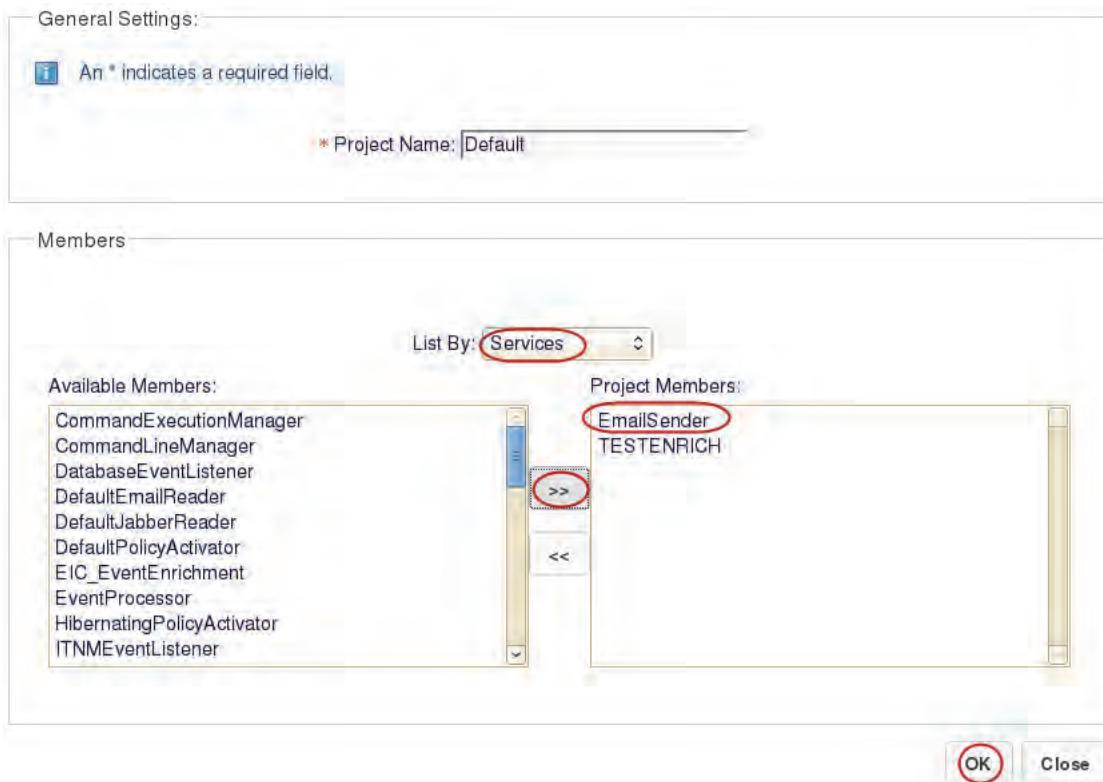
8. Repeat the steps in this exercise as the **netcool** user. Type a response to the original email as shown in the example slide.

Repeating the steps verifies that the netcool user **netcool** can send mail to the **root** user at **root@jazz01.tivoli.edu**.

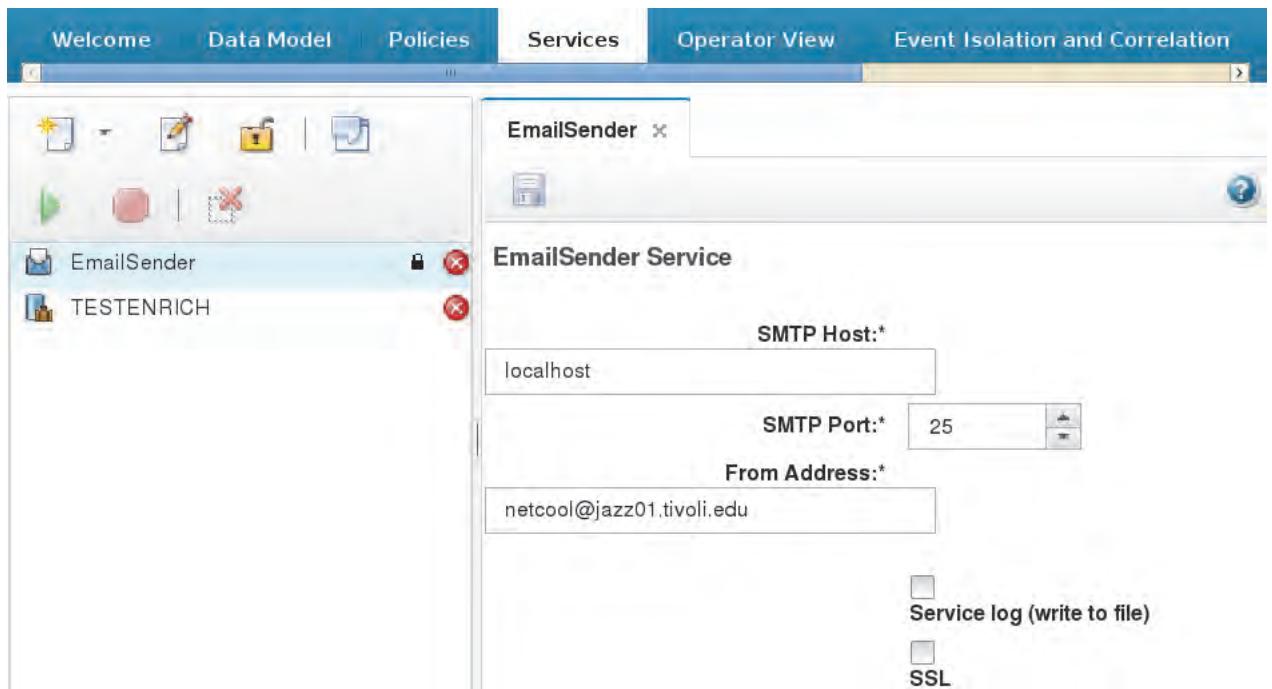
```
netcool@jazz01:~/Desktop> mail root@jazz01.tivoli.edu ←  
Subject: re:Network Status ←  
We are working on the problem. ←  
. ←  
EOT  
You have new mail in /var/spool/mail/netcool  
netcool@jazz01:~/Desktop> su - root ←  
Password:  
jazz01:- # mail ←  
Heirloom mailx version 12.2 01/07/07. Type ? for help.  
"/var/spool/mail/root": 1 message 1 new  
>N 1 netcool@jazz01.tivoli.edu Tue Jun 21 21:35 18/650 re:Network Status  
?  
Message 1:  
From netcool@jazz01.tivoli.edu Tue Jun 21 21:35:53 2016  
X-Original-To: root@jazz01.tivoli.edu  
Delivered-To: root@jazz01.tivoli.edu  
Date: Tue, 21 Jun 2016 21:35:53 +0000  
To: root@jazz01.tivoli.edu  
Subject: re:Network Status ←  
User-Agent: Heirloom mailx 12.2 01/07/07  
MIME-Version: 1.0  
Content-Type: text/plain; charset=us-ascii  
Content-Transfer-Encoding: 7bit  
From: netcool@jazz01.tivoli.edu (Netcool User)  
  
We are working on the problem.  
?
```

# Exercise 2 Configuring the EmailSender service

1. Add the **EmailSender** service to the **Default** project.



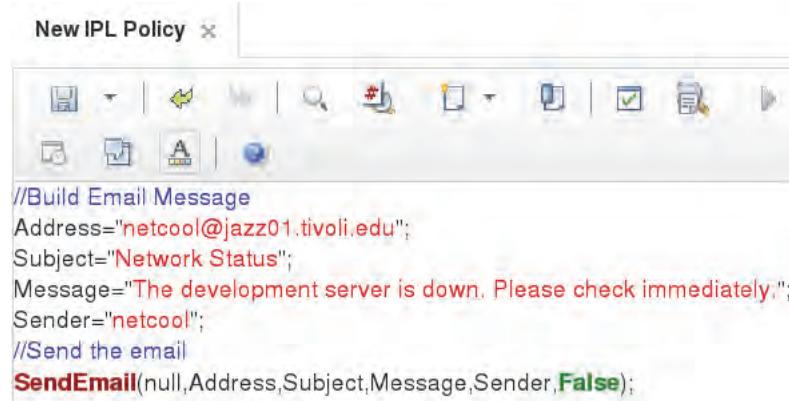
2. Edit the **EmailSender** service. Set the fields as shown in the following screen capture. Click the **Save** icon.



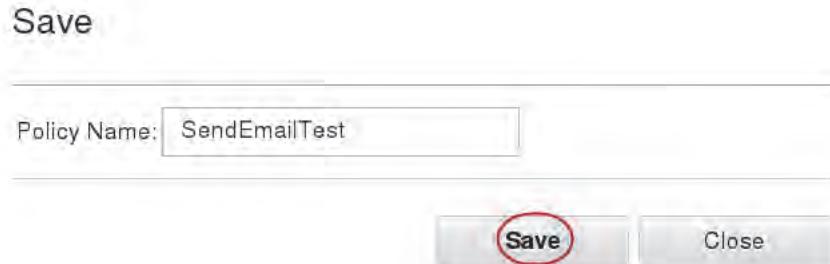
## Exercise 3 Creating the policy

1. Create a new IPL policy and add the lines as displayed.

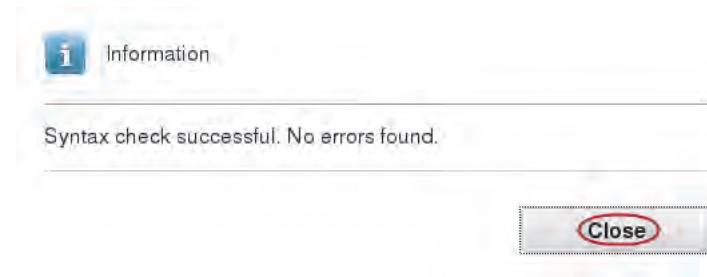
```
//Build Email Message
Address="netcool@jazz01.tivoli.edu";
Subject="Network Status";
Message="The development server is down. Please check immediately.";
Sender="netcool";
//Send the email
SendEmail(null,Address,Subject,Message,Sender,False);
```



2. Click the **Save** icon. Type **SendEmailTest** as Policy Name.



3. Check the syntax icon. Click **Close**.



4. Click the **Run** icon to test the policy. Change the **Subject** line in the policy and run the policy a few more times.
5. Click **OK** to continue each time.



# Exercise 4 Verifying SendEmailTest results

1. Click the **View Service Log** icon.



2. Verify that the **EmailSender** service log contains a **SendEmailTest** message.
  - a. On the **Services** tab, click the **EmailSender** service.
  - b. Click the **View Service Log** icon.



3. Open a GNOME Terminal.



- Type **mail** command in a terminal window to view the new messages for the **netcool** user. In this example, there are six messages. Press the **Enter key** to see new messages.

The screenshot shows two terminal windows. The top window is titled "netcool@jazz01:~/Desktop>" and displays a list of six email messages from the "Heirloom mailx" version 12.2. The bottom window is also titled "netcool@jazz01:~/Desktop>" and shows the details of the fourth message, which has a red box around it. The message content includes the recipient, subject, and body, with the body containing the text "The development server is down. Please check immediately." highlighted with a red oval.

```
netcool@jazz01:~/Desktop> mail
Heirloom mailx version 12.2 01/07/07. Type ? for help.
"/var/spool/mail/netcool": 6 messages 2 new 6 unread
U 1 root@jazz01.tivoli Tue Jun 21 21:17 19/657 Network Status
U 2 netcool@jazz01.tivoli Tue Jun 21 21:33 19/662 re:Network Status
U 3 netcool@jazz01.tivoli Wed Jun 22 03:59 24/886 Network Status
U 4 netcool@jazz01.tivoli Wed Jun 22 04:05 24/885 Network Status
>N 5 netcool@jazz01.tivoli Wed Jun 22 04:16 23/878 Network Status
N 6 netcool@jazz01.tivoli Wed Jun 22 04:16 23/871 Network Status

Message 5:
From netcool@jazz01.tivoli.edu Wed Jun 22 04:16:47 2016
X-Original-To: netcool@jazz01.tivoli.edu
Delivered-To: netcool@jazz01.tivoli.edu
From: netcool@jazz01.tivoli.edu
To: netcool@jazz01.tivoli.edu
Subject: Network Status
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="----=_Part_16_-2127512087.1466569007337"
Date: Wed, 22 Jun 2016 04:16:47 +0000 (UTC)

Part 1:
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

The development server is down. Please check immediately.
```

- Modify the policy to send an email to the **root** user.

The screenshot shows the "SendEmailTest" interface. It contains a code editor with the following Java code:

```
//Build Email Message
Address="root@jazz01.tivoli.edu";
Subject="Network Status";
Message="The development server is down. Please check immediately.";
Sender="netcool";
//Send the email
SendEmail(null,Address,Subject,Message,Sender,False);
```

- View the **EmailSender** log results.

The screenshot shows the "Default" log viewer. It displays a single log entry:

```
Service 1: EmailSender
Filter: [empty]
Apply: [empty]

START LOG 1
June 22, 2016 4:40:00 AM UTC[EmailSender]Sent E-Mail
to: root@jazz01.tivoli.edu
END LOG 1
```

7. Open a terminal session and change to the **root** user. Use the **Mail** command to view the email messages. The results will be similar to what is displayed below.

The screenshot shows a terminal window titled "netcool@jazz01:~". The window contains the following text:

```
File Edit View Terminal Help
netcool@jazz01:~> su - root
Password:
jazz01:~ # mail
Heirloom mailx version 12.2 01/07/07. Type ? for help.
"/var/spool/mail/root": 2 messages 2 new
>N 1 netcool@jazz01.tiv Tue Jun 21 21:35 18/650 re:Network Status
 N 2 netcool@jazz01.tiv Wed Jun 22 04:40 23/859 Network Status
?
Message 1:
From netcool@jazz01.tivoli.edu Tue Jun 21 21:35:53 2016
X-Original-To: root@jazz01.tivoli.edu
Delivered-To: root@jazz01.tivoli.edu
Date: Tue, 21 Jun 2016 21:35:53 +0000
To: root@jazz01.tivoli.edu
Subject: re:Network Status ←
User-Agent: Heirloom mailx 12.2 01/07/07
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
From: netcool@jazz01.tivoli.edu (Netcool User)

We are working on the problem. ←
```

Red annotations include: a red circle around "su - root", a red circle around "mail", a red arrow pointing to the "Subject" line, and a red arrow pointing to the "We are working on the problem." message.

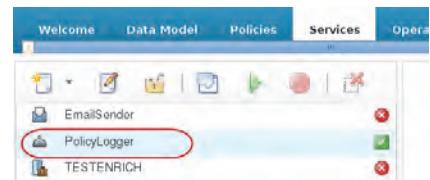
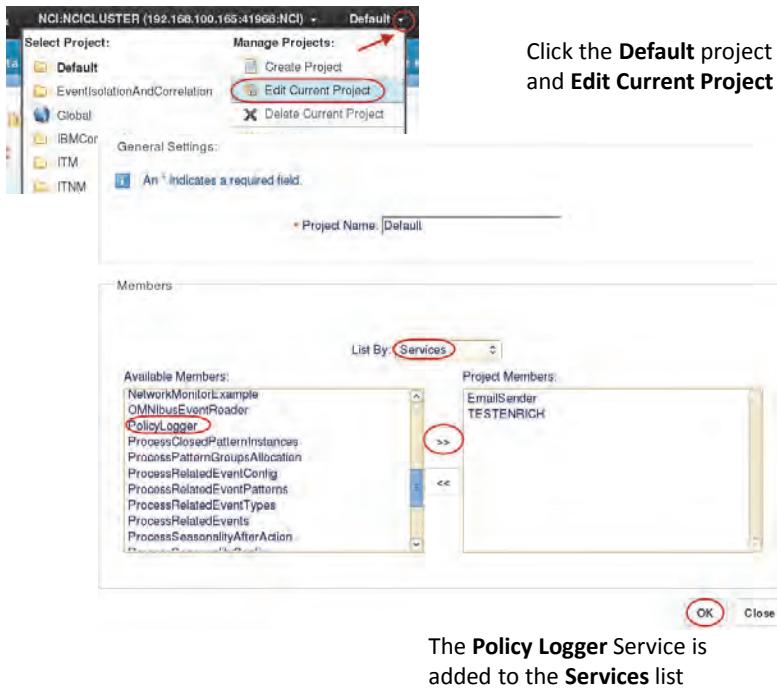
# Unit 10 Reports exercises

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

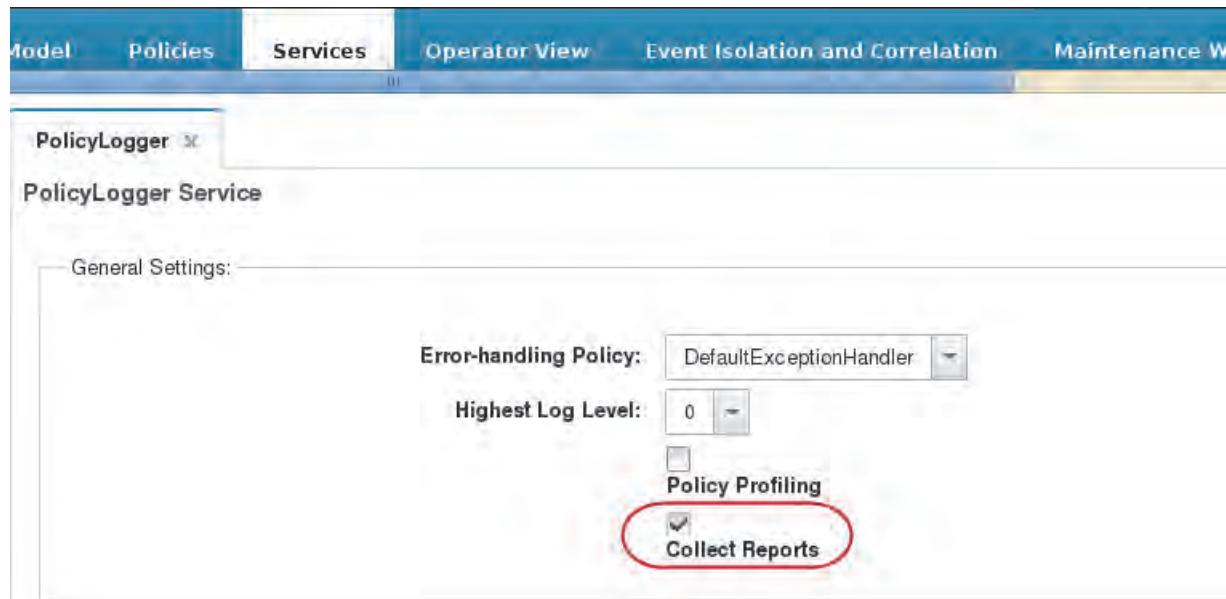
## Exercise 1 Enabling report collection

In this lab exercise, you use the Netcool/Impact preinstalled reporting tools to generate reports for policies created in class. Reporting data is generated during policy execution if the **Collect Reports** feature is enabled.

1. Click **Services** to open the **Services** tab.
2. If the **PolicyLogger** is not listed in the **Services** list, add the **PolicyLogger** service to the **Default** project as shown.



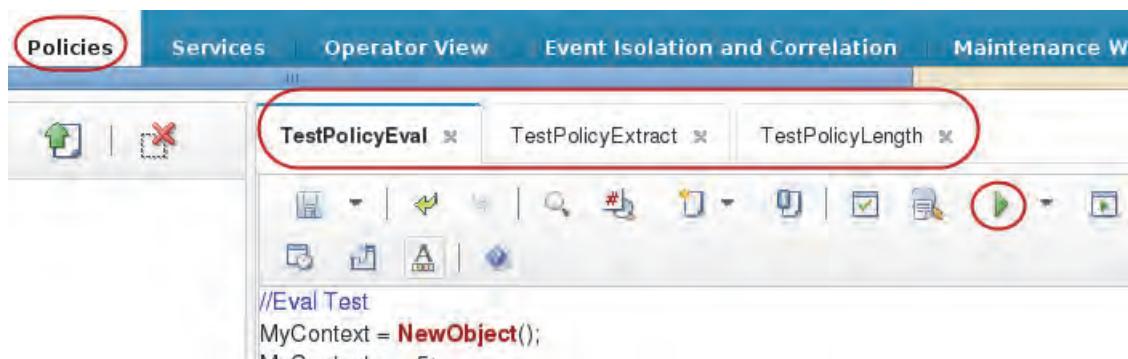
3. Edit the **PolicyLogger** configuration either by double-clicking the **Policy Logger** entry in the **Services** list or by right-clicking the entry and selecting **EDIT**. Select the check box to enable **Collect Reports** as shown in the display.



4. Click the **Save Service** icon.



5. Open the **Policies** window and run three policies that you created in **Unit 4: Policies Exercises** by clicking the green arrow for each. Run each policy three to five times to create reporting data.



## Exercise 2 Creating the Impact Profile report

1. Click the **Reports** tab on the far right.



The **Impact Profile Report** provides information about the efficiency of the Netcool/Impact server configuration. The detailed view shows information about SQL queries, policies that issued the queries, type of action, data sources queried, and metrics.

2. Select the **Impact Profile Report**.

Policy Efficiency Report	Profile Description	Result
Policy Error Report	Queries sent to same data source by same policy > 3 times in 5 seconds	0
Operator Efficiency Report	Queries done > 10 times in 60 seconds that are taking > 500 milliseconds	0
Node Efficiency Report	Queries made > 10 times in 60 seconds that return > 50 rows	0
Action Error Report	Inserts into any types > 10 times in 60 seconds that are taking > 500 milliseconds	0
Action Efficiency Report	Same identifier updated by ReturnEvent > 20 in 30 seconds	0
Impact ROI Efficiency Report	Same identifier inserted into the same objectServer that impact reads events from	0
Impact Profile Report	Internal types written more than 100 in 30 seconds	0
	JRExec calls done more than 3 times in 5 seconds that are taking > 60 seconds	0
	Hibernations built up in memory > 1000	false

Pages: 1 of 1 Rows: 9

3. Click the wrench icon to **Open Configuration** window.



The **Impact Profile Report Rules Editor** lists all the queries you can use to generate an **Impact Profile Report**. The current configuration is as displayed. This report is useful if you want to evaluate sql efficiency in an existing Netcool/Impact production environment.

**SQL Query X in Y Rules**

- SQL Query Count Threshold (times): 3
- SQL Query Count Time Window (seconds): 5

**SQL Hot Spot Rules**

- Query Return Row Threshold (rows): 500
- Query Execution Count Threshold (times): 500
- Query Execution Time Window (seconds): 50
- Insert Execution Time Threshold (milliseconds): 10
- Query Execution Time Threshold (milliseconds): 60

**JRExecAction Rules**

- Execution Count Threshold (times): 3
- Execution Time Threshold (seconds): 5
- Execution Time Window (seconds): 60

**Hibernation Rules**

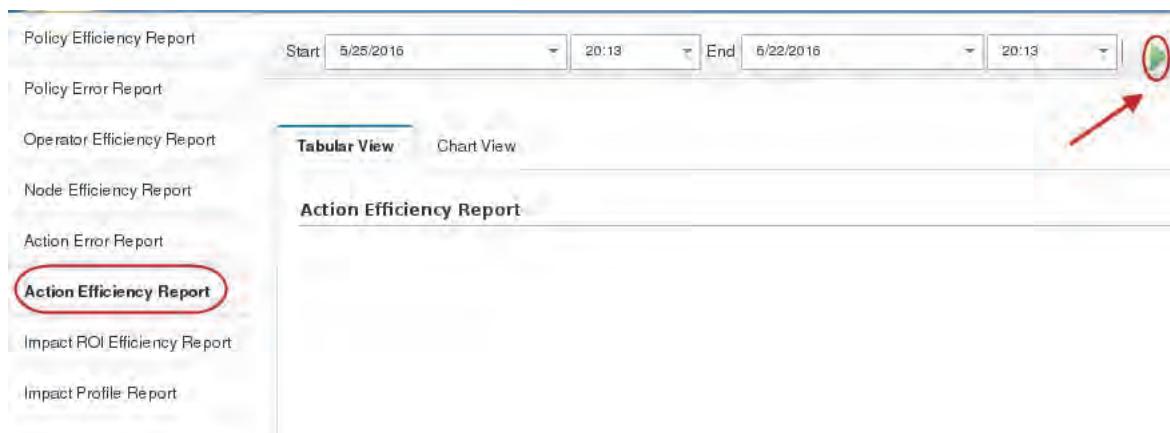
- Hibernation In Memory Threshold: 1000

Buttons: Help, Ok, Cancel

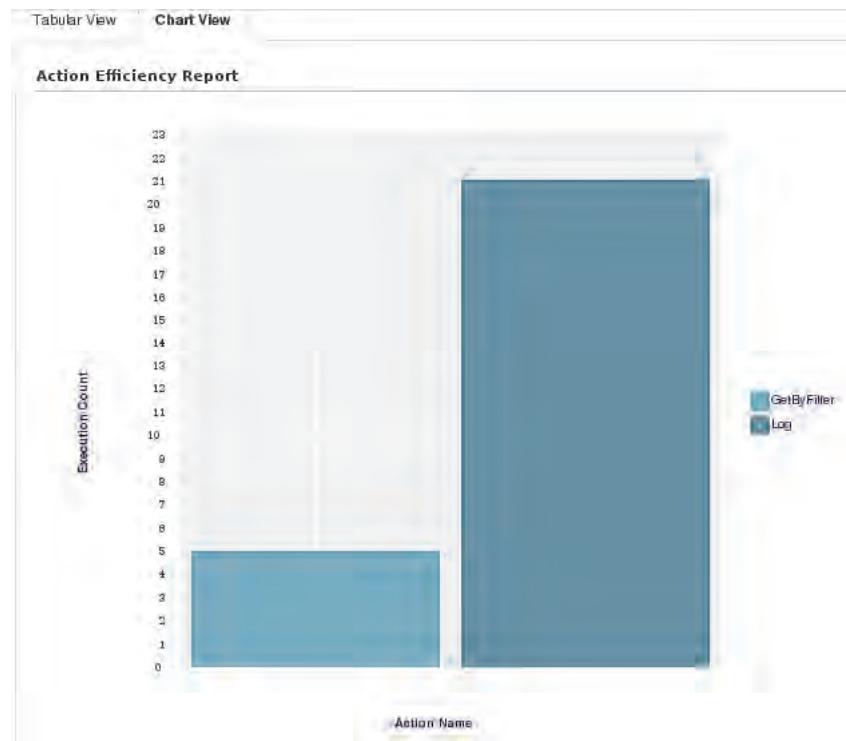
## Exercise 3 Creating an Action Efficiency report

The Action Efficiency report shows the total number of actions that were processed over a selected time range. You use this report to determine how many actions the Impact server performed and which actions are used most often.

1. Select the **Action Efficiency Report**. Set the **Start** and **End** dates to include yesterday and tomorrow. Click the green arrow to run the report.



The tabular view and the chart view shows how many times each action has been performed by policies in the Impact server.



2. Click the **Tabular View** tab.

The tabular view contains a table that shows how many times an action was run and the average time it took to process the action.

A table titled "Action Efficiency Report" showing the execution count and average time for two actions: "Log" and "GetByFilter".

Action	Average Time	Execution Count
GetByFilter	0	5
Log	0	21

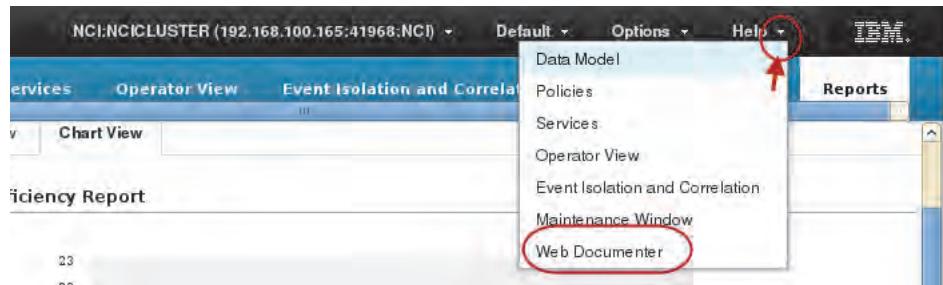
Pages: 1 of 1 Rows: 25 Total: 2

**Note:** The policies have Average Times equal to 0 because they run very quickly, and use very little resource.

# Exercise 4 Viewing the configuration documenter contents

In Netcool/Impact v7 the **Configuration Documenter** is also referred to as the **Web Documenter**, and is accessible from the **Help** menu.

1. Two options are available to access the tool:
  - a. Open in the GUI. Click **Help** and select **Web documenter** in the list to start the tool.



The Configuration Documenter pane is displayed.

A screenshot of the Configuration Documenter web interface. The top navigation bar includes 'IBM Tivoli Netcool/Impact 7.1.0.5', 'Refresh', and the 'IBM' logo. Below the header, it says 'Configuration Documenter' and 'Server Name: NCI Version: 7.1.0.5 (201603221218) Generated on: Wed Jun 22 20:30:01 UTC 2016'. A toolbar at the top right includes links for '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**

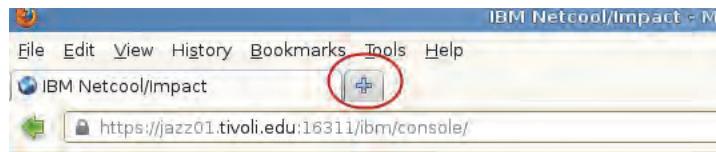
NCI

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

- b. Minimize the **Configuration Documenter** window. To open in a separate window. Click the plus sign.



2. Cut and paste or type the URL into the **New Tab**. Click the green arrow next to the URL to display the Configuration Documenter page.

<https://jazz01.tivoli.edu:16311/documenter/WebDoc.jsp?clusterName=NCICLUSTER>

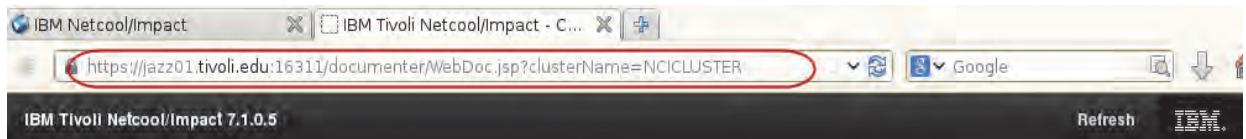
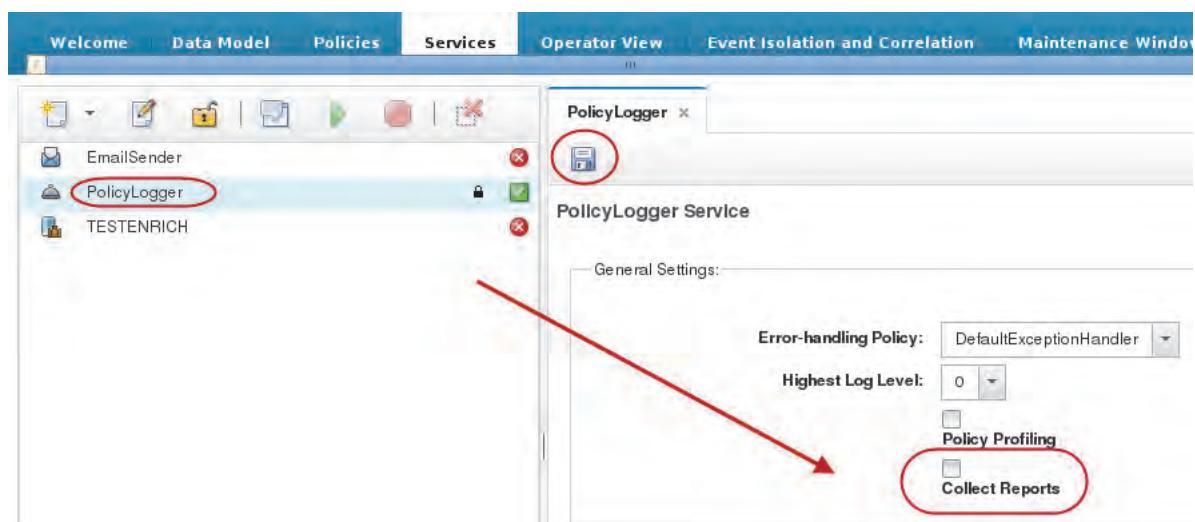


Figure 1 Data Model selection in earlier releases

## Exercise 5 Disable reporting

Collecting data for reports can be resource intensive. Turn off reporting when reports are not required.

1. Click the **Services** tab to list all services.
2. Edit the **PolicyLogger** service
3. Delete the **Collect Reports** check mark to disable reporting.
4. Click the **Save Service** icon.





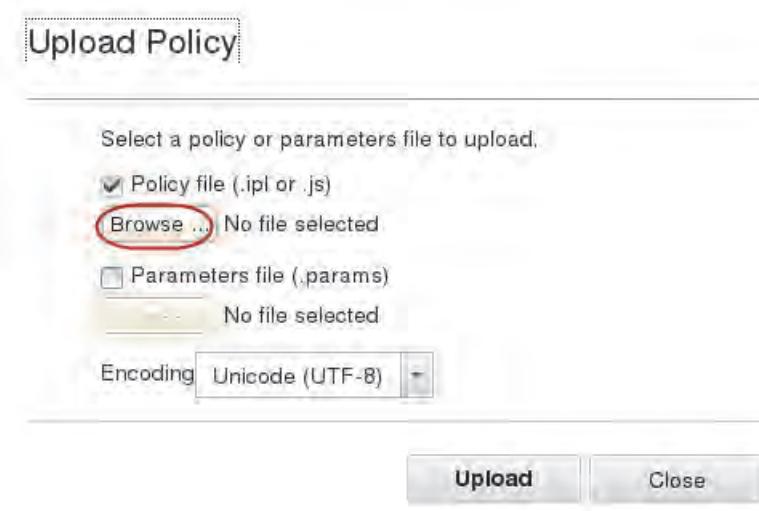
# Unit 11 Operator views exercises

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.

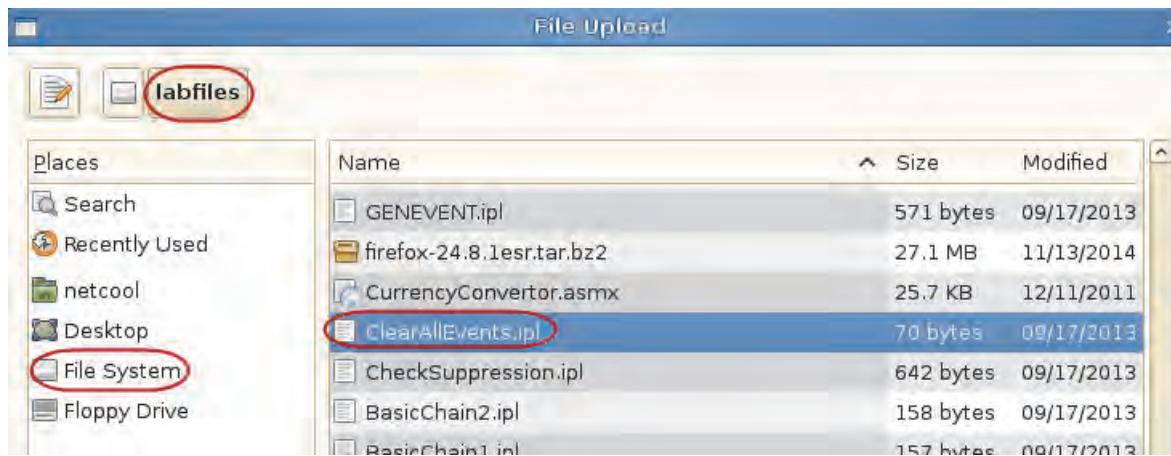
## Exercise 1 Creating a basic operator view

### Uploading the policy for the Action panel

1. Click **Policies** to open the **Policies** tab. From the policy menu, click the **Upload a Policy File** icon .
2. The **Upload Policy** box opens. Click **Browse**.



3. Locate the /labfiles/ClearAllEvents.ipl file.



4. With **ClearAllEvents.ipl** in the policy file window, click **Upload** to import the policy.

### Upload Policy

Select a policy or parameters file to upload.

Policy file (.ipl or .js)

File to upload: ClearAllEvents.ipl

Parameters file (.params)

No file selected

Encoding: Unicode (UTF-8)

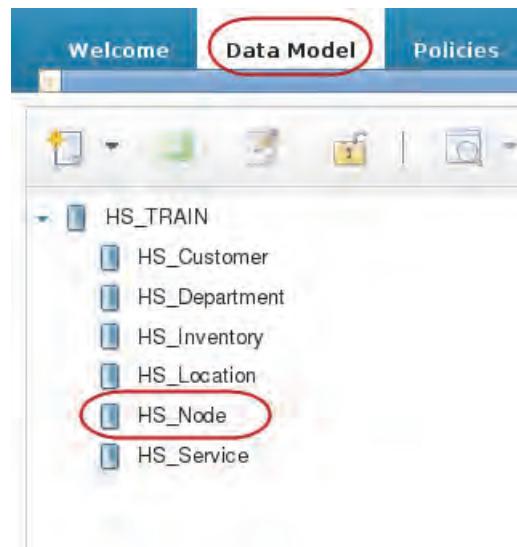
The **ClearAllEvents** policy uses the Netcool/Impact DirectSQL function so that Netcool/Impact policies issue SQL statements against a data source. The policy deletes all events in the ObjectServer.

```
DirectSQL("defaultobjectserver", "delete from alerts.status;", false);
```

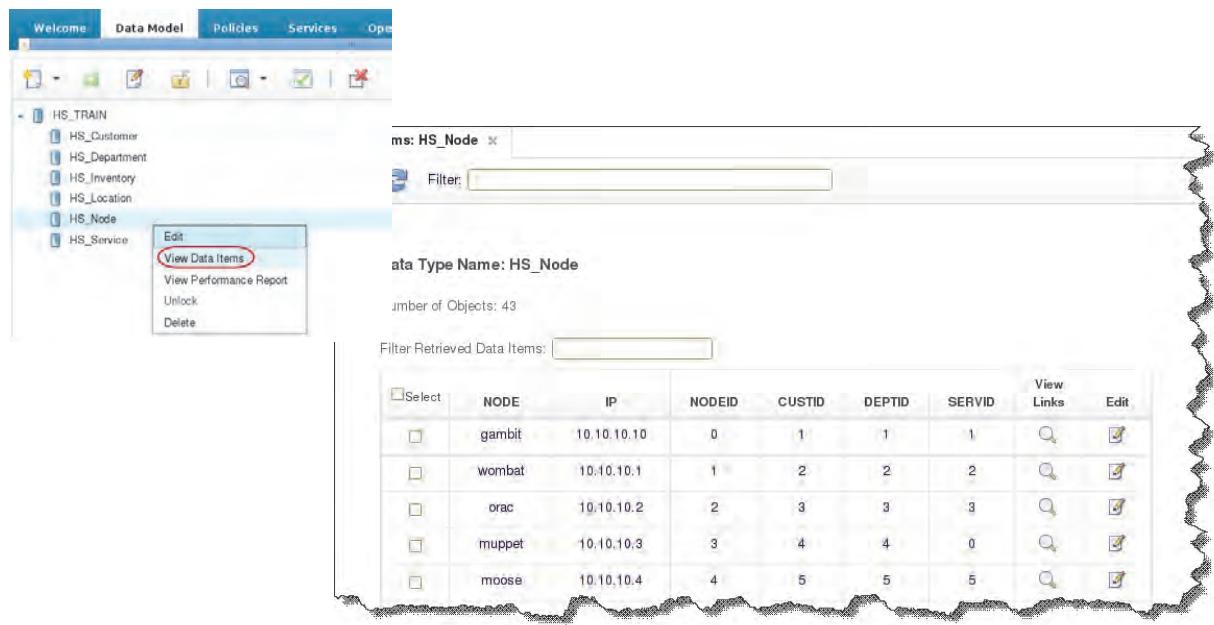
# Creating and configuring the data source

Create a data source definition for the database that holds the sample data for the operator view. The sample data was loaded into the Derby database. You created the **HS\_TRAIN** data source and the **HS\_Node** data type earlier.

1. Verify that **HS\_Node** is listed under the data source **HS\_TRAIN** in the **Data Model** tab.



2. Verify the data type definition by right-clicking **HS\_Node** and selecting **View Data Items**.



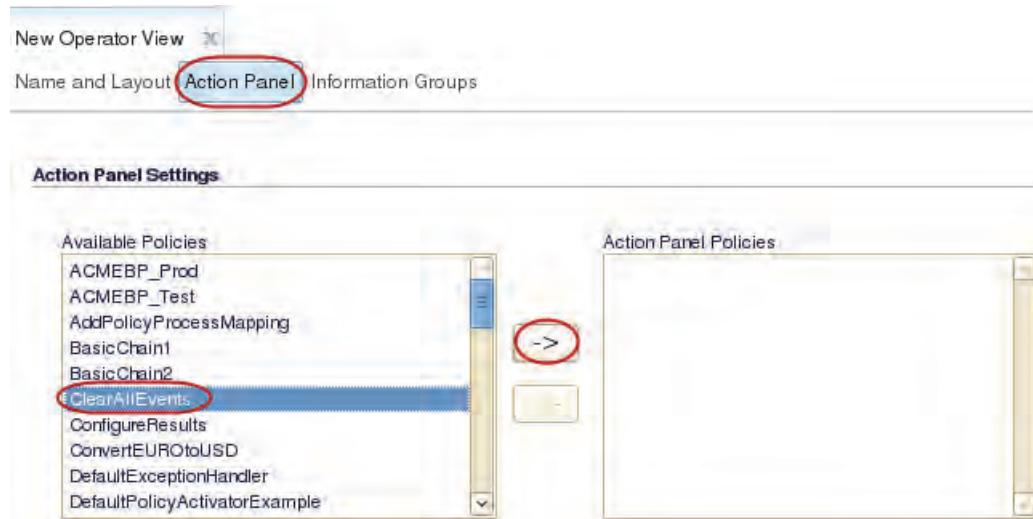
# Creating the operator view

Create the basic operator view.

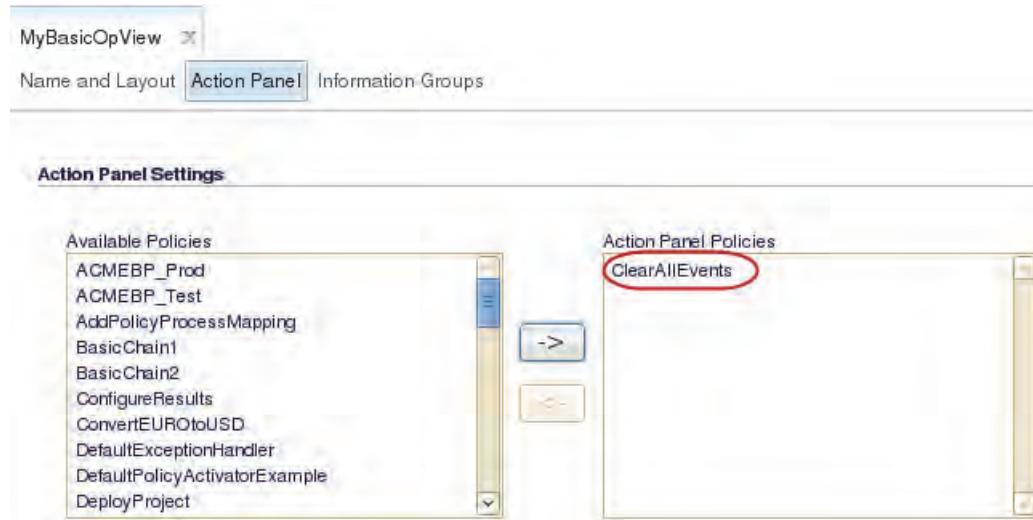
1. Click **Operator Views** to display the **Operator View** tab. Click the **New Operator View** icon.
2. Configure the new operator view as follows:
  - Operator View Name: **MyBasicOpView**
  - Event Panel: **Off**



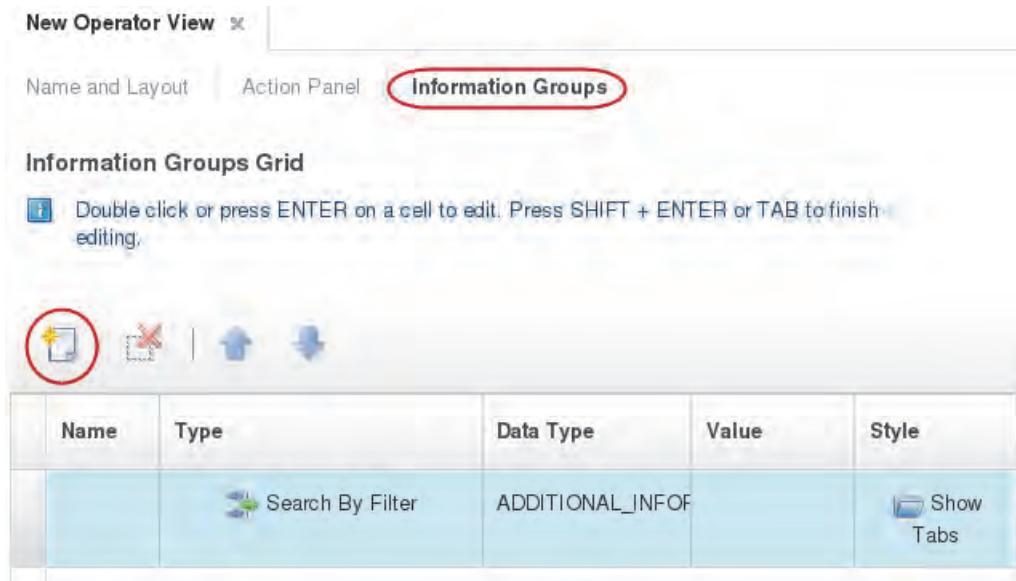
3. Click the **Action Panel** tab and move **ClearAllEvents** from the **Available Policies** list to the **Action Panel Policies** list.



The resulting Action Panel Policies window opens.



4. Click the **Information Groups** tab. Click the **New Information Group** icon.



The screenshot shows the 'New Operator View' window. The 'Information Groups' tab is selected, indicated by a red oval around the tab name. Below the tabs, there is a message: 'Double click or press ENTER on a cell to edit. Press SHIFT + ENTER or TAB to finish editing.' At the top of the grid, there are several icons: a blue folder with a plus sign (highlighted with a red oval), a red delete icon, and two blue navigation arrows. The grid itself has columns labeled 'Name', 'Type', 'Data Type', 'Value', and 'Style'. One row is visible, showing 'Search By Filter' under 'Type', 'ADDITIONAL\_INFOF' under 'Data Type', and a blue folder icon with the text 'Show Tabs' under 'Style'.

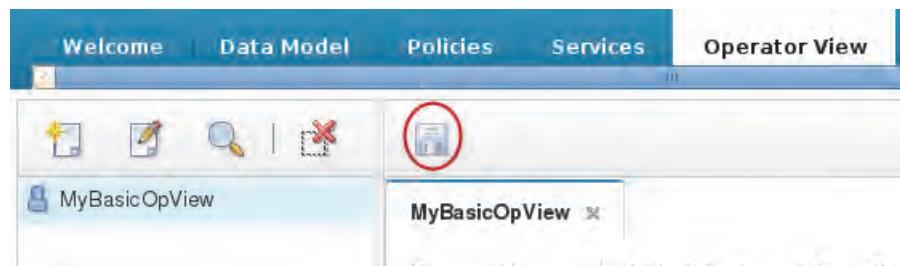
Name	Type	Data Type	Value	Style
	Search By Filter	ADDITIONAL_INFOF		Show Tabs

5. Specify the following information by double-clicking the dark blue bar under group name, type, data type, and style to edit the entries:
- Group Name: **AllNodes**
  - Type: **Filter**
  - Data Type: **HS\_Node**
  - Style: **Table**

The results are like the following example.

The screenshot shows the 'New Operator View' interface with three tabs: 'Name and Layout', 'Action Panel', and 'Information Groups'. The 'Information Groups' tab is selected. A table lists an 'AllNodes' group with a type of 'By Filter'. A dropdown menu next to 'Type' shows options: 'By Filter' (selected), 'By Filter', and 'By Key'. To the right of the table is a list of objects, with 'HS\_Node' highlighted. Below the table, the 'Style' dropdown is set to 'Table'. The 'Value' column for the 'AllNodes' group is set to 'HS\_Node'.

6. Click the icon to save the operator view definition.



7. Click the **View Operator View Display** (magnifying glass) icon to display the operator view data.

The operator view **MyBasicOpView** opens as shown:

The screenshot shows the 'Operator View: MyBasicOpView' window. At the top is a blue header bar. Below it is a dark blue 'Actions' bar containing a single button labeled 'ClearAllEvents'. The main area is titled 'Information Groups' and contains a table with the heading 'AllNodes'. The table has columns: NODE, IP, NODEID, CUSTID, DEPTID, and SERVID. The data rows are as follows:

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
angel	10.10.10.8	8	0	9	2
link1	10.10.10.9	9	0	10	0
link2	10.10.10.10	10	9	0	4
link3	10.10.10.11	11	10	0	5
link4	10.10.10.12	12	1	1	0
link5	10.10.10.13	13	2	2	7
link5	10.10.10.14	14	3	3	1
link6	10.10.10.15	15	4	4	2
batman	10.10.10.16	16	4	4	0
robin	10.10.10.17	17	5	5	4
twoface	10.10.10.18	18	6	6	5
joker	10.10.10.19	19	7	7	6
alfred	10.10.10.20	20	8	8	0

**Note:** You can also view the operator view using this URL:  
<https://jazz01.tivoli.edu:16311/opview/displays/NCICLUSTER-MyBasicOpView.html>

8. On the Linux desktop, click the **GNOME Terminal** icon.



9. Go to the Netcool/OMNIbus **bin** directory by typing the following command:

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

10. Verify that Netcool/OMNIbus is started with the **nco\_ping** command:

```
./nco_ping NCOMS &
```



```
netcool@jazz01:~> cd /opt/IBM/tivoli/netcool/omnibus/bin
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ./nco_ping NCOMS
NCO_PING: Server available.
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin>
```

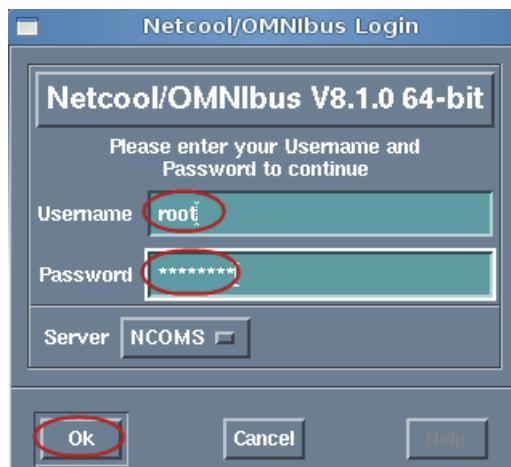
If the server is unavailable, start it with the following command:

```
./nco_objserv &
```

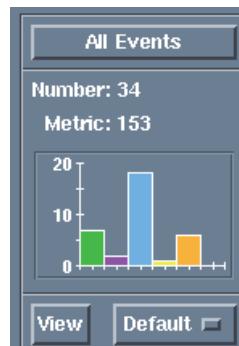
11. Verify that there are existing alerts. Open the Netcool/OMNIbus event list with the following command:

```
./nco_event &
```

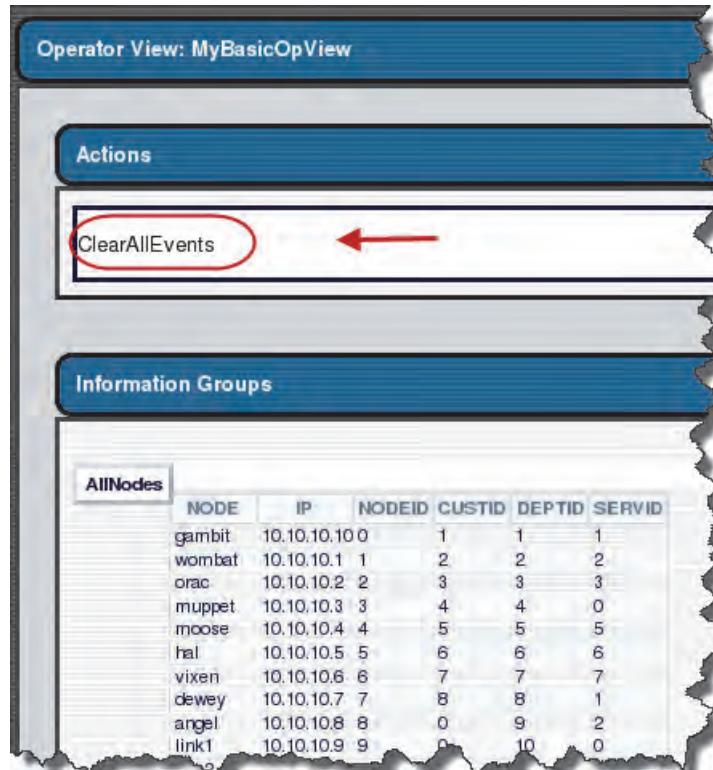
12. When prompted, type **root** as the user and **object00** for the password. Click **Ok**.



13. Expand the screen to show the alert group **All Events** to view the existing events.



14. To test the **Actions** link in the operator view, click **ClearAllEvents** to clear all Netcool/OMNibus events.



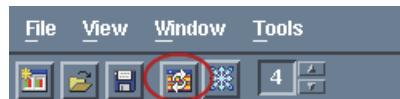
15. Click **Ok** to close the window.

Operator View:

'ClearAllEvents'  
executed  
successfully!

Ok

16. Verify that this action occurred in the Netcool/OMNibus **Event Viewer**. Be sure that you refresh the view in Netcool/OMNibus.



17. From a terminal session, make sure the Simnet Probe is not running to see all the events clear.

- a. To stop the simnet process if it is running, type the commands as shown here.

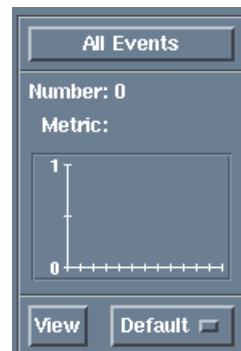
```
ps -ef|grep nco_p_simnet  
kill -9 11503
```



**Note:** Use the process id on your system to replace 11503.

```
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ps -ef|grep nco_p_simnet
netcool 11503 1 0 18:57 ? 00:00:01 /opt/IBM/tivoli/netcool/omnibus/
platform/linux2x86/probes64/nco_p_simnet
netcool 13443 12923 0 19:12 pts/2 00:00:00 grep nco_p_simnet
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> kill -9 11503
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ps -ef|grep nco_p_simnet
netcool 13504 12923 0 19:12 pts/2 00:00:00 grep nco_p_simnet
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin>
```

- b. Click **ClearAllEvents** in the **Operator View**. Refresh **All Events** in OMNIbus. The resulting view in OMNIbus is as shown in the following screen capture:

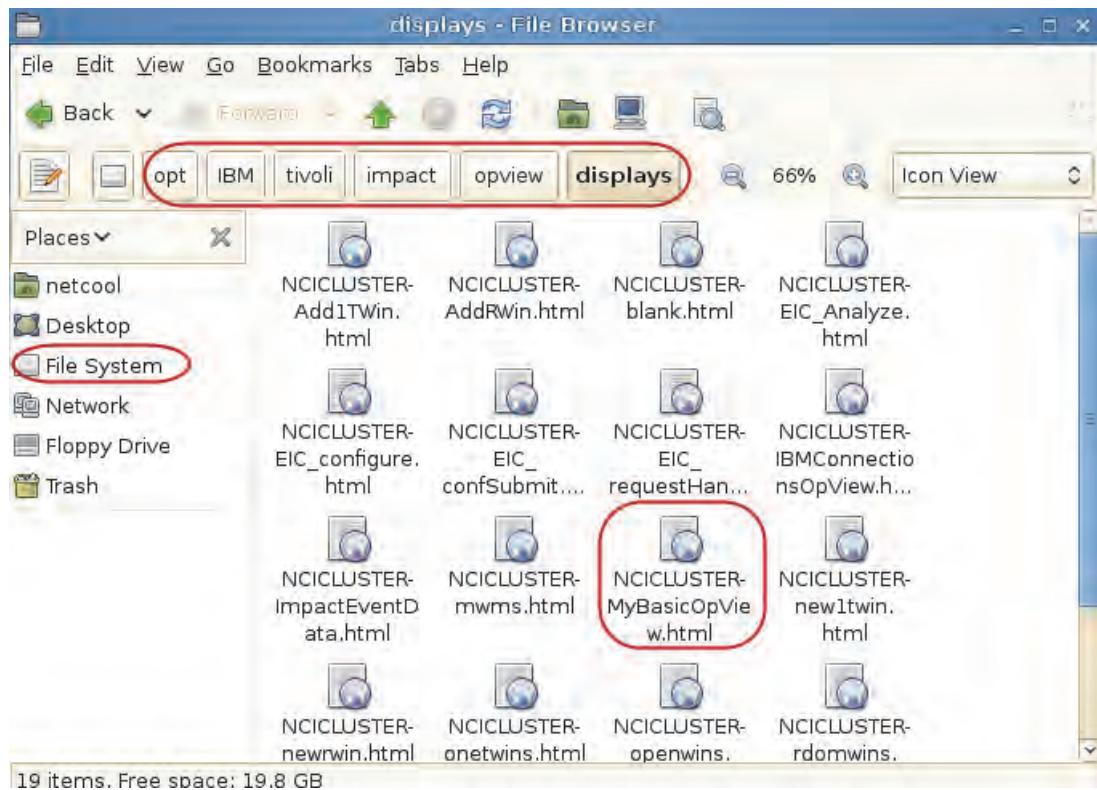


After the operator view is created, a policy is automatically created with the name **OpView\_MyBasicOpview**. A corresponding HTML file called **NCICLUSTER-MyBasicOpView.html** is created in **\$NCHOME/opview/displays**.

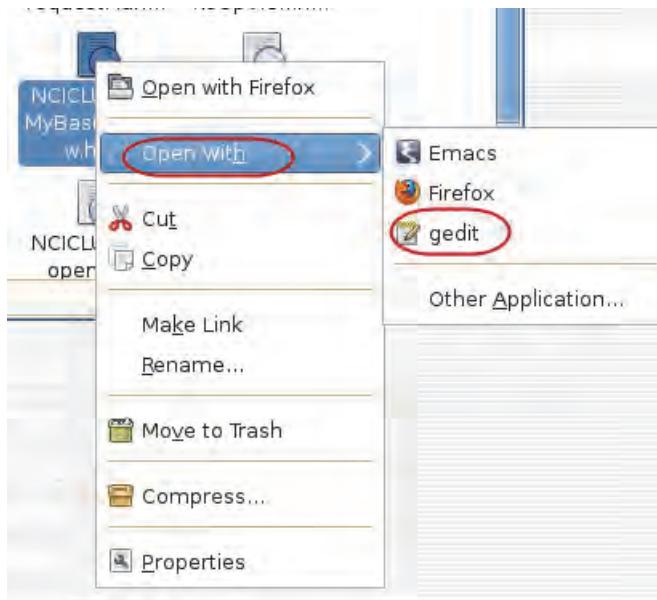
18. Double click the icon shown below. Examine the **NCICLUSTER-MyBasicOpView.html** file in the following directory:

**/opt/IBM/tivoli/impact/opview/displays**



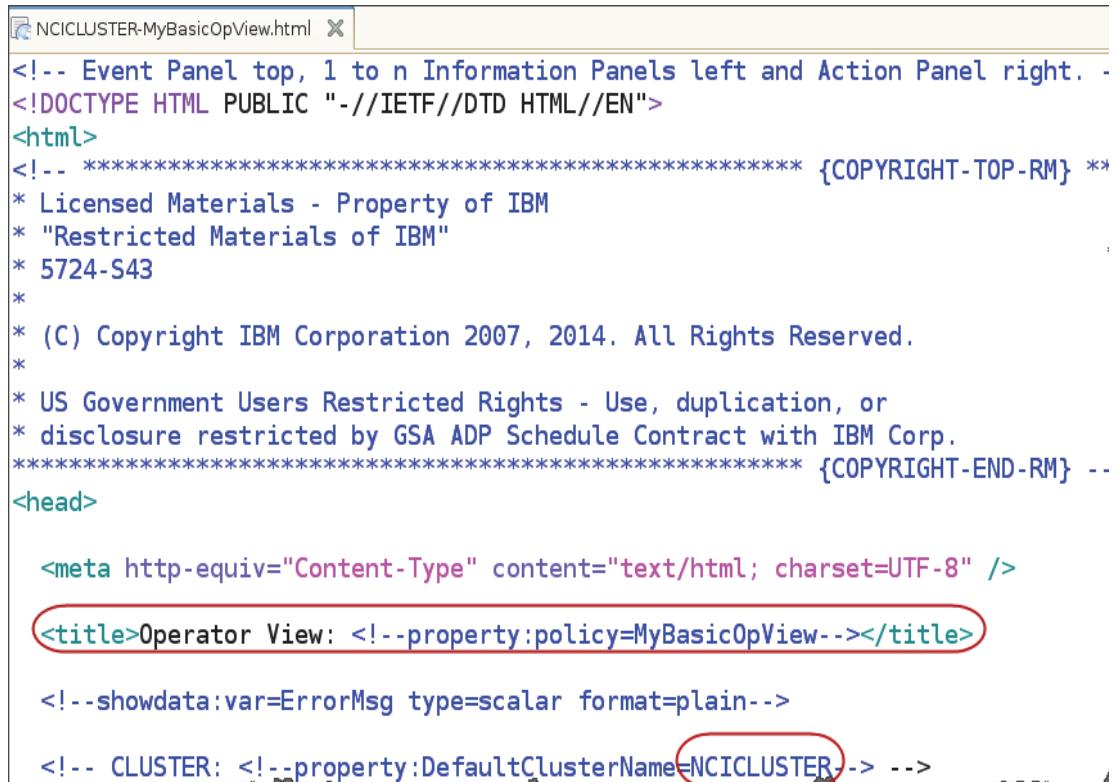


19. Right-click the file icon and select **gedit** to edit the file.



Locate the operator view smart tags that are responsible for the appearance and content of the newly created **MyBasicOpView**.

Note the two property smart tags: one to define the Netcool/Impact policy behind the web page and one to identify the Netcool/Impact cluster. The smart tags are HTML comments.



```
<!-- 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--> -->
```

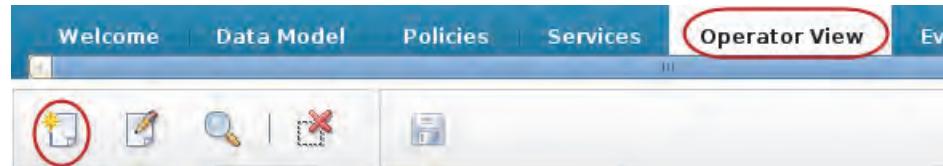
# Exercise 2 Creating an advanced (custom) operator view

Netcool/Impact operator views have two primary components: *a policy* (or policies) that retrieve data fields from one or more sources, and *an HTML page*. When the HTML page is opened, it starts the Netcool/Impact policy. The policy contains the required data fields, which reference smart tags in the HTML page.

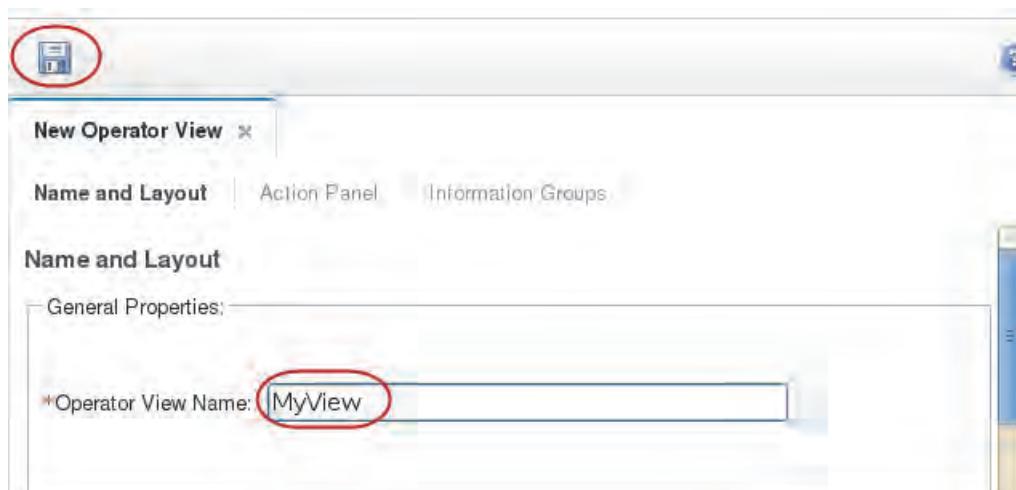
This document does not cover the detailed mechanics of HTML pages. The exercise is designed to introduce the process required to produce an advanced operator view.

## Creating the base operator views

1. Click **Operator Views** to display the **Operator View** tab. Click the **New Operator View** icon.



2. Create the first operator view and name it **MyView**. Save it without any customization.



3. When you save the operator view, a policy named **Opview\_MyView** is automatically generated and is displayed in the list of policies. Select the new policy in the list to display it in the policy editor, as shown in the following figure.

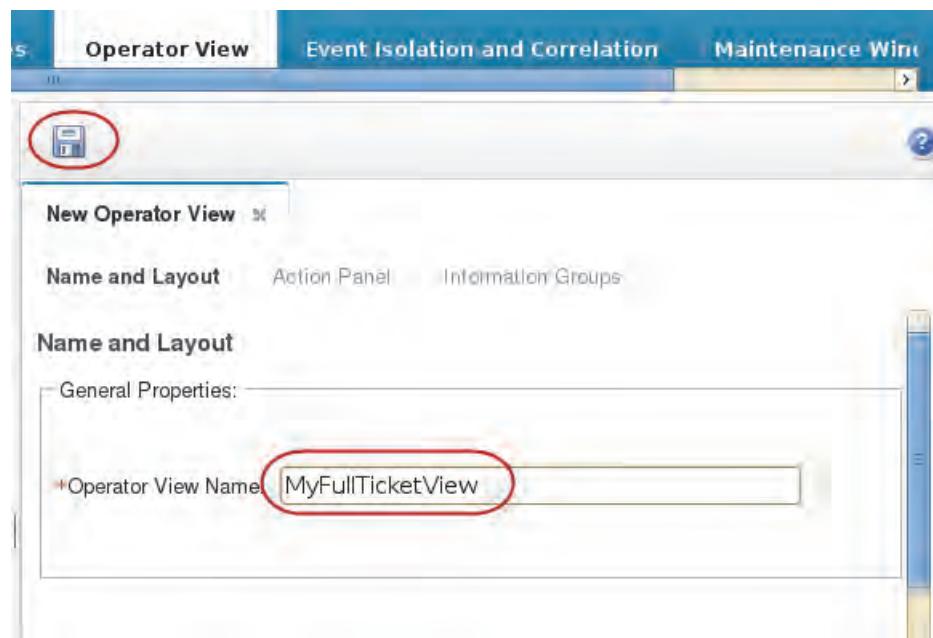


**Note:** If the operator view policy does not open, refresh the browser.

The screenshot shows the IBM Netcool/Impact interface in Mozilla Firefox. The title bar reads "IBM Netcool/Impact - Mozilla Firefox". The menu bar includes File, Edit, View, History, Bookmarks, Tools, and Help. The toolbar has icons for Back, Forward, Stop, Home, and Search. The address bar shows the URL "https://jazz01.tivoli.edu:16311/ibm/console/". The main window has a navigation bar with tabs: Welcome, Data Model, Policies, Services, Operator View, Event Isolation and Correlation, and Maintenance Window. The "Policies" tab is selected. On the left, a sidebar lists policies: BasicChain1, BasicChain2, ClearAllEvents, GENEVENT, GetPartID, Opview\_MyBasicOpView, Opview\_MyView (circled in red), POLICY-EventEnrichment-Adv, POLICY-EventEnrichment-Basic, and SendEmailTest. The main content area displays the "Opview\_MyView" policy code. The code starts with a note: // This policy generated by Impact Operator View. It then says // Modify at your own risk! // Once modified, this policy may no longer be configurable through the Impact Operator View GUI. Below this is a section // INFO PANEL - START followed by // each smart tag on display page should have corresponding // \_OVUpdateTAGVARNAME() function, where TAGVARNAME is the var property of the. At the bottom, it says // checks for a list of tags to update, by ajax\_tags parameter // sent through EventContainer.

In addition to the policy, an HTML page is also automatically generated. This page is located in **/opt/IBM/tivoli/impact/opview/displays** and is named **NCICLUSTER-MyView.html**.

4. Create another operator view called **MyFullTicketView**. Save it without customization.



5. When you save it, another policy is automatically generated, and it shows in the list of policies as **Opview\_MyFullTicketView**. The corresponding HTML file is also created.

## Sample database

The data that is used for this operator view exercise is already loaded into the Derby database. These four tables are required for this lab exercise:

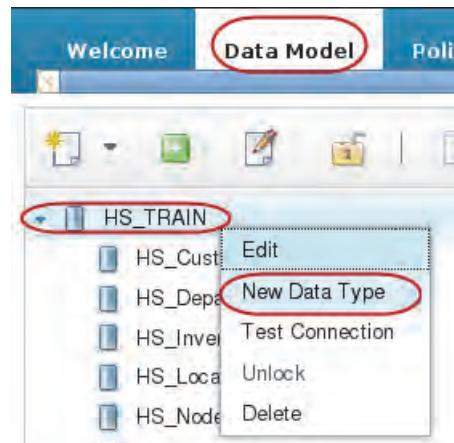
- ASSET
- SERVAPPS
- SERVINFO
- TICKET

## Creating the data source and data types required by the policies

The policies that are used in the operator views require the following data source and data type definitions:

- Data source

Because the sample tables were installed in the Derby database and there is already a data source definition for it, you can use the same data source, **HS\_TRAIN**.



- Data types

You need a data type definition for each of the four sample tables mentioned previously.

1. Create four data types under the data source **HS\_TRAIN** using **Schema: IMPACT**. Use the following information for other specifications. Remember to click **Refresh** to see the fields in the table. Don't forget to save your changes.

Data Type Name	Key field	Table
OV_ASSET	DEVICE	ASSET
OV_SERVAPPS	SERVER	SERVAPPS
OV_SERVINFO	SNAME	SERVINFO
OV_TICKET	TICKETNO	TICKET

The resulting data model is like the following examples:

### OV\_ASSET

ID	Field Name	Format	Display Name	Description	Key Field
<input type="checkbox"/>	DEVICE	STRING	DEVICE	DEVICE	<input checked="" type="checkbox"/> Yes
<input type="checkbox"/>	CUSTOMER	STRING	CUSTOMER	CUSTOMER	No
<input type="checkbox"/>	CUSTCONTACT	STRING	CUSTCONTACT	CUSTCONTACT	No
<input type="checkbox"/>	CCPHONE	STRING	CCPHONE	CCPHONE	No
<input type="checkbox"/>	ASSET	STRING	ASSET	ASSET	

### OV\_SERVAPPS

ID	Field Name	Format	Display Name	Description	Key Field
<input type="checkbox"/>	SERVER	STRING	SERVER	SERVER	<input checked="" type="checkbox"/> Yes
<input type="checkbox"/>	APP	STRING	APP	APP	No
<input type="checkbox"/>	STATUS	STRING	STATUS	STATUS	No
<input type="checkbox"/>	PID	INTEGER	PID	PID	No

## OV\_SERVINFO

**Table Description**

General Settings:

- \* Data Type Name: **OV\_SERVINFO**
- + 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:	Tables:	Completed
fetching schemas	<b>SERVINFO</b>	Completed table

**SQL Data Type Config Page**

Table Description

Dynamic Links Cache Settings

Refresh Fields

Show New / Deleted Fields

Delete Selection New Edit Move Up Move Down

ID	Field Name	Format	Display Name	Description	Key Field
	SNAME	STRING	SNAME	SNAME	<b>Yes</b>
	IP	STRING	IP	IP	No
	MAC	STRING	MAC	MAC	No
	LOCATION	STRING	LOCATION	LOCATION	No

## OV\_TICKET

**Table Description**

General Settings:

- \* Data Type Name: **OV\_TICKET**
- + 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:	Tables:	Completed
fetching schemas	<b>TICKET</b>	Completed

**SQL Data Type Config Page**

Table Description

Dynamic Links Cache Settings

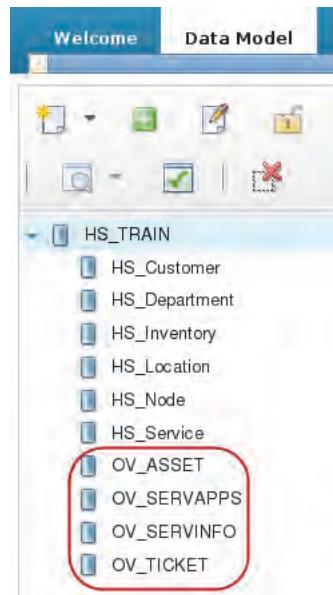
Refresh Fields

Show New / Deleted Fields

Delete Selection New Edit Move Up Move Down

ID	Field Name	Format	Display Name	Description	Key Field
	TICKETNO	STRING	TICKETNO	TICKETNO	<b>Yes</b>
	SHORTDESC	STRING	SHORTDESC	SHORTDESC	No
	LONGDESC	STRING	LONGDESC	LONGDESC	No
	LASTMOD	STRING	LASTMOD	LASTMOD	No
	STATUS	STRING	STATUS	STATUS	No

## Resulting Data Model



## Building the web pages

At the beginning of this exercise, you created two operator views:

- MyView
- MyFullTicketView

The process of generating those operator views created Netcool/Impact policies and HTML pages. Typically, these provide the starting point for the customization that is required to produce the views. That process consists of modifying the HTML pages to include smart tags that reference data that is populated by Netcool/Impact. You revise the policies to retrieve data from various sources (data types) and produce the data used to populate the smart tags.

For this exercise, you replace the base files that were automatically generated by the creation of the two operator views with the corresponding HTML pages, and completed policies in the **labfiles** directory.

## Policies

Two policies are provided. One policy is used to populate the base dashboard. The second policy is used to retrieve ticket detail and to populate an additional area on the dashboard. Each of these policies is provided in Netcool/Impact Policy Language (IPL) format. The easiest way to get these policies into Netcool/Impact is to copy and paste the policy text.

The policies can be found in this location:

```
/labfiles/policies  
    NCI_Opview_OVGuide.ipl  
    NCI_Opview_FullTicket.ipl
```

1. Open the **Opview\_MyView** policy in the Netcool/Impact policy editor. You use this policy to populate the contents of the base dashboard.
2. Delete all the policy contents from the editor window. Click the **Save** icon.



3. In a separate terminal window on the operating system (or upload the policy into the policy editor), edit the policy file, **/labfiles/policies/NCI\_Opview\_OVGuide.ipl**.
4. Select and copy the entire contents of the **NCI\_Opview\_OVGuide.ipl** policy, and paste it into the empty **Opview\_MyView** policy editor window. Save the **Opview\_MyView** policy.

```
// Scalar Tags Date formatting for dashboard  
  
currserver = "impactserver1";  
log("CURRSERVER :" + currserver);  
today = GetDate();  
currdate = String(LocalTime(today, "MMM dd, yyyy KK:mm a"));  
log("CURREDATE :" + currdate);  
  
// Scalar String Tags Server Info Section  
  
servinfo = GetByFilter("****MODIFY****", "sname=" + currserver + "", false);  
sname = servinfo[0].sname;  
ip = servinfo[0].ip;  
mac = servinfo[0].mac;  
location = servinfo[0].location;  
stype = servinfo[0].stype;  
os = servinfo[0].os;  
amtdisk = servinfo[0].amtdisk;  
amtmem = servinfo[0].amtmem;  
processors = servinfo[0].processors;  
  
// List Tags Application Info Section
```

## Modifying the policies

Change the \*\*\*MODIFY\*\*\* entries in the policies as described in the next few steps.

1. Locate the following lines in the policy and change the data type name from \*\*\*MODIFY\*\*\* to the following name:

```
servinfo = GetByFilter('OV_SERVINFO',"sname=''" + currserver + "'",false);

    // Scalar Tags Date formatting for dashboard

    currserver = "impactserver1";
    log("CURRSERVER :" + currserver);
    today = GetDate();
    currdtate = String(LocalTime(today, "MMM dd, yyyy KK:mm a"));
    log("CURRDATE :" + currdtate);

    // Scalar String Tags Server Info Section

    servinfo = GetByFilter('OV_SERVINFO',"sname=''" + currserver + "'",false); ←
    sname = servinfo[0].sname;
    ip = servinfo[0].ip;
    mac = servinfo[0].mac;

apps = GetByFilter('OV_SERVAPPS',"server=''" + currserver + "'",false);

    // List Tags Application Info Section

    apps = GetByFilter('OV_SERVAPPS',"server=''" + currserver + "'",false); ←
    appcount = Length(apps);
    log("APPCOUNT :" + appcount);
    i = 0;

asset = GetByFilter('OV_ASSET',"device =''" + ip + "'",false);

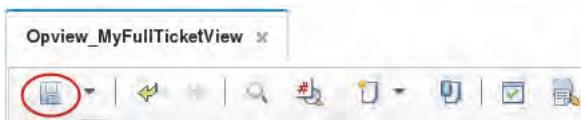
    asset = GetByFilter('OV_ASSET',"device =''" + ip + "'",false); ←
    custemail = asset[0].ccemail;
    owneremail = asset[0].oemail;
    ec_buttitle = "Email Customer";

tickets = GetByFilter('OV_TICKET',"catid =''" + ip + "'",false);

    // OrgNode CustomTable Type Ticket Info View Section

    tickets = GetByFilter('OV_TICKET',"catid =''" + ip + "'",false); ←
```

2. Check the syntax, and save the changes to the policy.
3. Open the **Opview\_MyFullTicketView** policy in the Netcool/Impact policy editor.
4. Delete all of the policy text in the editor window. Click the Save icon.



5. View and edit **/labfiles/policies/NCI\_Opview\_FullTicket.ipl** and copy its contents into the empty policy in the Policy Editor window.

This policy is used to retrieve ticket detail information when you select a **view** button on the dashboard. It contains a data type name that you must change as follows.

```
fulltick = GetByFilter('OV_TICKET',"ticketno =' + @TICKETNO + "'",false);
```

6. Save the changes to the policy, which are like the following example.

```
// Scalar Tags - Full Ticket View Section
log("===== >TICKET_NUMBER:" + @TICKETNO);

fulltick = GetByFilter('OV_TICKET',"ticketno =' + @TICKETNO + "'",false);

ticketno = fulltick[0].ticketno;
shortdesc = fulltick[0].shortdesc;
longdesc = fulltick[0].longdesc;
lastmod = fulltick[0].lastmod;
status = fulltick[0].status;
openedby = fulltick[0].openedby;
assigned = fulltick[0].assigned;
category = fulltick[0].category;
catid = fulltick[0].catid;
```

## HTML pages

The HTML files are located in the **labfiles/html** directory. As was mentioned previously, HTML smart tags are used to place Netcool/Impact policy data into the web page. After completing this lab exercise, time permitting, go back and modify the HTML one panel at a time to see what changes.

In the first step in this exercise, you generated two operator views:

- Opview\_MyView
- Opview\_MyFullTicketView

That process also produced two corresponding HTML files:

```
/opt/IBM/tivoli/impact/opview/displays
    NCICLUSTER-MyView.html
    NCICLUSTER-MyFullTicketView.html
```

In the next step, you replace each of these files with the corresponding file from the **labfiles** directory. The replacement files are in this location:

```
/labfiles/html
NCICLUSTER-OVGuide.html
NCICLUSTER-FullTicket.html
```

- From a command line, execute these commands to copy each file from the **labfiles** directory and rename them:

```
cd /labfiles/html
cp NCICLUSTER-OVGuide.html
/opt/IBM/tivoli/impact/opview/displays/NCICLUSTER-MyView.html
cp NCICLUSTER-FullTicket.html
/opt/IBM/tivoli/impact/opview/displays/NCICLUSTER-MyFullTicketView.html
```

Each of these HTML files requires additional modifications. These changes revise the policy references.

- Use **gedit** to edit **/opt/IBM/tivoli/impact/opview/displays/NCICLUSTER-MyView.html**:

- Locate the following lines at the top of this file:

```
<!--
<!--property:policy=***MODIFY***-->
<!--property:DefaultClusterName=NCICLUSTER-->
-->
```

- Change the policy name reference:

```
<!--property:policy=MyView-->
```

- Locate the following line:

```
action_policy="***MODIFY***"
```

- Change as follows:

```
action_policy="MyFullTicketView"
```

- Save the changes.

- Edit **/opt/IBM/tivoli/impact/opview/displays/NCICLUSTER-MyFullTicketView.html**.

- Locate the following line near the top of this file:

```
<!--property:policy=***MODIFY***-->
```

- Change as follows:

```
<!--property:policy=MyFullTicketView-->
```

- Save the changes.

# Viewing the advanced operator view

The advanced operator view configuration is complete and ready to test.

1. Click **Operator View** to display the **Operator View** tab.

Highlight and right-click **MyView**. Select **View**.

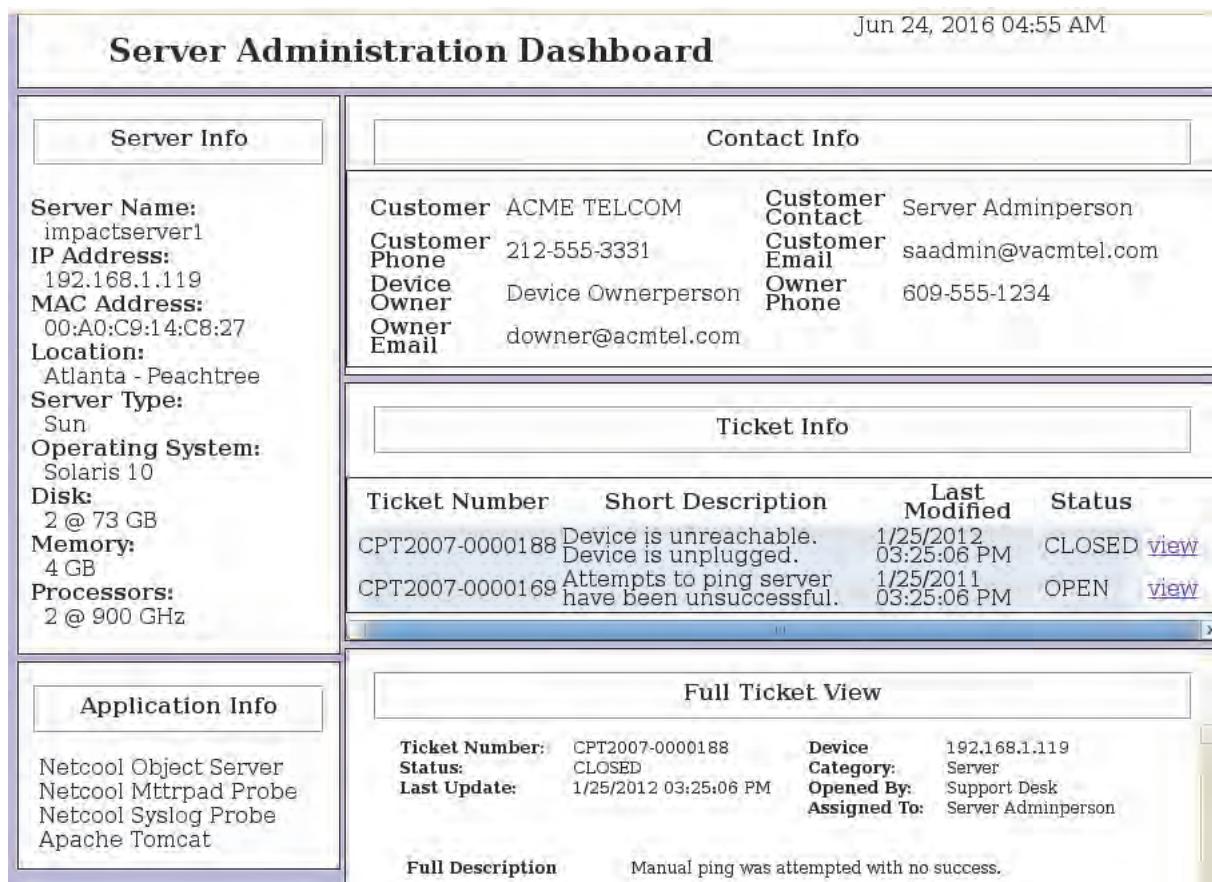


The operator view is also accessible from a browser.

2. Type the following URL:

<https://jazz01.tivoli.edu:16311/opview/displays/NCICLUSTER-MyView.html>

3. Click the **view** link to the right of the ticket number to display the **Full Ticket View**. The results are similar to the following example.



The screenshot shows the Server Administration Dashboard with the following sections:

- Server Info:**
  - Server Name: impactserver1
  - IP Address: 192.168.1.119
  - MAC Address: 00:A0:C9:14:C8:27
  - Location: Atlanta - Peachtree
  - Server Type: Sun
  - Operating System: Solaris 10
  - Disk: 2 @ 73 GB
  - Memory: 4 GB
  - Processors: 2 @ 900 GHz
- Contact Info:**

Customer	ACME TELCOM	Customer	Server Adminperson
Customer Phone	212-555-3331	Customer Email	saadmin@vacmtel.com
Device Owner	Device Ownerperson	Owner Phone	609-555-1234
Owner Email	downer@acmtel.com		
- Ticket Info:**

Ticket Number	Short Description	Last Modified	Status
CPT2007-0000188	Device is unreachable. Device is unplugged.	1/25/2012 03:25:06 PM	CLOSED <a href="#">view</a>
CPT2007-0000169	Attempts to ping server have been unsuccessful.	1/25/2011 03:25:06 PM	OPEN <a href="#">view</a>
- Application Info:**
  - Netcool Object Server
  - Netcool Mttrpad Probe
  - Netcool Syslog Probe
  - Apache Tomcat
- Full Ticket View:**

Ticket Number:	CPT2007-0000188	Device	192.168.1.119
Status:	CLOSED	Category:	Server
Last Update:	1/25/2012 03:25:06 PM	Opened By:	Support Desk
		Assigned To:	Server Adminperson

Full Description: Manual ping was attempted with no success.

When the window initially opens, the ticket detail in the lower portion of the dashboard is empty. Selecting the **view** action link shows the corresponding detail in the **Full Ticket View** panel.

This operator view page starts a policy that retrieves data for a specific device, **impactserver1**. A review of the **MyView** policy shows that the server name value was hard coded as **currserver = "impactserver1";**. You can revise the policy to replace the **currserver** constant with a variable. You can then use the variable to launch the operator view in context.

The **MyView** HTML file refers to the **Opview\_MyView** Netcool/Impact policy. That policy retrieves data from four data tables and passes selected fields into the HTML page with smart tag references. The **view** link to the right of each ticket on the dashboard opens the **MyFullTicketView** HTML page for that particular ticket. This page references the **Opview\_MyFullTicketView** policy. That policy retrieves detail data regarding the corresponding ticket. Those details are passed back to the HTML page using smart tag references.

View the **MyView HTML** file to examine the different smart tags that are associated with each of the panels of the finished main operator view page.

# **Unit 12 Working with web services exercises**

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

The Web Services wizard performs the following actions:

- Parses the WSDL file
- Creates the necessary JAR package
- Determines the available methods and parameters for making requests against the web services application
- Dynamically loads the Java classes into the Netcool/Impact server without requiring a restart of the Netcool/Impact server

The resulting policy contains the necessary parameters to invoke the user-selected method, invokes the method, waits for the response, and stores the returned information, which is now ready for additional processing by the policy.

## **Exercise 1 Creating a policy using the Web Services wizard**

In this exercise, you build a policy to request a web services application to determine the conversion rate from Euro to US dollar. Typically, an Internet connection is required to test policies accessing public web services. However, the Internet will not be required for this exercise. You build a sample return result in order to examine web services, and Jazz for Service Management components.

### **Building the Web Service policy**

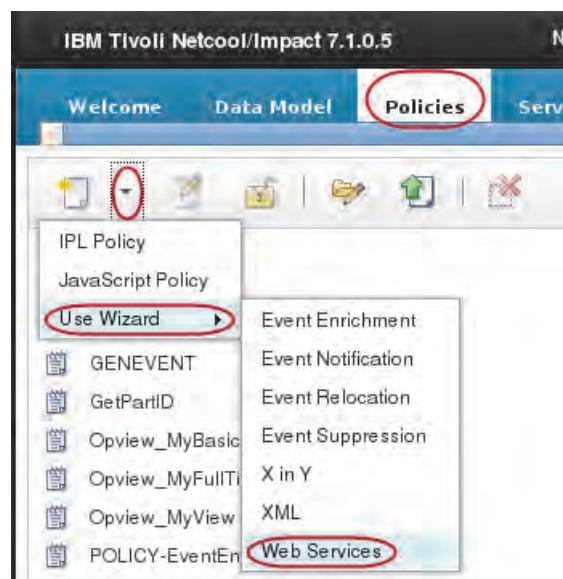


**Note:** The Netcool/Impact URL is <https://jazz01.tivoli.edu:16311/ibm/console/>.

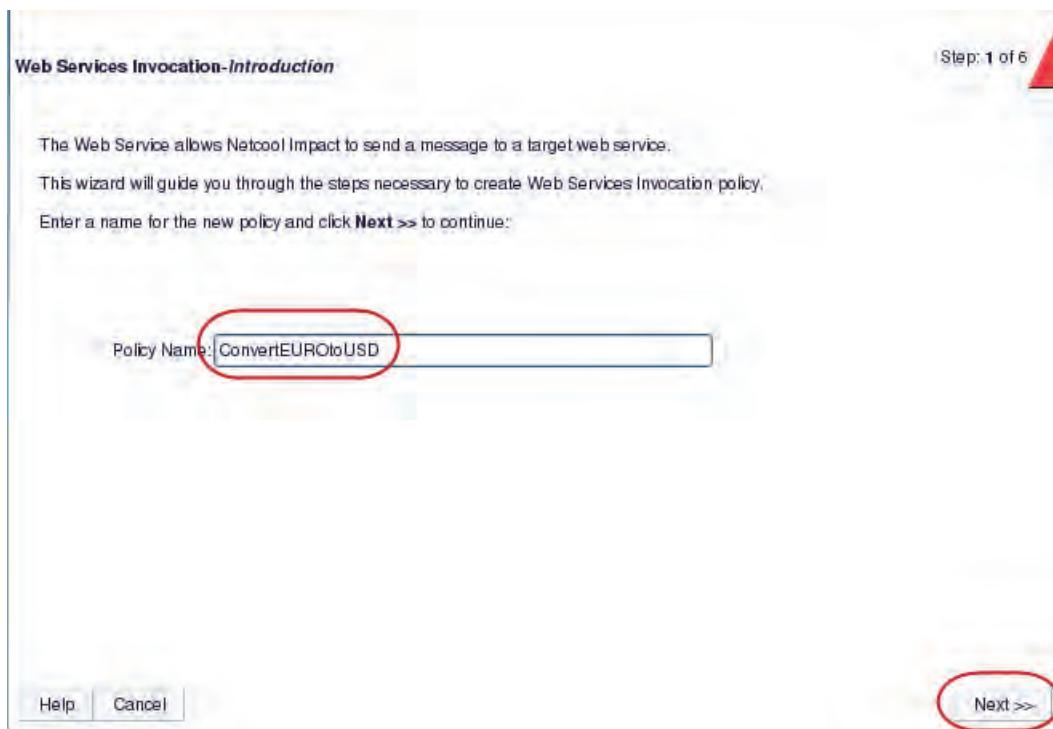
1. Log in as **impactadmin** with password **object00**. Click **Go**.



- a. Click **Policies** to display the policy editor screen.
- b. Click the **New Policy** icon . Scroll to **Use Wizard**.
- c. Click **Web Services**.



- At the **Web Services Invocation** window, enter the policy name **ConvertEUROtoUSD**. Click **Next**.



- Follow these steps for the WSDL File and Client Stub configuration:
  - At Step 2 under **WSDL File**, specify the URL or Path to WSDL as **/labfiles/CurrencyConvertor.asmx**.  
This file was downloaded to the local disk from the URL <http://www.webservicex.net/CurrencyConvertor.asmx?WSDL>.  
If the system has Internet connection, you can specify this URL at this step in the policy creation instead of pointing to the local copy.
  - Under **Client Stub**, select **Provide a package name for the new client stub**. Give it a package name of **CurrencyConvertor**. This name is the name of the JAR file that is built to

support the Web Services API for this particular **CurrencyConvertor** application. Click **Next**.

Web Services Invocation-WSDL File and Jar File Step: 2 of 6

**WSDL File**

URL or Path to WSDL: /labfiles/CurrencyConvertor.asmx

**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:

Package Name:  Edit

Provide a package name for the new jar file:

Package Name: CurrencyConvertor

Help Cancel << Back Next >>

- At Step 3 in the GUI, accept the default values for the **Web Service**, **Web Service Port Type**, and **Web Service Method**. Click **Next**.

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: CurrencyConvertor

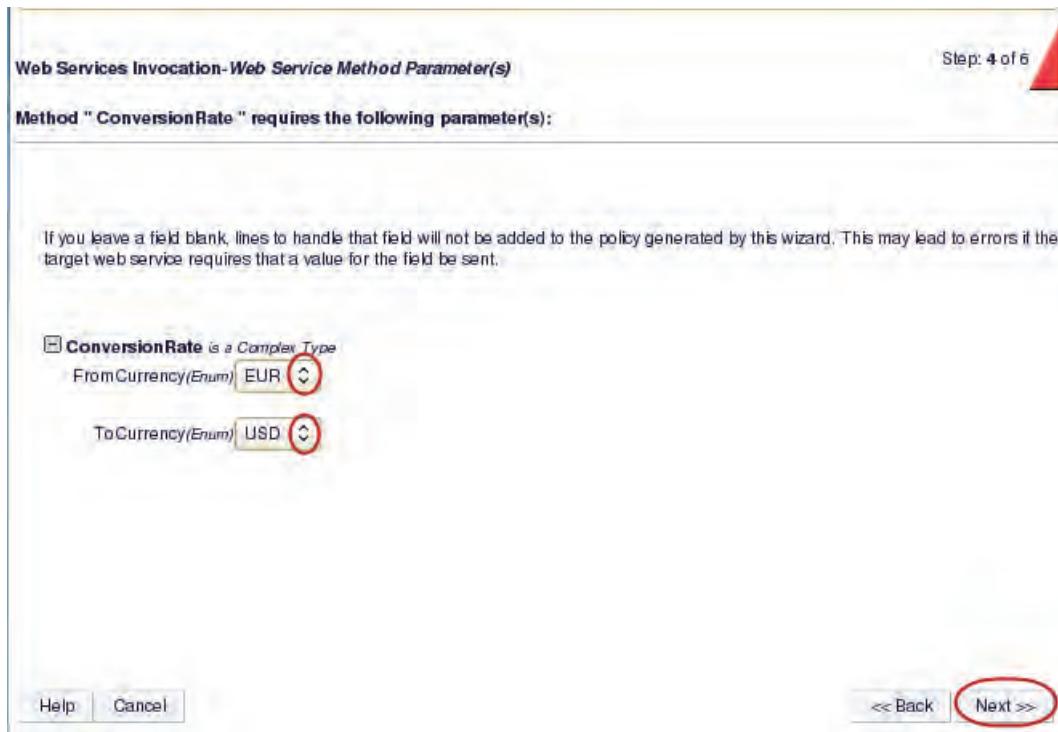
Web Service Port Type: CurrencyConvertorSoap

Web Service Method: ConversionRate

Click **Next** will compile the WSDL and add client jar to Impact. This will take several seconds.

Help Cancel << Back Next >>

- At Step 4 in the GUI, expand **ConversionRate** and specify the **FromCurrency** and **ToCurrency** values. The hard-coded choices, which are taken from the WSDL file, are presented in the drop-down menus. Click **Next**.



- At Step 5 in the GUI, note the **End Point URL**, the access point of the target SOAP interface.

Do not change this entry. It was derived from the WSDL file that you designated in Step 2. Do not select **Enable web service security**. Click **Next**.

https://jazz01.tivoli.edu:16311/impactAdmin/WizardFramesServlet?wizardstepinfo=com.micros... Step: 5 of 6

**Web Services Invocation - Web Service End Point And Security**

**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:

(Red circle)

7. At the Summary and Finish screen, click **Finish**.

https://jazz01.tivoli.edu:16311/impactAdmin/WizardFramesServlet?wizardstepinfo=com.micros... Step: 6 of 6

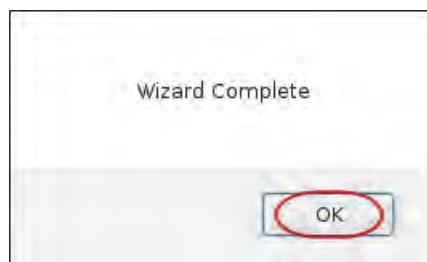
**Web Services Invocation - Summary and Finish**

The wizard will create the following policy when you click Finish. 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. It may be useful to note this information down, to help you find them later for modifications and maintenance.

Policy Name: ConvertEUROtoUSD

(Red circle)

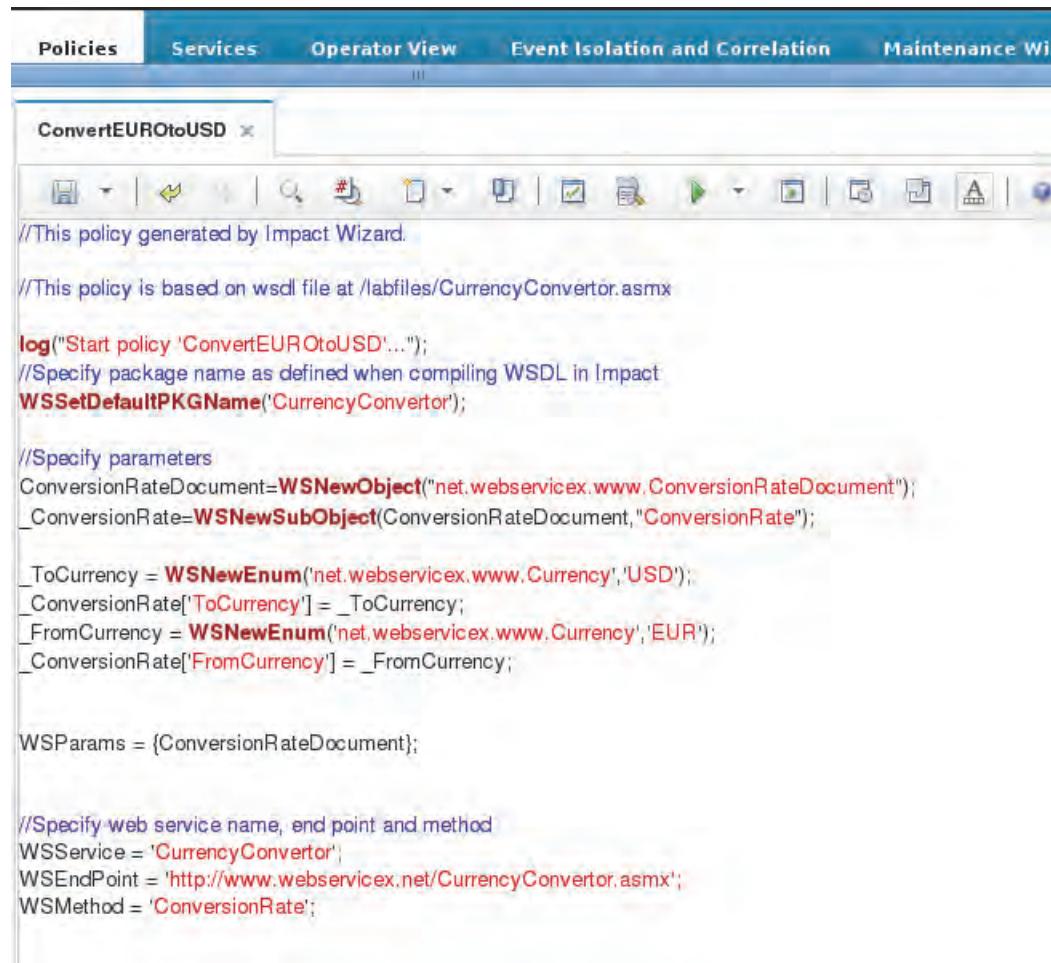
8. When the Wizard Complete screen opens, click **OK**.



9. If the screen is not visible or closes, at the bottom of the session, click **IBM Netcool/Impact - Mozilla Firefox** to redisplay the screen.



The policy that is displayed in the policy editor is like the following example.



```
Policies Services Operator View Event Isolation and Correlation Maintenance Wiz

ConvertEUROtoUSD ×

//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
WSSetName('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'
WSEndPoint = 'http://www.webservicex.net/CurrencyConvertor.asmx';
WSMethod = 'ConversionRate';
```

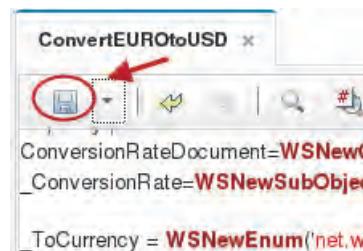
10. Examine the policy to see how the web services call **WSInvokeDL** was configured.

```
//Specify web service name, end point and method
WSService = 'CurrencyConvertor';
WSEndPoint = 'http://www.webservicex.net/CurrencyConvertor.asmx';
WSMethod = 'ConversionRate';

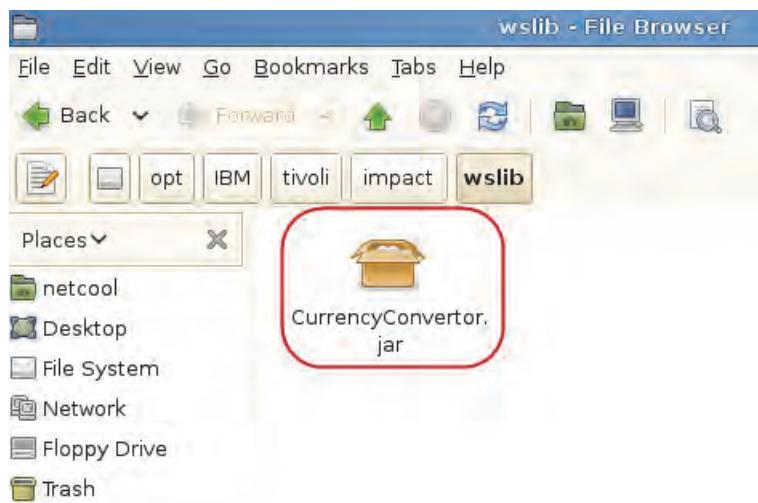
log("About to invoke Web Service call ConversionRate .....");

WSInvokeDLResult = WSInvokeDL(WSService, WSEndPoint, WSMethod, WSParams);
log("Web Service call ConversionRate return result: " +WSInvokeDLResult);
```

11. Save the policy created by the wizard. Click the **Save** icon.



The Web Services wizard created a JAR file and placed in  
**/opt/IBM/tivoli/impact/wslib/CurrencyConvertor.jar**.

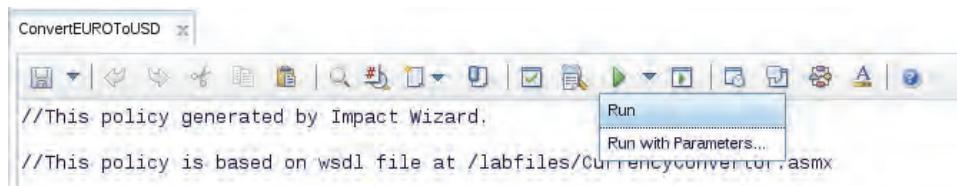


The Netcool/Impact server is configured by default to look in this location for application JAR files.  
They are automatically deployed into the Netcool/Impact server application.

## Testing the policy

Test the new policy in the policy editor.

1. While editing the policy **ConvertEUROToUSD**, click the Run ➔ icon.



If an Internet connection is not available, the Error window opens.

Running policy



2. Click the **Policy Log** icon to see your results. If the policy executed successfully, in the PolicyLogger you see the **ConversionRateResponse** from the **CurrencyConvertor** application, like the following example. The word **double** is replaced by the result of the conversion.

```
<soap:Envelope>
<soap:Body>
<ConversionRateResponse xmlns="http://www.webserviceX.NET/">
    <ConversionRateResult>double</ConversionRateResult>
<ConversionRateResponse>
</soap:Body>
</soap:Envelope>
```

Note: Web Services can change over time. Be sure to verify that the soap response tags are the same.

# Exercise 2 The web service response

In this exercise, for clarity you upload a second policy to parse the web service result.

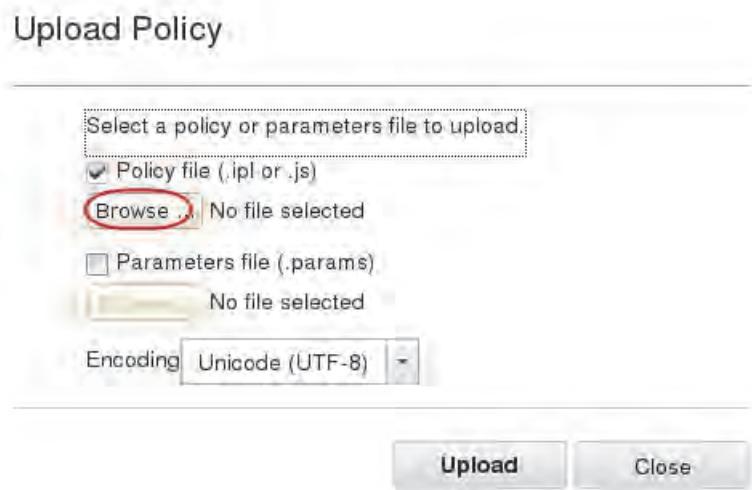
## Parsing the response

In this case, the returned result contains one variable that is surrounded by several soap tags. To continue our testing without an Internet connection you can create a similar structure within a policy.

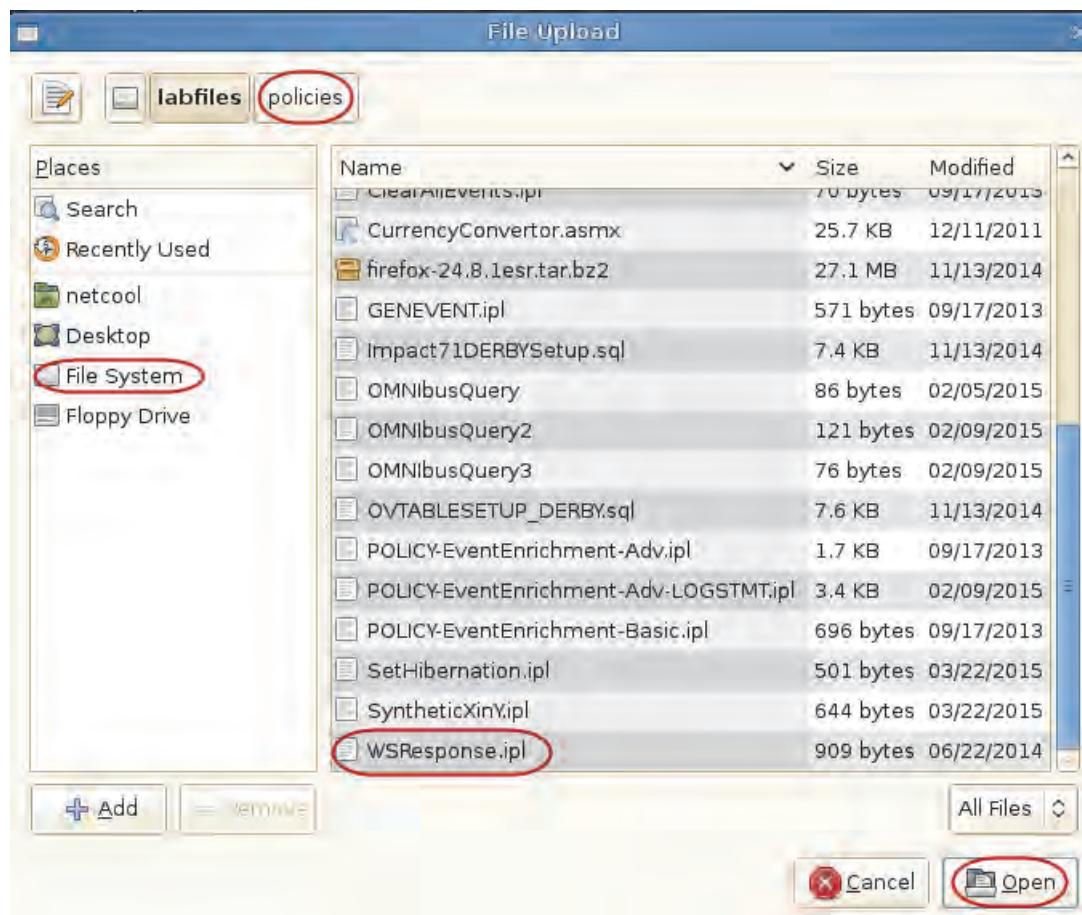
1. Click **Policies** to open the policy editor window.
2. Click the **Upload** icon.



3. When the **Upload a Policy File** window opens, click **Browse**.



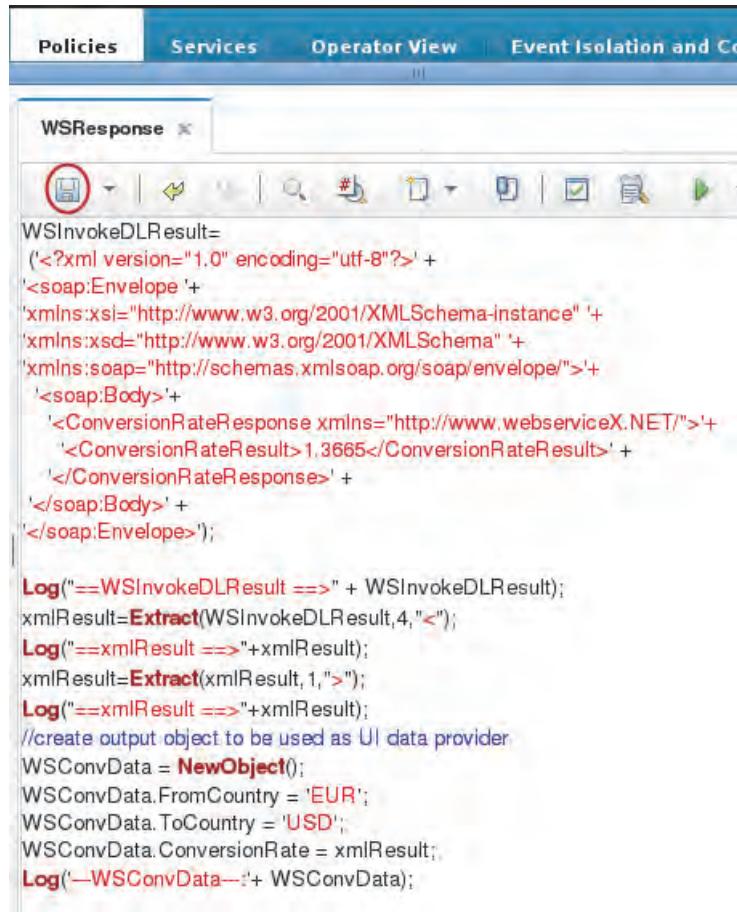
4. Highlight the **File System>labfiles** directory, select **WSResponse.ipl**, and click **Open**.



5. Click **Upload**.



6. Double-click **WSResponse** in the left panel to open the policy. The resulting policy should look like the following screen captures. Click the **Save** icon.



```
WSInvokeDLResult=
(<?xml version="1.0" encoding="utf-8"?> +
<soap:Envelope >+
+xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" +
+xmlns:xsd="http://www.w3.org/2001/XMLSchema" +
+xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">+
<soap:Body>+
<ConversionRateResponse xmlns="http://www.webserviceX.NET/">+
<ConversionRateResult>1.3665</ConversionRateResult> +
</ConversionRateResponse> +
</soap:Body> +
</soap:Envelope>);

Log("==WSInvokeDLResult ==>" + WSInvokeDLResult);
xmlResult=Extract(WSInvokeDLResult,4,"<");
Log("==xmlResult ==>" + xmlResult);
xmlResult=Extract(xmlResult,1,>);
Log("==xmlResult ==>" + xmlResult);
//create output object to be used as UI data provider
WSConvData = NewObject();
WSConvData.FromCountry = 'EUR';
WSConvData.ToCountry = 'USD';
WSConvData.ConversionRate = xmlResult;
Log('—WSConvData—:' + WSConvData);
```

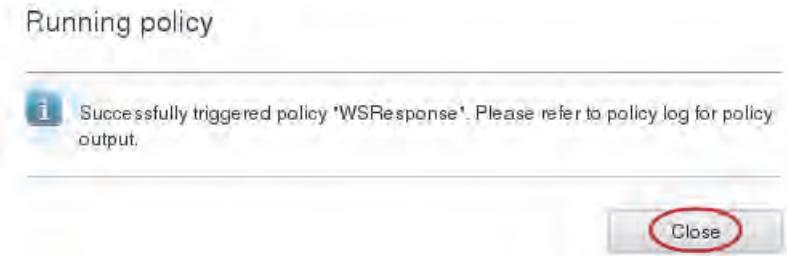
This policy accomplishes three tasks:

- The **WSInvokeDLResult** field contains the soap response that would have been returned from the CurrencyConvertor web service.
- The Extract function is used to parse the web service result.
- The NewObject function is used to create UI provider data for Jazz for Service Management.

7. Click the **Run** icon to execute the policy.



8. Click **Close** in the **Information** window.



9. Click **View Policy Log** icon to view the results.



At the bottom of the log entry you should see the following message:

---WSConvData---:"Created by parser"=(ConversionRate=1.3665, ToCountry=USD, FromCountry=EUR)

```
June 24, 2016 5:52:15 AM UTC[PolicyLogger][WSResponse][pool-3-thread-1574]Parser log: ==WSInvokeDLResult
==><?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body>
<ConversionRateResponse xmlns="http://www.webserviceX.NET"
/><ConversionRateResult>1.3665</ConversionRateResult></ConversionRateResponse></soap:Body></soap:Envelope>
June 24, 2016 5:52:15 AM UTC[PolicyLogger][WSResponse][pool-3-thread-1574]Parser log: ==xmlResult
==>ConversionRateResult:1.3665
June 24, 2016 5:52:15 AM UTC[PolicyLogger][WSResponse][pool-3-thread-1574]Parser log: ==xmlResult ==>1.3665
June 24, 2016 5:52:15 AM UTC[PolicyLogger][WSResponse][pool-3-thread-1574]Parser log: ---WSConvData---:"Created by
parser"=(ConversionRate=1.3665, ToCountry=USD, FromCountry=EUR)
```



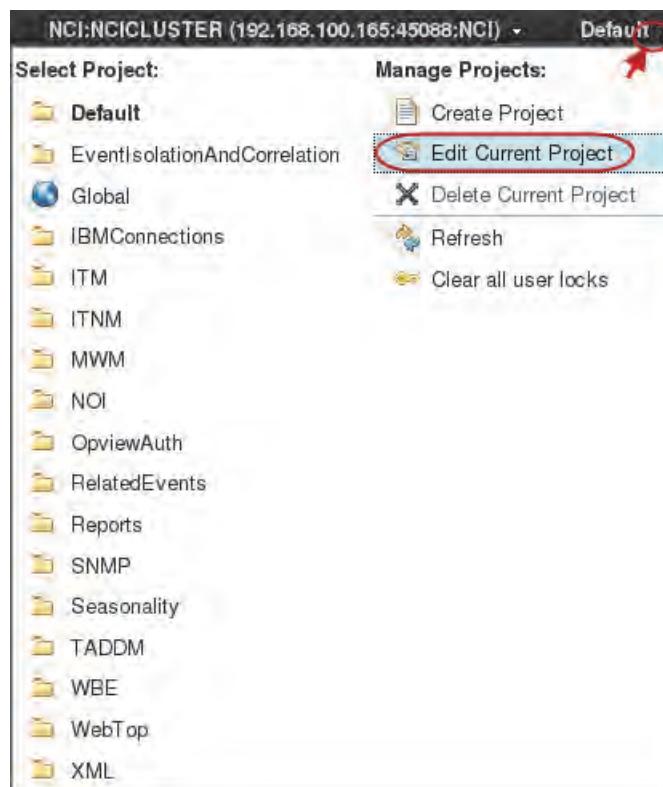
# Unit 13 Hibernation, X in Y, and synthetic events exercises

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

## Exercise 1 Hibernations

In this exercise, you experiment with hibernations.

1. Edit the **Default** project.



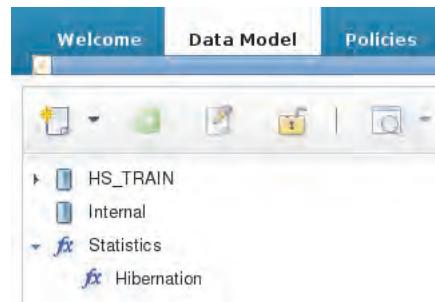
2. Add the **Internal** and **Statistics** Data Sources to the **Project Members** list. Click **OK**



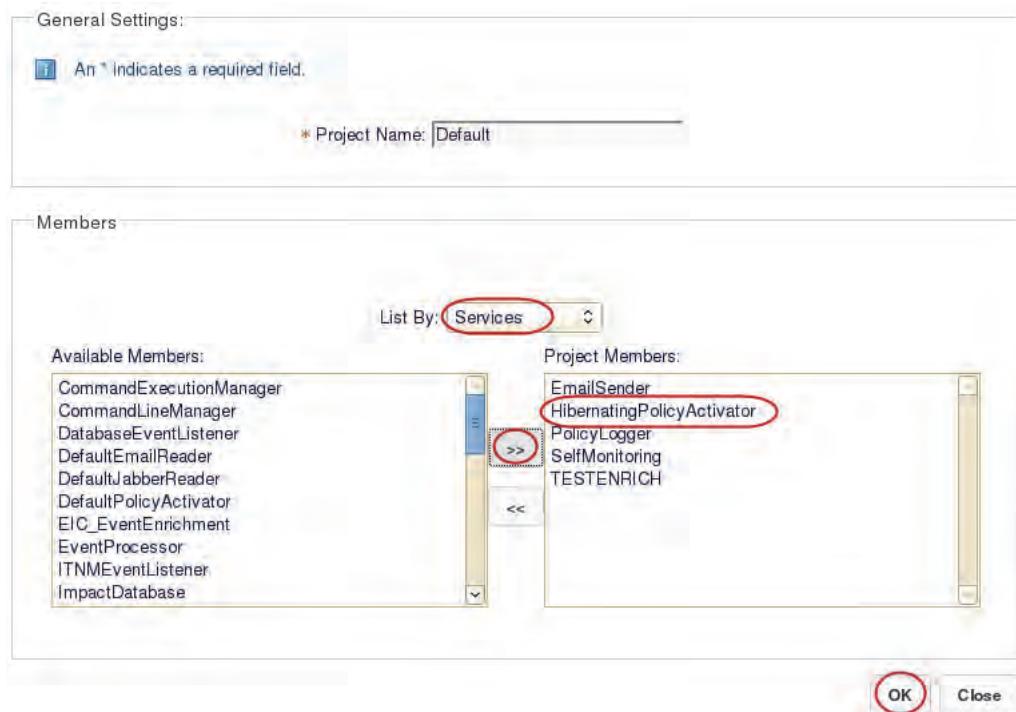
3. Select **Statistics** as the Data Source, List By **Data Types**, and move the **Hibernation** data type to the **Project Members**. Click **OK**.



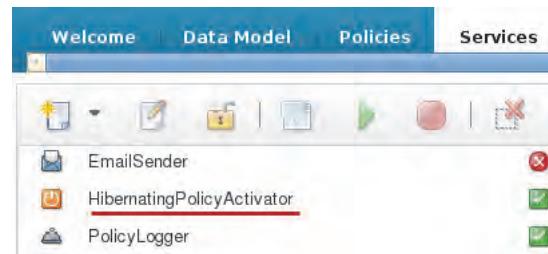
The result should be similar to this display.



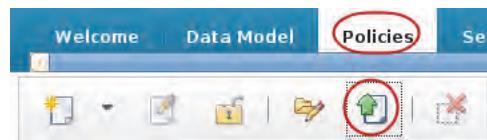
4. Select **Services** and add the **HibernatingPolicyActivator** from the Global Repository to the Project Members list. Click **OK**.



The result should be similar to this display.



5. Click the **Policies** tab and use the policy uploader to import the **SetHibernation.ipl** and **ActivateHibernation.ipl** policies from the **labfiles** directory.



SetHibernation.ipl

Upload Policy

---

Select a policy or parameters file to upload.

Policy file (.ipl or .js)  
Browse ... File to upload: SetHibernation.ipl

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

Encoding Unicode (UTF-8) ▾

---

**Upload** **Close**

ActivateHibernation.ipl

Upload Policy

---

Select a policy or parameters file to upload.

Policy file (.ipl or .js)  
Browse ... File to upload: ActivateHibernation.ipl

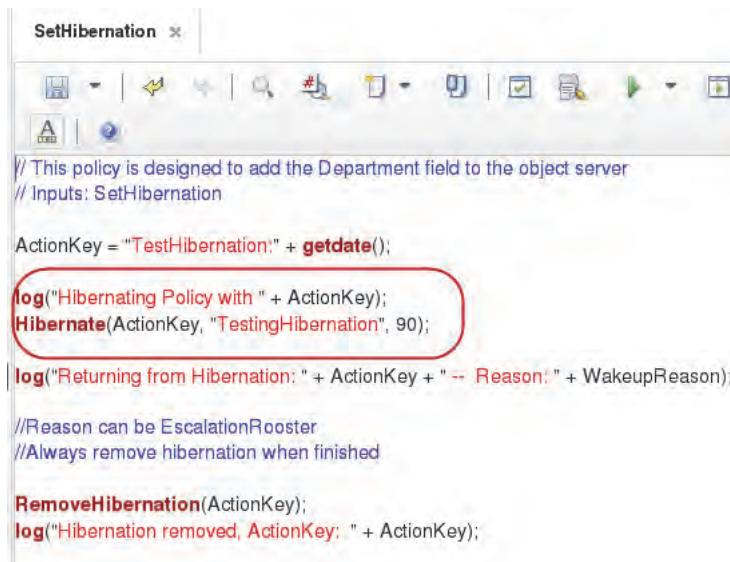
Parameters file (.params)  
Browse ... No file selected

Encoding Unicode (UTF-8) ▾

---

**Upload** **Close**

6. The **SetHibernation** policy hibernates a policy uniquely identified by the **ActionKey** for 90 seconds. Verify this parameter in the policy where shown.



```

SetHibernation ✎

// This policy is designed to add the Department field to the object server
// Inputs: SetHibernation

ActionKey = "TestHibernation:" + getdate();

log("Hibernating Policy with " + ActionKey);
Hibernate(ActionKey, "TestingHibernation", 90);

log("Returning from Hibernation: " + ActionKey + " -- Reason: " + WakeupReason);

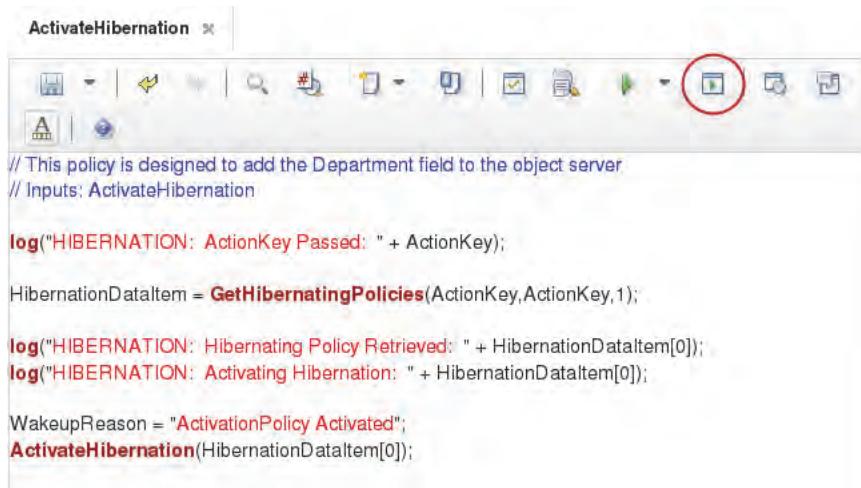
//Reason can be EscalationRooster
//Always remove hibernation when finished

RemoveHibernation(ActionKey);
log("Hibernation removed, ActionKey: " + ActionKey);

```

Hibernations are removed from hibernation in one of several ways: the user removes it through the GUI action (Wakeup Hibernation), the hibernation period expires, or the **ActivateHibernation** policy is triggered. The reason or method for waking up from hibernation is noted in the PolicyLogger log.

7. To test the **ActivateHibernation** policy, click the **Configure Policy Settings** icon.



```

ActivateHibernation ✎

// This policy is designed to add the Department field to the object server
// Inputs: ActivateHibernation

log("HIBERNATION: ActionKey Passed: " + ActionKey);

HibernationDataItem = GetHibernatingPolicies(ActionKey, ActionKey, 1);

log("HIBERNATION: Hibernating Policy Retrieved: " + HibernationDataItem[0]);
log("HIBERNATION: Activating Hibernation: " + HibernationDataItem[0]);

WakeupReason = "ActivationPolicy Activated";
ActivateHibernation(HibernationDataItem[0]);

```

8. Under **Policy Input Parameters** click the **New** button.

**Policy Settings**

Policy Input Parameters:

<input type="checkbox"/> Select	Parameter Name	Label	Format	Default Value	Description	Edit
<a href="#">New</a>						
<a href="#">Delete</a>						

9. Specify the new runtime parameter **ActionKey** as shown, and click **OK**.

Create a new Policy Input Parameter:

An \* indicates a required field.

* Name:	<input type="text" value="ActionKey"/>
Label:	<input type="text" value="ActionKey"/>
Format:	<input type="text" value="String"/>
Default Value:	<input type="text" value="HibernationID"/>
Description:	<input type="text"/>

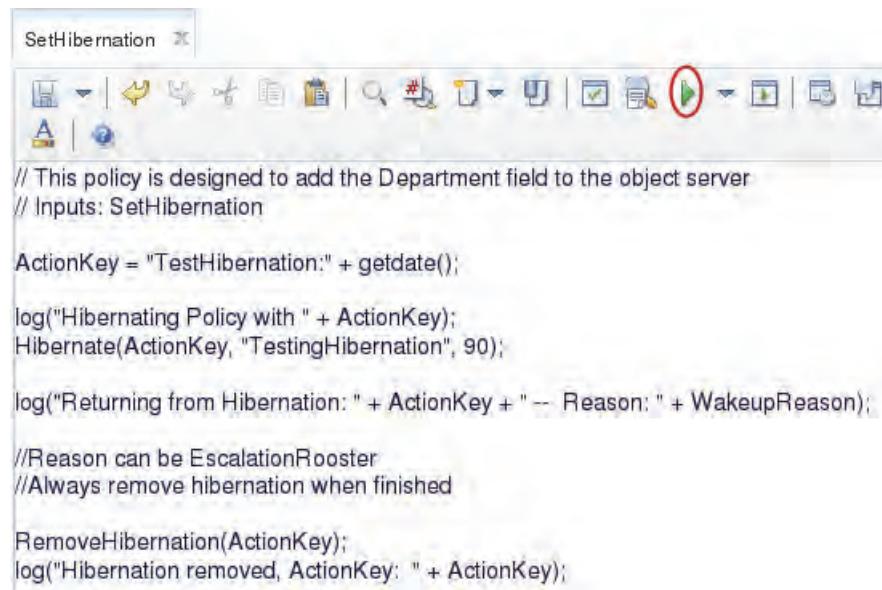
The resulting entry looks similar to the following example.

Policy Input Parameters:

<input type="checkbox"/> Select	Parameter Name	Label	Format	Default Value	Description	Edit
<input checked="" type="checkbox"/>	ActionKey	ActionKey	String	HibernationID		<a href="#">Edit</a>
<a href="#">Delete</a>						

10. Scroll to the bottom to click **OK**.

11. Open the **SetHibernation** policy and run the policy five or six times.



```
// This policy is designed to add the Department field to the object server
// Inputs: SetHibernation

ActionKey = "TestHibernation:" + getdate();

log("Hibernating Policy with " + ActionKey);
Hibernate(ActionKey, "TestingHibernation", 90);

log("Returning from Hibernation: " + ActionKey + " -- Reason: " + WakeupReason);

//Reason can be EscalationRooster
//Always remove hibernation when finished

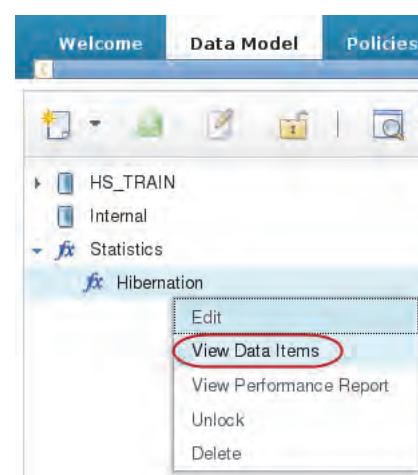
RemoveHibernation(ActionKey);
log("Hibernation removed, ActionKey: " + ActionKey);
```

12. Log entries will look similar to the display:

```
June 26, 2016 6:55:24 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernating Policy with TestHibernation:1466924124
June 26, 2016 6:56:13 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernating Policy with TestHibernation:1466924173
June 26, 2016 6:56:19 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernating Policy with TestHibernation:1466924179
June 26, 2016 6:56:24 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernating Policy with TestHibernation:1466924184
June 26, 2016 6:56:29 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernating Policy with TestHibernation:1466924189
June 26, 2016 6:56:35 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernating Policy with TestHibernation:1466924195
June 26, 2016 6:56:44 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernating Policy with TestHibernation:1466924204
June 26, 2016 6:56:55 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Returning from Hibernation: TestHibernation:1466924124
-- Reason: EscalationRooster
June 26, 2016 6:56:55 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernation removed,
ActionKey: TestHibernation:1466924124
```

13. View the data items for the **Hibernation** data type and look at the hibernations just started.

You may have to run the SetHibernation policy a few more times to catch entries here.



14. Click the pencil icon to edit one of the hibernations.

Select	Policy Name	Action Key	Edit
<input type="checkbox"/>	SetHibernation	TestHibernation:1466924464	
<input type="checkbox"/>	SetHibernation	TestHibernation:1466924473	
<input type="checkbox"/>	SetHibernation	TestHibernation:1466924482	
<input type="checkbox"/>	SetHibernation	TestHibernation:1466924468	
<a href="#">Delete</a>			

15. Write down the **Action Key** information for the hibernation you selected.
- 

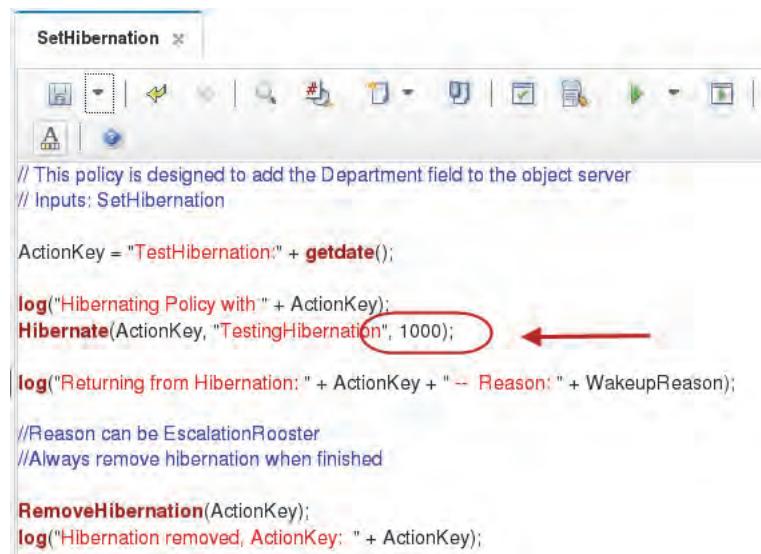
Edit Hibernation

Policy Name:	SetHibernation
Action Key:	TestHibernation:14669244
Wake Up Reason:	TestingHibernation
Wake Up Time:	Jun 26 2016, 07:02:34 UT
<a href="#">Wake Up</a>	

16. Review the log find the entries related to the Action Key that you selected. The reason for the return from hibernation is **EscalationRooster**.

Reason: EscalationRooster  
 June 26, 2016 6:56:55 AM UTC[PolicyLogger][SetHibernation][pool-3-thread-119]Parser log: Hibernation removed.  
 ActionKey: TestHibernation:1466924124

17. Modify the **SetHibernation** policy and replace the time value with **1000**. Check the syntax, and save the policy. Set new hibernations by clicking the green arrow 5 or 6 times.



```

SetHibernation.x
// This policy is designed to add the Department field to the object server
// Inputs: SetHibernation

ActionKey = "TestHibernation:" + getdate();

log("Hibernating Policy with " + ActionKey);
Hibernate(ActionKey, "TestingHibernation", 1000); ←

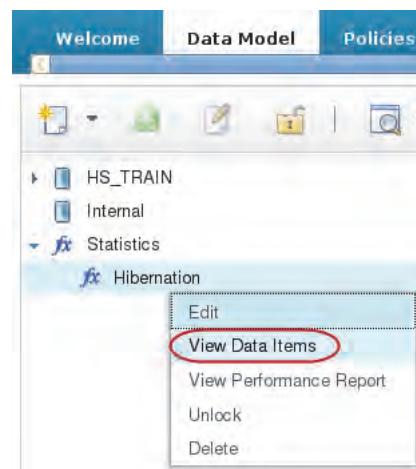
log("Returning from Hibernation: " + ActionKey + " -- Reason: " + WakeupReason);

//Reason can be EscalationRooster
//Always remove hibernation when finished

RemoveHibernation(ActionKey);
log("Hibernation removed, ActionKey: " + ActionKey);

```

18. View the data items for the **Hibernation** data type and look at the new hibernations.



19. Select and edit a hibernation.

Data Type Name: Hibernation

Number of Objects: 4

Filter Retrieved Data Items:

<input type="checkbox"/> Select	Policy Name	Action Key	Edit
<input type="checkbox"/>	SetHibernation	TestHibernation:1466924978	<input checked="" type="checkbox"/>
<input type="checkbox"/>	SetHibernation	TestHibernation:1466924990	<input checked="" type="checkbox"/>
<input type="checkbox"/>	SetHibernation	TestHibernation:1466924973	<input checked="" type="checkbox"/>
<input type="checkbox"/>	SetHibernation	TestHibernation:1466924982	<input checked="" type="checkbox"/>
<a href="#">Delete</a>			

20. Write down the **Action Key** information.

Data Items: Hibernation

Hibernating Policy

Edit Hibernation

Policy Name:	SetHibernation
Action Key:	TestHibernation:1466924978
Wake Up Reason:	TestingHibernation
Wake Up Time:	Jun 26 2016, 07:26:18 UT
<a href="#">Wake Up</a>	

21. Edit the **ActivateHibernation** policy and select **Run With Parameters**.

ActivateHibernation SetHibernation

// This policy is designed to add the Department field to the object  
// Inputs: ActivateHibernation

```

log("HIBERNATION: ActionKey Passed: " + ActionKey);
HibernationDataItem = GetHibernatingPolicies(ActionKey, ActionKey, 1);
log("HIBERNATION: Hibernating Policy Retrieved: " + HibernationDataItem[0]);
log("HIBERNATION: Activating Hibernation: " + HibernationDataItem[0]);

WakeUpReason = "ActivationPolicy Activated";
ActivateHibernation(HibernationDataItem[0]);

```

22. Enter the **ActionKey** that you recorded earlier, including **TestHibernation**: Click **Execute** to run the policy. Click **Close** when complete.

Policy Input Parameters:

ActionKey: stHibernation:1426262201

Help      Execute      Close

Successfully triggered policy 'ActivateHibernation'. Please refer to policy log for policy output.

23. View the PolicyLogger log. Find the entries for the preceding action.



**Note:** The reason for **Returning from Hibernation** is logged as **ActivationPolicy Activated** which was set in the **ActivateHibernation** policy.

March 13, 2015 4:00:41 PM UTC[PolicyLogger][SetHibernation][pool-3-thread-46]Parser log: Returning from Hibernation:  
TestHibernation:1426262201 -- Reason: ActivationPolicy Activated

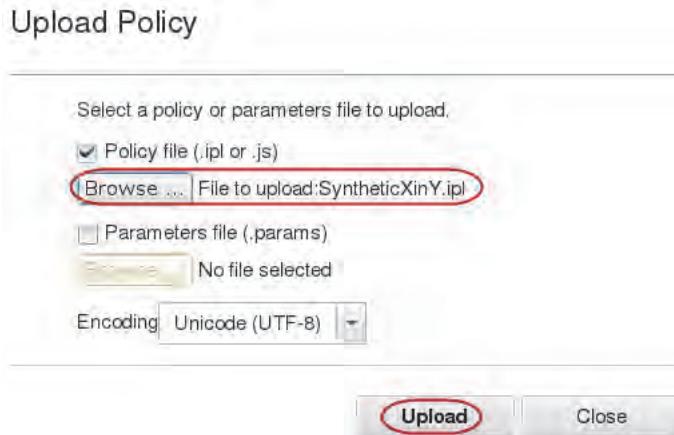
Eventually the hibernation interval for the remaining hibernations expires. When this interval expires, the log file entries indicate that the reason for returning from hibernation is **EscalationRooster**.

After a few minutes, the **Data Item** list for the data type **Hibernation** no longer lists entries.

# Exercise 2 Using a policy to create synthetic events

In this exercise, you use a policy to create synthetic events in the **alerts.status** table in OMNIbus.

1. Use the **Policy Uploader** to upload the **SyntheticXinY.ipl** policy from the **labfiles** directory.

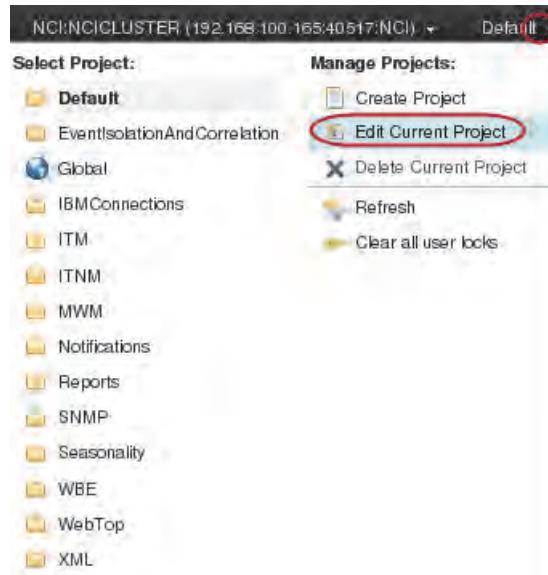


2. Edit the policy in the policy window. Examine the field settings for the **SynthEvent NewObject** object and the **AddDataItem** function. Click **Save**.

```
// Generating a synthetic event using the NewObject() function
SynthEvent = NewObject();
SynthEvent.Identifier = "SynthEvent-NewEvent" + GetDate();
SynthEvent.Node = "jazz01";
SynthEvent.AlertKey = "LoginAttempt";
SynthEvent.Manager = "jazz01";
SynthEvent.Summary = "Attempt to login as netcool from host jazz01 failed";
SynthEvent.Severity = 4;
SynthEvent.AlertGroup = "GET_LOGIN_TOKEN";
SynthEvent.Summary = "Attempt to login as netcool from host jazz01 failed";
SynthEvent.FirstOccurrence = GetDate();
SynthEvent.LastOccurrence = GetDate();
SynthEvent.Type = 1;
SynthEvent.Tally = 1;
AddDataItem("NCOMS_ALERTS", SynthEvent);
```

3. Click **Data Model** to open the **Data Model** tab. If **defaultobjectserver** is not listed, add it to the **Default** Project.

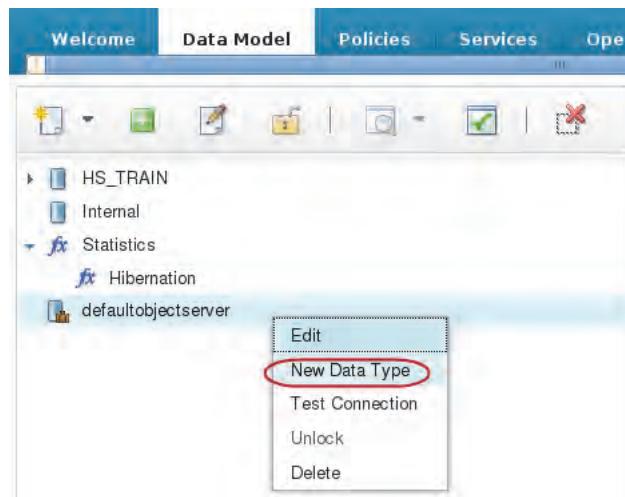
- a. Edit the **Default** project.



- b. Select List By **Data Sources**, and add the **defaultobjectserver** to the Project Members. Click **OK**.



4. Right-click the **defaultobjectserver** entry in the list. Select **New Data Type**.



5. Create the **NCOMS\_ALERTS** data type as depicted in the following screen capture. The **Base Table** entry is **alerts.status**.

A screenshot of the 'New Data Type for defaultobjectserver' configuration page. The title bar says 'New Data Type for defaultobjectserver'. Below it is a toolbar with a save icon. The main area is titled 'SQL Data Type Config Page' and has tabs for 'Table Description', 'Dynamic Links', and 'Cache Settings'. The 'Table Description' tab is selected. It contains fields for 'Data Type Name' (set to 'NCOMS\_ALERTS'), 'Data Source Name' (set to 'defaultobjectserver'), and checkboxes for 'Enabled' (checked) and 'Access the data through UI data provider' (unchecked). Below this is a 'Table Description' section with a 'Configure the database table and fields' sub-section. It shows a 'Base Table' field set to 'alerts.status' (also highlighted with a red oval) and a 'Refresh Fields' button (also highlighted with a red oval). There is also a checkbox for 'Show New / Deleted Fields' (unchecked).

**Note:** Remember to click the **Refresh** button.

6. Scroll down to display the schema. Select Identifier as the Key Field.

SQL Data Type Config Page

Table Description		Dynamic Links		Cache Settings	
Delete Selection		New	Edit	Move Up	Move Down
ID	Field Name	Format	Display Name	Description	Key Field
Identifier	Identifier	LONG_STRING	Identifier	Identifier	<input checked="" type="checkbox"/> Yes
Serial	Serial	INTEGER	Serial	Serial	No
Node	Node	LONG_STRING	Node	Node	No
NodeAlias	NodeAlias	LONG_STRING	NodeAlias	NodeAlias	No
Manager	Manager	LONG_STRING	Manager	Manager	No

7. Click **Save**. Close the window.

## Exercise 3 Testing synthetic events

1. Click **View Data Items** for the **NCOMS\_ALERTS** data type. Click **Select**. Scroll to the bottom and to click **Delete** to remove all items from the list.

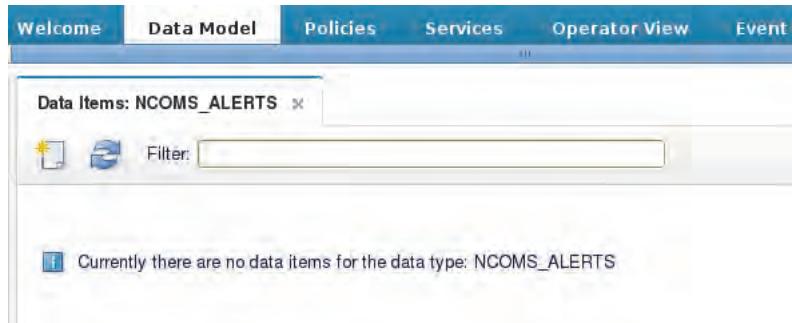
Data Items: NCOMS\_ALERTS

<input checked="" type="checkbox"/> Select	Identifier	Serial	Node	NodeAlias	Manager	Agent	AlertGroup
<input checked="" type="checkbox"/>	OMNibus ObjectServer: Connections available for NCOMS:	1577	jazz01		OMNibus Self Monitoring @NCOMS	OMNibus SelfMonitoring	ConnectionSt
<input checked="" type="checkbox"/>	Impact Memory Status for NCI:NCICLUSTER at 2016-06-26 07:54:33.347	1590	jazz01.tivoli.edu	192.168.100.165	Impact SelfMonitoring@NCI:NCICLUSTER	Impact SelfMonitoring	MemoryStat
<input checked="" type="checkbox"/>	Impact Memory Status for NCI:NCICLUSTER at 2016-06-26 07:54:33.347	1588	jazz01.tivoli.edu	192.168.100.165	Impact SelfMonitoring@NCI:NCICLUSTER	Impact SelfMonitoring	MemoryStat

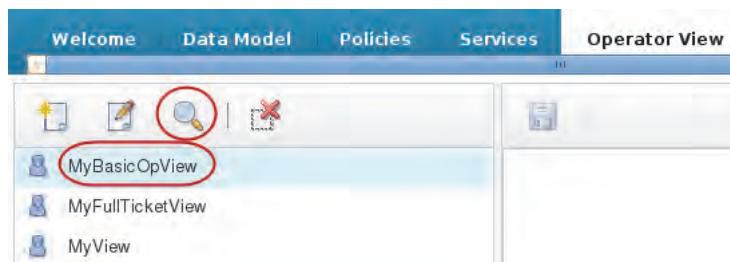
The resulting message opens, as shown in the example. If events continue to show, turn off the Simnet Probe.



**Note:** To stop the Simnet Probe, locate the processid for `nco_p_simnet`. Use `ps -ef | grep nco_p_simnet` as described earlier. Issue the `kill <processid>` command to stop the process.



2. **MyBasicOpView** can also be used to delete object server events. Click the Operator View tab. Single click to highlight **MyBasicOpView**.
3. Click the hour glass to display the operator view.



4. Click the Action **ClearAllEvents**.

The screenshot shows the 'Actions' section of the Operator View. A button labeled 'ClearAllEvents' is circled in red.

**Information Groups**

AllNodes	NODE	IP	NODEID	CUSTID	DEPTID	SERVID
gambit	10.10.10.0	1	1	1		
wombat	10.10.10.1	2	2	2		
orac	10.10.10.2	3	3	3		
muppet	10.10.10.3	4	4	4	0	
moose	10.10.10.4	5	5	5	5	
hal	10.10.10.5	6	6	6	6	
vixen	10.10.10.6	7	7	7	7	
dewey	10.10.10.7	8	8	8	1	

5. Click **Ok** to continue.



6. Edit the **SyntheticXinY.ipl** policy.

- Click the green arrow to execute the **SyntheticXinY.ipl** policy three times. **View Data Items** again to see the new **NCOMS\_ALERTS** events. Click the **Refresh** icon if events do not immediately show in the list.

Data Items: NCMS\_ALERTS

Select	Identifier	Serial	Node	NodeAlias	Manager	Agent	AlertGroup	AlertKey	Severity	Summary
<input type="checkbox"/>	SynthEvent-NewEvent1466928625	1639	jazz01		jazz01		GET_LOGIN_TOKEN	LoginAttempt	4	Attempt to log as netcool from host jazz01 failed
<input type="checkbox"/>	SynthEvent-NewEvent1466928630	1640	jazz01		jazz01		GET_LOGIN_TOKEN	LoginAttempt	4	Attempt to log as netcool from host jazz01 failed
<input type="checkbox"/>	SynthEvent-NewEvent1466928635	1641	jazz01		jazz01		GET_LOGIN_TOKEN	LoginAttempt	4	Attempt to log as netcool from host jazz01 failed

- Navigate back to the **Policies** window. Edit the **SyntheticXinY.ipl** policy and remove the **GetDate()** function from the **SynthEvent.Identifier** line as shown in the example. Click the **Check Syntax** icon. Click **Save**.

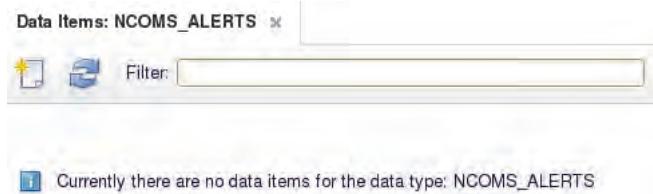
SyntheticXinY

```
// Generating a synthetic event using the NewObject() function

SynthEvent = NewObject();

SynthEvent.Identifier = "SynthEvent-NewEvent";
SynthEvent.Node = "jazz01";
SynthEvent.AlertKey = "LoginAttempt";
SynthEvent.Manager = "jazz01";
SynthEvent.Summary = "Attempt to login as netcool from host jazz01 failed";
SynthEvent.Severity = 4;
SynthEvent.AlertGroup = "GET_LOGIN_TOKEN";
SynthEvent.Summary = "Attempt to login as netcool from host jazz01 failed";
SynthEvent.FirstOccurrence = GetDate();
SynthEvent.LastOccurrence = GetDate();
SynthEvent.Type = 1;
SynthEvent.Tally = 1;
AddDataItem("NCOMS_ALERTS", SynthEvent);
```

9. Clear the events in the **NCOMS\_ALERTS** table.



10. Click the **Run** icon to execute the **SyntheticXinY.ipl** policy four times.

11. Click **View Data Items** to see the new **NCOMS\_ALERTS** events.



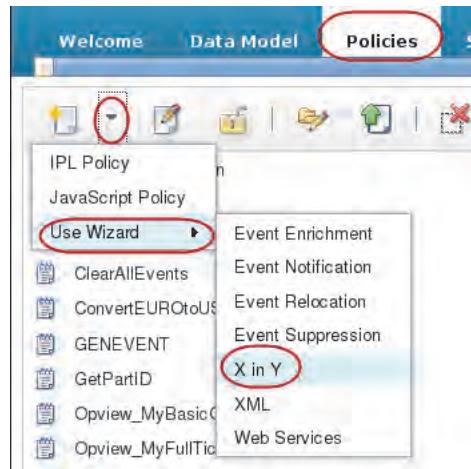
**Hint:** With deduplication, only one Synthetic event is displayed. The **GetDate** function created unique entries before it was removed.

Data Items: NCOMS_ALERTS								
	NCOMS							
<input type="checkbox"/>	OMNibus ObjectServer: Total SQL time for all clients NCOMS:	1657	jazz01		OMNibus Self Monitoring @NCOMS	OMNibus SelfMonitoring	ClientStatus	2
<input type="checkbox"/>	OMNibus ObjectServer: Trigger Status for NCOMS:	1659	jazz01		OMNibus Self Monitoring @NCOMS	OMNibus SelfMonitoring	TriggerStatus	2
<input type="checkbox"/>	SynthEvent-NewEvent	1656	jazz01	jazz01			GET_LOGIN_TOKEN	LoginAttempt
								4

# Exercise 4 X in Y policy

X in Y policies monitor the number of occurrences of an event within a sliding period of time. This policy detects five failed ObjectServer attempts that occur within 30 seconds. When the 30-second threshold is exceeded, the event summary is updated to indicate that the threshold is exceeded.

1. Click Policies to open the Policies tab. Click the New Policy icon. Select Use Wizard > X in Y.



2. In the Policy Name field, type the value **XinYLogins**. Click Next.

X Events in Y Seconds-Introduction Step: 1 of 4

X Events in Y Seconds (X in Y) is the process of suppressing Netcool events until a certain number of identifiable events occur within a specified period of time.

This wizard will guide you through the steps necessary to create X Events in Y Seconds policy.

Enter a name for the new policy and click **Next >>** to continue.

Policy Name:

Help Cancel Next >>

3. Select **OMNIbusEventReader** as the event reader. In the **Restriction Filter** field, type **Severity > 0**. Click **Next**.

X Events in Y Seconds-Connect to Event Reader Step: 2 of 4

Event Source:

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

Event Reader: OMNIbusEventReader

Restriction Filter:

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

Severity > 0

Help Cancel << Back Next >>



4. Set the **Event Threshold** to 3. Set the **Time Window** to 30. Click **Next**.

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 (events): 3

Time Threshold

Set the length of the time window in seconds:

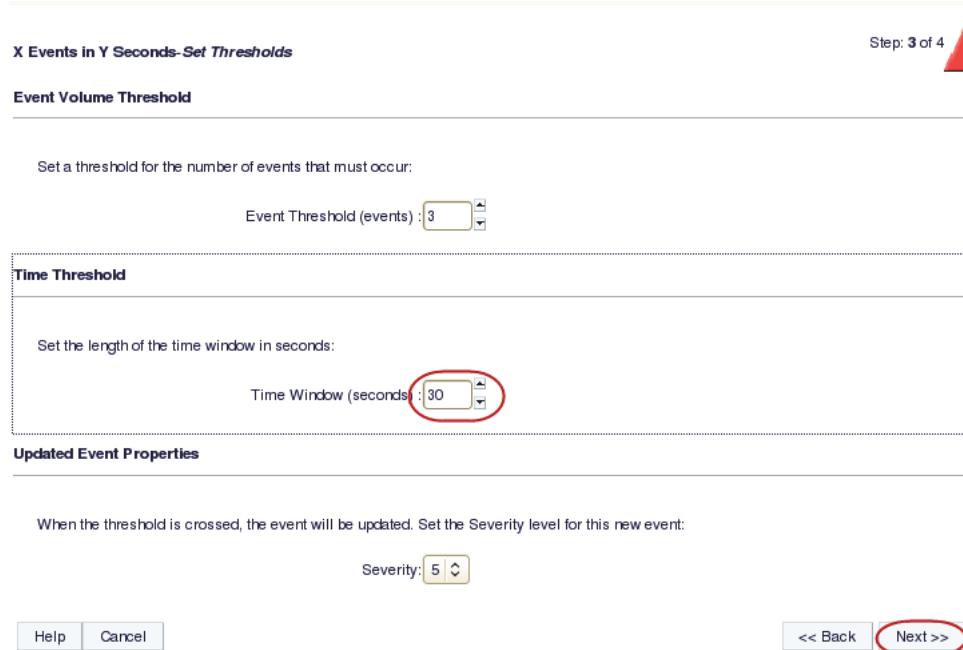
Time Window (seconds): 30

Updated Event Properties

When the threshold is crossed, the event will be updated. Set the Severity level for this new event:

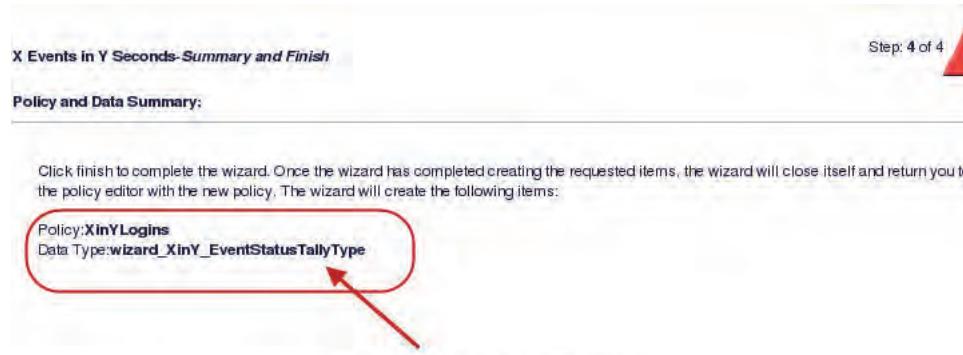
Severity: 5

Help Cancel << Back Next >>

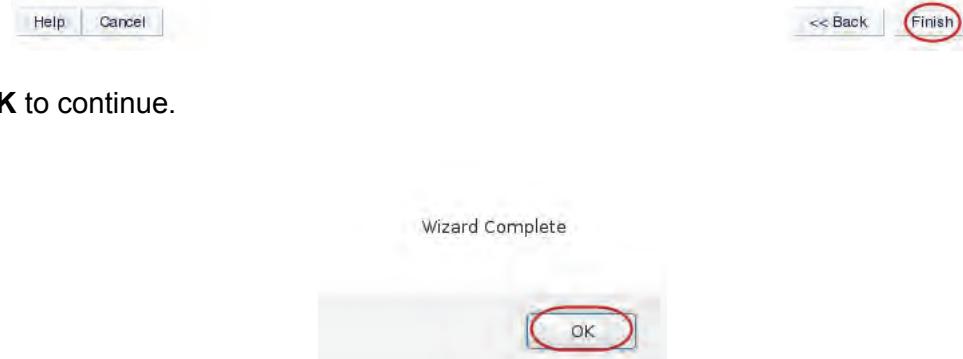


Notice the data type automatically created for the policy to use.

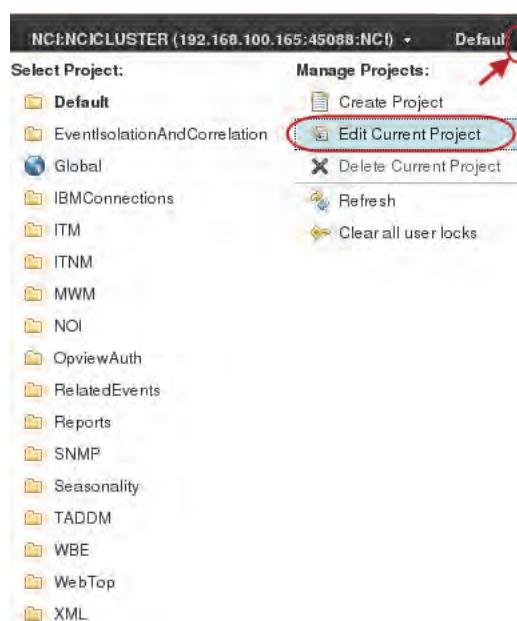
5. Click **Finish**.



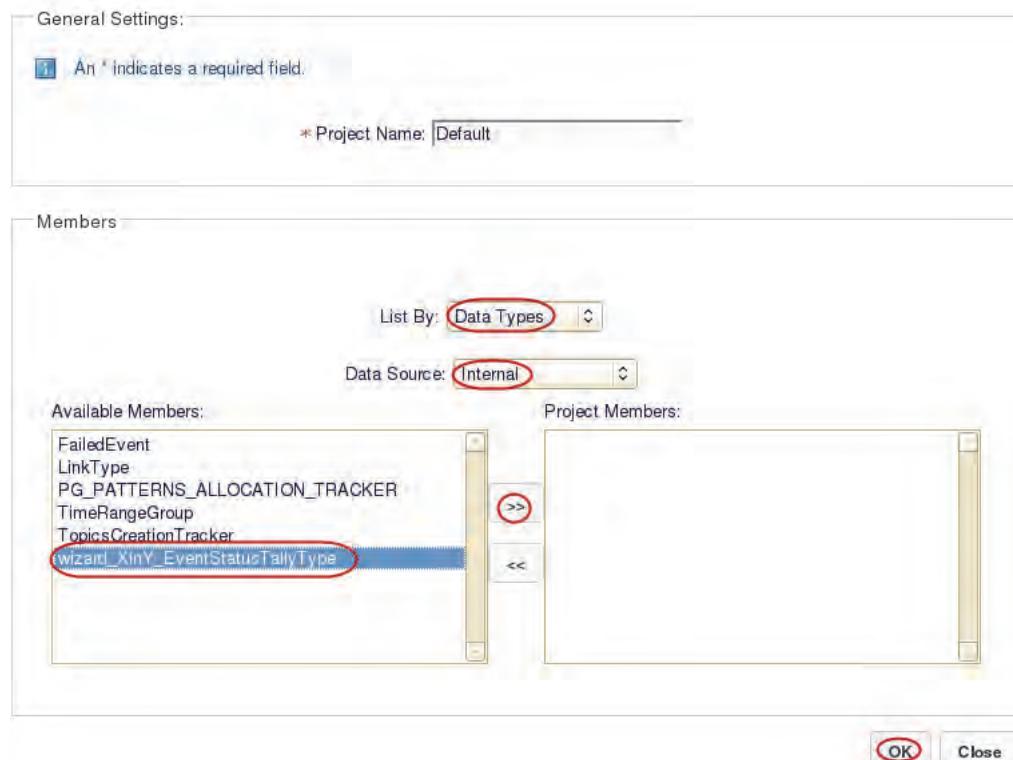
6. Click **OK** to continue.



7. Edit the **Default** project.



8. Add the **wizard\_XinY\_EventStatusTallyType** Data Type created by the wizard to the **Internal** Data Source as shown in the example.

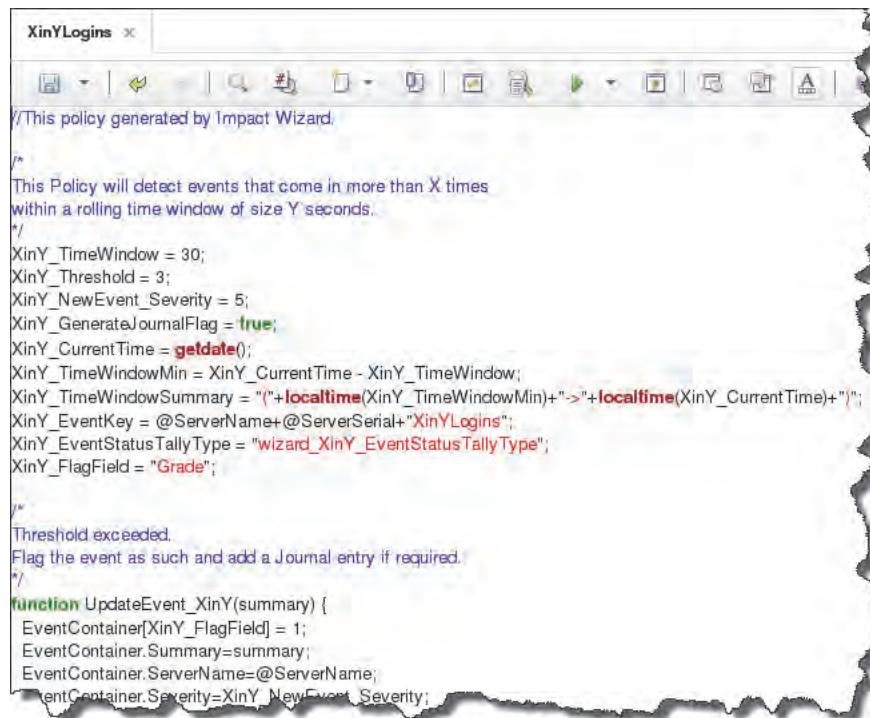


9. Select the **Internal** Data Source and examine the **wizard\_XinY\_EventStatusTallyType** Data Type.

The screenshot shows the Data Model interface. The top navigation bar includes 'Welcome', 'Data Model' (selected), 'Policies', 'Services', 'Operator View', 'Event Isolation and Correlation', 'Maintenance Window', and 'Repo'. The left sidebar has icons for HS\_TRAIN, Internal (selected), Statistics, and defaultobjectserver. The main area shows the 'wizard\_XinY\_EventStatusTallyType' data type configuration page. The title bar says 'wizard\_XinY\_EventStatusTallyType'. The page header includes a help icon and a close button. Below the header is a section titled 'Internal Data Type Config Page'. There are two tabs: 'Table Description' (selected) and 'Dynamic Links'. The 'Table Description' tab displays a table with four columns: Name, Type, Description, and Actions. The table contains four rows:

Name	Type	Description	Actions
RunningTally	INTEGER	RunningTally	RunningTally
PreviousTally	INTEGER	PreviousTally	PreviousTally
LastOccurrence	INTEGER	LastOccurrence	LastOccurrence

10. View the policy **XinYLogins** in the **Policies** window.



```
XinYLogins x

//This policy generated by Impact Wizard.

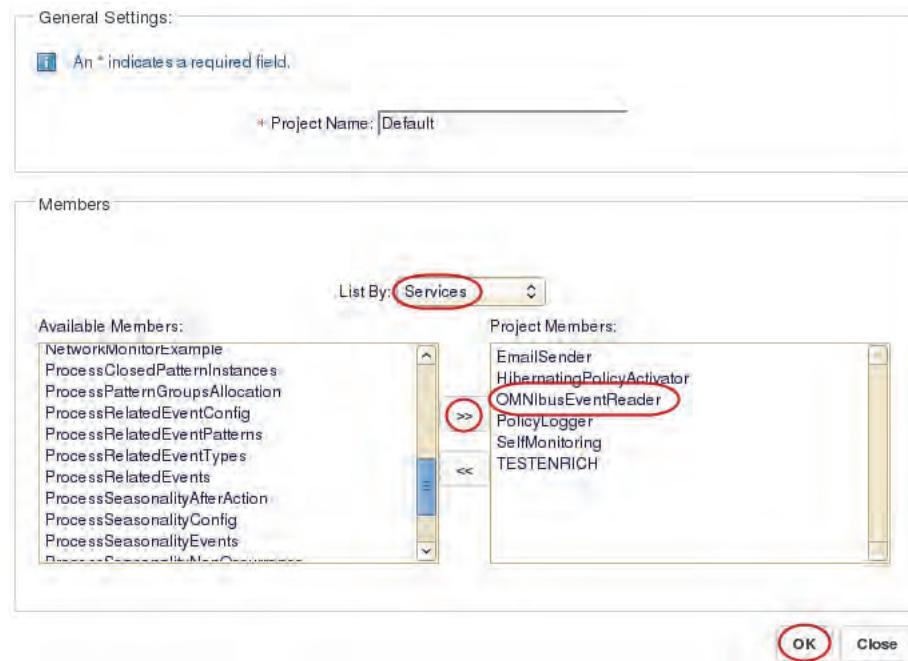
/*
This Policy will detect events that come in more than X times
within a rolling time window of size Y seconds.
*/
XinY_TimeWindow = 30;
XinY_Threshold = 3;
XinY_NewEvent_Severity = 5;
XinY_GenerateJournalFlag = true;
XinY_CurrentTime = getdate();
XinY_TimeWindowMin = XinY_CurrentTime - XinY_TimeWindow;
XinY_TimeWindowSummary = "("+localtime(XinY_TimeWindowMin)+">" +localtime(XinY_CurrentTime)+")";
XinY_EventKey = @ServerName+@ServerSerial+"XinYLogins";
XinY_EventStatusTallyType = "wizard_XinY_EventStatusTallyType";
XinY_FlagField = "Grade";

/*
Threshold exceeded.
Flag the event as such and add a Journal entry if required.
*/
function UpdateEvent_XinY(summary) {
    EventContainer[XinY_FlagField] = 1;
    EventContainer.Summary=summary;
    EventContainer.ServerName=@ServerName;
    EventContainer.Severity=XinY_NewEvent_Severity;
```

**Note:** Log statements assist in debugging the policy.

# Exercise 5 Testing the policy

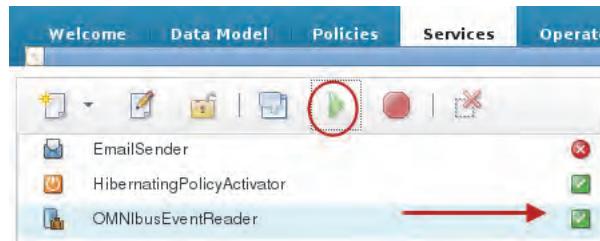
- Add the **OMNIBusEventReader** service to the **Default** project, as shown in the example.



- Click the **Services** tab and Edit **OMNIBusEventReader** general settings. Notice the mapping tab is already configured.

The screenshot shows the configuration interface for the 'OMNIBusEventReader' service. At the top, there are tabs: 'Services', 'Operator View', 'Event Isolation and Correlation', and 'Maintenance Window'. Below that is a search bar with the text 'OMNIBusEventReader'. The main area is titled 'OMNIBusEventReader Service' and has two tabs: 'General Settings' (selected) and 'Event Mapping' (circled in red). Under 'General Settings', there is a section 'Event Matching:' with a note: 'Set events to trigger policies when they match a specific filter.' It contains two radio buttons: 'Test events with all filters' (selected) and 'Stop testing after first match'. Below this is a section 'Action:' with three checkboxes: 'Get updated events' (checked), 'Get Status Events', and 'Run policy on deletes'. A dropdown menu 'Policy:' is set to 'ACMEBP\_Prod'. To the right of this is another 'OMNIBusEventReader Service' panel with tabs 'General Settings' and 'Event Mapping'. The 'Event Mapping' tab is selected and shows a table titled 'Event Mapping Table'. The table has columns: 'Restriction Filter', 'Policy Name', 'Active', and 'Chain'. One row is visible: 'Severity > 0' under 'Restriction Filter', 'XinYLogins' under 'Policy Name', 'Yes' under 'Active', and 'No' under 'Chain'. At the bottom of the table are buttons: 'Delete Selection', 'New', 'Edit', 'Move Up', 'Move Down', and 'Quick Add'.

3. Start the **OMNIbusEventReader** service.



4. Delete the rows from the **NCOMS\_ALERTS** table.



5. Test the policy by generating three login errors within 10 seconds. From a GNOME Terminal, type the following commands:

```
cd /opt/IBM/tivoli/netcool/omnibus/bin  
.nco_event
```

6. When the login window opens, type **netcool** as the user name, and **test** as the password. Click **OK** three times quickly.



An error in OMNIbus error occurs, and the **XinY** policy generates an threshold event.

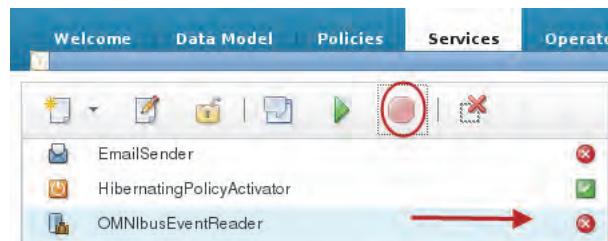
7. Click **OK** to continue.



A successful test displays **XinYLogins Threshold Exceeded: 3 >= 3** as part of the **Summary** field in **NCOMS\_ALERTS** table.

Attempt to login as jetcool from host jazz01 failed	1688	jazz01	SecurityWatch	e@CoA864A5@CoA864A5:0.0	5 XinYLogins Threshold Exceeded: 3 >= 3 , over the last 30 s(2016-06-09:00:07.000->2016-06-09:00:37.000)
---	------	--------	---------------	-------------------------	--

8. Stop the OMNIbusEventReader service in the Services panel.



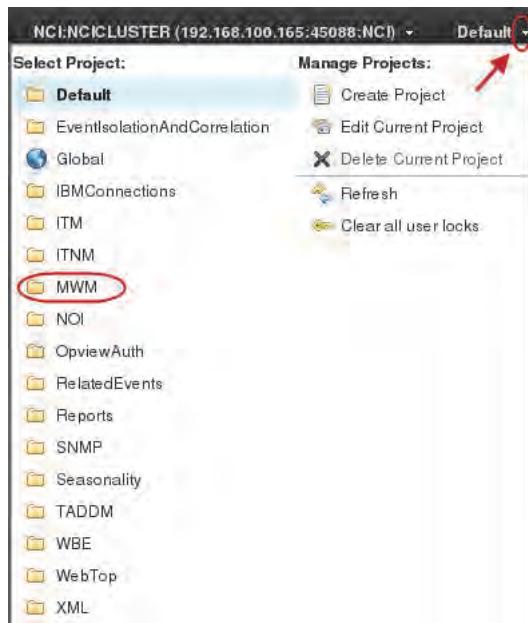


# Unit 14 Maintenance window management exercises

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

## Exercise 1 Maintenance window management

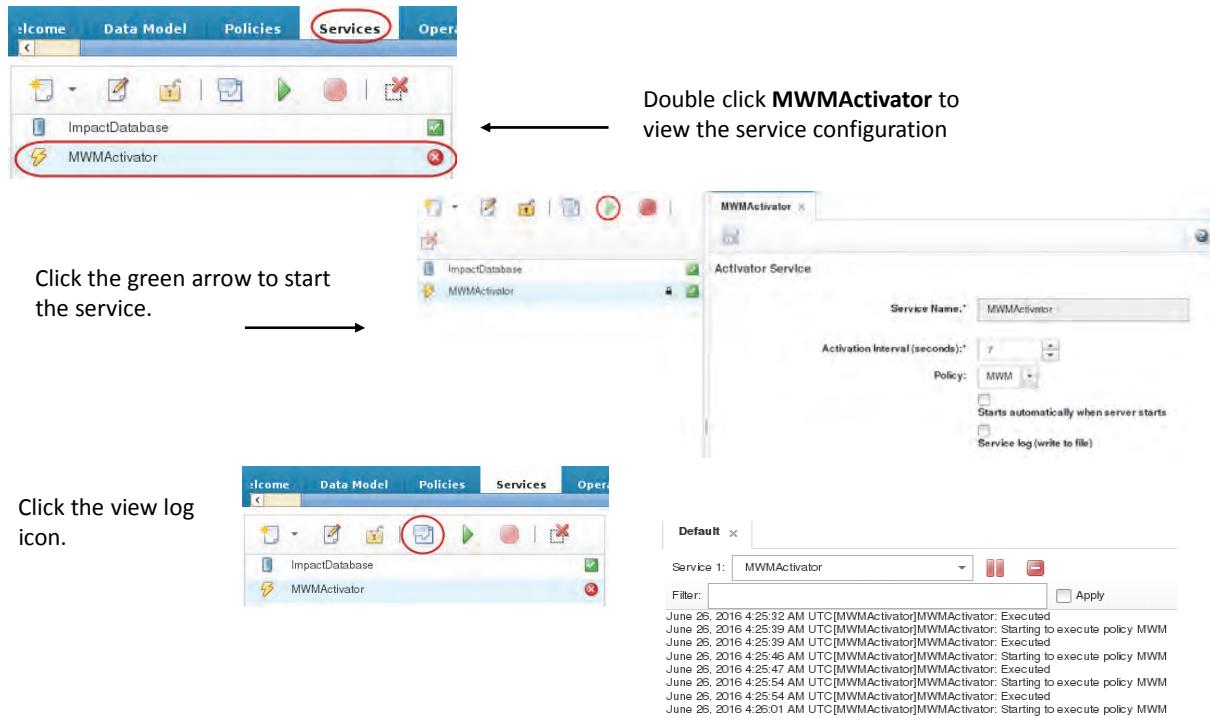
1. Switch to the MWM project.



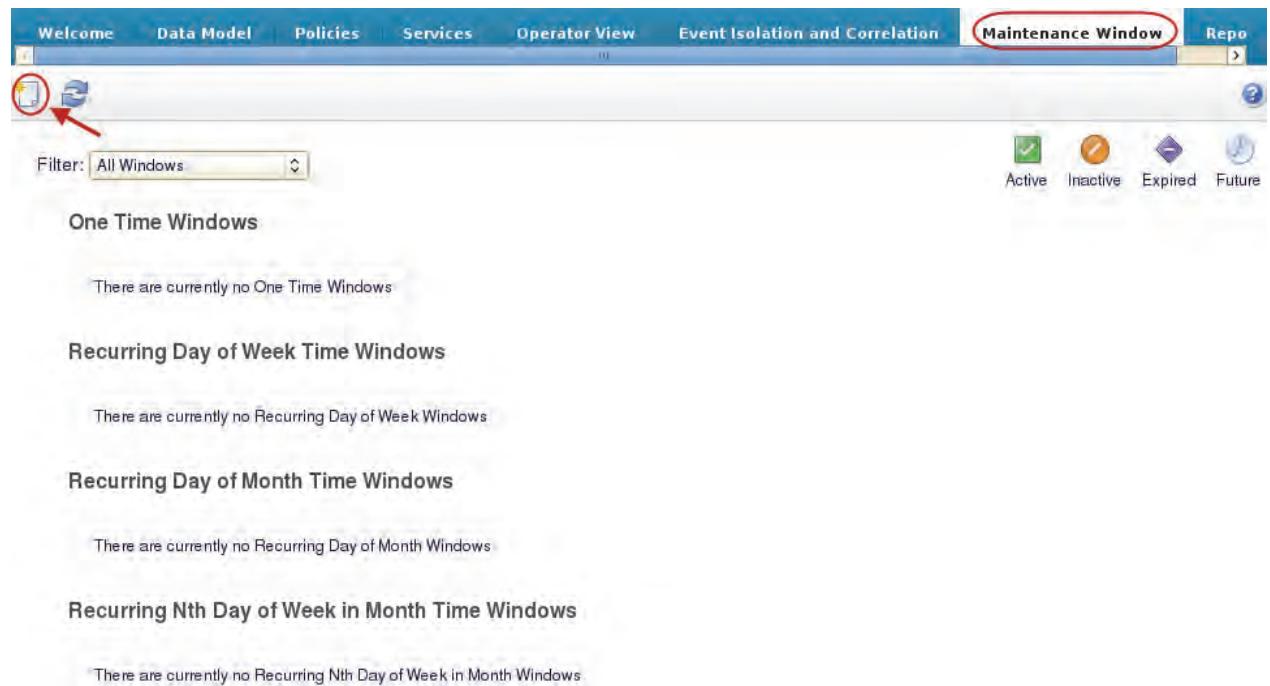
2. Click **OK**.



- Click the **Services** tab. Start the **MWMActivator** service as shown in the following screen capture.



- Click the **Maintenance Window** tab to open the home screen. Click the icon to create a new maintenance window.



5. Click **Time Zone** and select to **New York, Toronto**. Type **Link** in the **AlertGroup** field.

The screenshot shows the 'Maintenance Window Editor' interface. In the 'Maintenance Window General Settings' section, the 'Type of Maintenance window' is set to 'One Time' and the 'Time Zone' is set to '(UTC-05:00) New York, Toronto'. In the 'Identify Affected Events' section, there is a table with four rows. The second row, where 'AlertGroup' is set to 'Link', has its value 'Link' circled in red. Other columns in the table include 'Node', 'AlertKey', and 'Location'.

6. Scroll down to **Set the Window Time Frame**.

- Set the **Start Time** to yesterday's date and the **End Time** to tomorrow's date.
- Scroll to the top of the page to click the **Save** button.

The screenshot shows the 'Window Time Frame' configuration. It displays two time intervals: 'Start Date: 06/26/2016' at 4:39:00 and 'End Date: 06/27/2016' at 4:39:00. Both times are circled in red.

7. Click the Back arrow to view the configured service.

The screenshot shows the 'Maintenance Window Editor' interface again. A large red circle highlights the back arrow icon in the top left corner of the browser window.

The green checkbox icon indicates a **Active** maintenance window.

A screenshot of a software interface titled "One Time Windows". At the top, there are buttons for "Active" (green checkmark), "Inactive" (orange circle), "Expired" (blue diamond), and "Future" (light blue circle). A filter dropdown shows "All Windows". Below the header is a table with columns: Select (all), Suppression Filter, Start Time, End Time, Time Zone, Edit, and Status. A single row is listed: "AlertGroup = 'Link'" with a start time of "Sat 25 Jun 2016 04:39:00 AM UTC" and an end time of "Mon 27 Jun 2016 04:39:00 AM UTC" in "(UTC-05:00) New York, Toronto". The "Status" column shows a green checkmark. The "Edit" button has a red circle around it, indicating it's the current action being performed.

- Double-click the GNOME Terminal icon on the desktop.



- To generate OMNIbus events, start the Simnet Probe by typing the following commands from the command line as shown:

```
cd /opt/IBM/tivoli/netcool/omnibus/probes  
.nco_p_simnet &
```

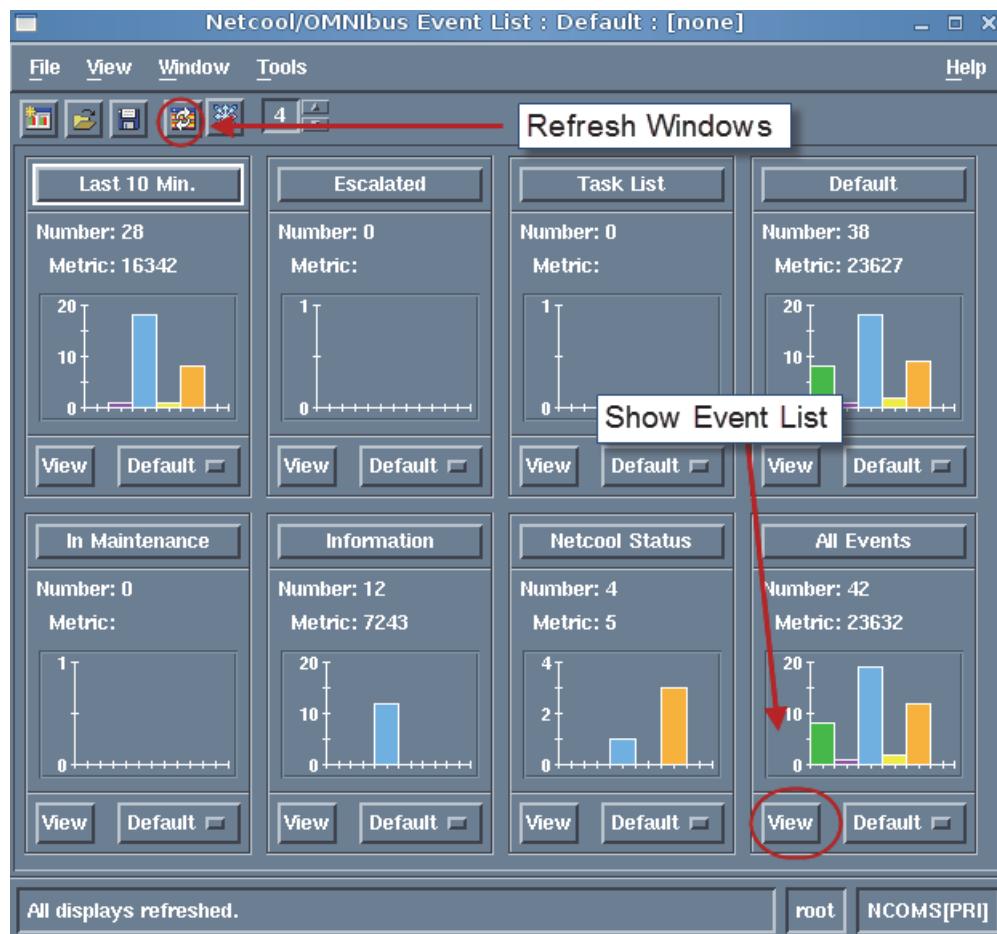
A screenshot of a terminal window titled "netcool@jazz01:~>". The user has typed the command ".nco\_p\_simnet &" into the terminal. The output shows "[1] 13249", indicating the process ID. The "nco\_p\_simnet" command is circled in red.

**Note:** Recall that **SuppressEscI** = 6 indicates that the event is in maintenance.

- To check the **SuppressEscI** field in OMNIbus type the following commands to start the OMNIbus Event Viewer.

```
cd /opt/IBM/tivoli/netcool/omnibus/bin  
.nco_event
```

11. Click the Refresh Windows button to see the metrics change. Click View to see the event list.

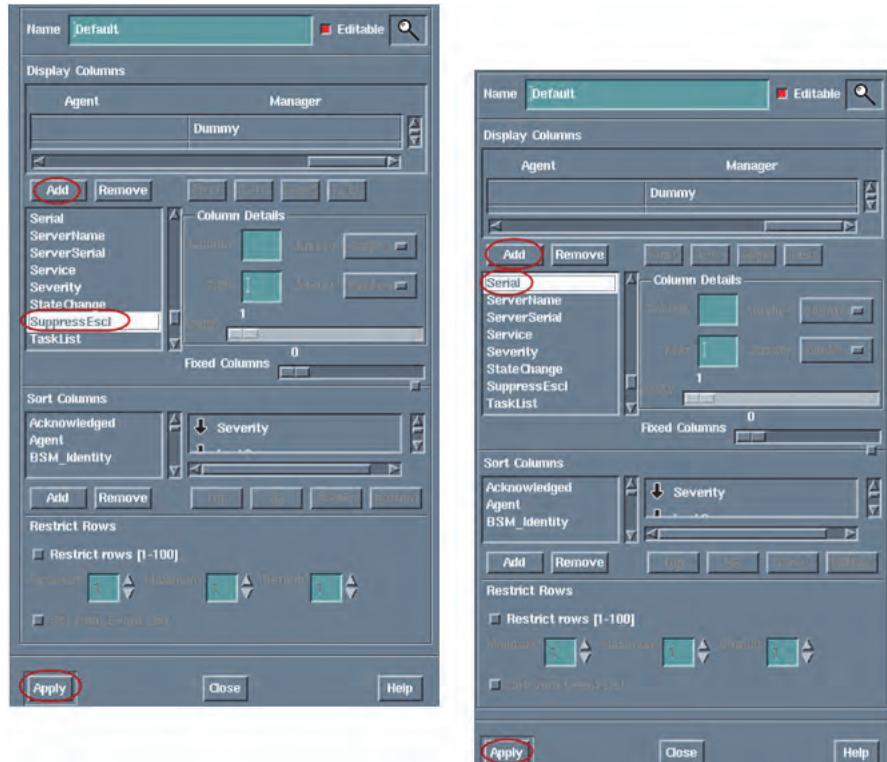


12. Click the magnifying glass.

The screenshot shows the 'File Edit View Alerts Tools' menu. The toolbar includes icons for file operations and a magnifying glass icon. The search bar contains 'All Events'. The event list table has columns: Occurrence, Count, Type, ExpireTime, and Agent. Two rows are shown:

st Occurrence	Count	Type	ExpireTime	Agent
2015 09:43:26 P	318	Information	330	OMNibus SelfMonitoring
2015 09:43:26 P	318	Information	330	OMNibus SelfMonitoring

13. Select the **SuppressEscI** and the **Serial** fields. Click **Add** to include the fields in the list and click **Apply** to make the changes.



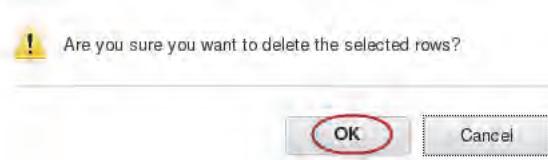
14. Scroll to the right and to the bottom to see events in Maintenance and Serial numbers.

Last Occurrence	Count	Type	ExpireTime	Agent	Manager	Suppr./EscI.	Serial
6/2016 03:49:32 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal	1013
6/2016 03:48:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal	1012
6/2016 03:48:32 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal	1011
6/2016 05:25:48 A	193	Resolution	Not Set	LinkMon	Simnet Probe	Maintenance	1030
6/2016 05:25:38 A	44	Resolution	Not Set	LinkMon	Simnet Probe	Maintenance	1164
6/2016 05:25:36 A	210	Resolution	Not Set	LinkMon	Simnet Probe	Maintenance	1023
6/2016 05:25:33 A	192	Problem	Not Set	LinkMon	Simnet Probe	Maintenance	1032
6/2016 05:25:27 A	200	Resolution	Not Set	LinkMon	Simnet Probe	Maintenance	1022
6/2016 05:25:21 A	219	Problem	Not Set	LinkMon	Simnet Probe	Maintenance	819
6/2016 05:25:20 A	37	Resolution	Not Set	LinkMon	Simnet Probe	Maintenance	1183
6/2016 05:25:11 A	95	Problem	Not Set	LinkMon	Simnet Probe	Maintenance	834
6/2016 05:25:07 A	208	Problem	Not Set	LinkMon	Simnet Probe	Maintenance	1031
6/2016 05:25:02 A	65	Problem	Not Set	LinkMon	Simnet Probe	Maintenance	1116

15. Select the table entry and click delete to remove the maintenance window.

One Time Windows			
Select (all)	Suppression Filter	Start Time	End Time
<input checked="" type="checkbox"/>	AlertGroup = 'Link'	Sat 25 Jun 2016 04:39:00 AM UTC	Mon 27 Jun 2016 04:39:00
<b>Delete</b>			

16. Confirm the delete. Click **OK**.



17. Refresh the event list to see the change. The **SuppressEscl** field should not have maintenance events.

File	Edit	View	Alerts	Tools	All Events	Default	Top [ OFF ]	Help
Last Occurrence	Count	Type	ExpireTime	Agent	Manager	Suppr./Escl.	Serial	
6/2016 05:39:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1269
6/2016 05:39:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1270
6/2016 05:38:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1267
6/2016 05:38:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1268
6/2016 05:37:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1265
6/2016 05:37:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1266
6/2016 05:36:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1263
6/2016 05:36:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1264
6/2016 05:35:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1261
6/2016 05:35:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1262
6/2016 05:34:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1259
6/2016 05:34:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1260
6/2016 05:33:33 A	1	Information	Not Set	Impact SelfMonitoring	Impact SelfMonitoring@NCICLUS	Normal		1257

## Exercise 2 Modifying Maintenance Window Management properties

1. Open the **MWM\_Properties** policy.
2. Set **clearFlag** to **FALSE**.
3. Click the **Check Syntax** icon. Click **Save**.

When the **clearFlag** variable is set to **FALSE**, the events are left tagged as in maintenance after the maintenance window expires. Click the **Check Syntax** icon. Click **Save**.



```
// Default is TRUE
// SET YOUR VALUES HERE
clearFlag = FALSE;
flagExistingEvents = TRUE;

propsContext.clearFlag = clearFlag;
propsContext.flagExistingEvents = flagExistingEvents;
}

// FUNCTION TESTING:
MWM_Properties.getProperties(mwmProps);
log(mwmProps.clearFlag);
log(mwmProps.flagExistingEvents);
***/
```

4. Stop and start the **MWMActivator** service.



5. Recreate the One Time maintenance from the earlier exercise. Set the **Window Time Frame**, set the **Start Time** to yesterday's date and the **End Time** to tomorrow's date.

The result should look as follows:

One Time Windows					Edit	Status
	Select (all)	Suppression Filter	Start Time	End Time	Time Zone	
<input type="checkbox"/>	AlertGroup = 'Link'	Sat 25 Jun 2016 05:56:00 AM UTC	Mon 27 Jun 2016 05:56:00 AM UTC	(UTC-05:00) New York, Toronto	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<a href="#">Delete</a>					

Maintenance events reappear.

- View the results in the OMNIbus event viewer.

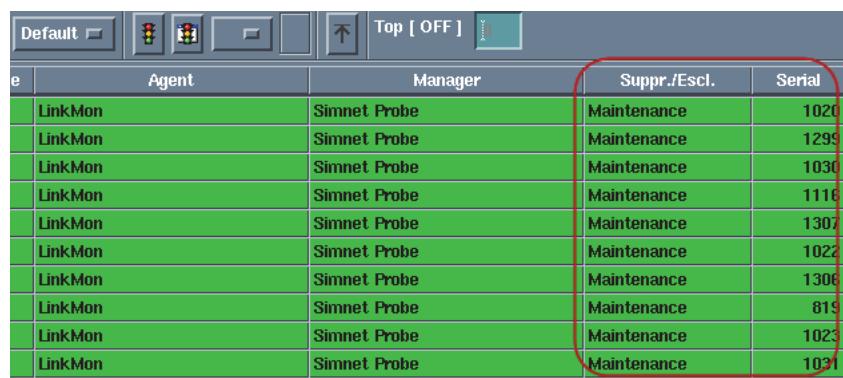


Time	Agent	Manager	Suppr./Escl.	Serial
Set	LinkMon	Simnet Probe	Maintenance	1030
Set	LinkMon	Simnet Probe	Maintenance	1022
Set	LinkMon	Simnet Probe	Maintenance	1307
Set	LinkMon	Simnet Probe	Maintenance	1023
Set	LinkMon	Simnet Probe	Maintenance	1031
Set	LinkMon	Simnet Probe	Maintenance	1299

- Force the maintenance window to end by deleting the table entry.

One Time Windows					Edit	Status
	Select (all)	Suppression Filter	Start Time	End Time	Time Zone	
<input checked="" type="checkbox"/>	AlertGroup = 'Link'	Sat 25 Jun 2016 05:56:00 AM UTC	Mon 27 Jun 2016 05:56:00 AM UTC	(UTC-05:00) New York, Toronto	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<a href="#">Delete</a>					

- Refresh and view the results in the OMNIbus event viewer. Notice that the **SuppressEscal** status is unchanged.



Time	Agent	Manager	Suppr./Escl.	Serial
Set	LinkMon	Simnet Probe	Maintenance	1020
Set	LinkMon	Simnet Probe	Maintenance	1299
Set	LinkMon	Simnet Probe	Maintenance	1030
Set	LinkMon	Simnet Probe	Maintenance	1116
Set	LinkMon	Simnet Probe	Maintenance	1307
Set	LinkMon	Simnet Probe	Maintenance	1022
Set	LinkMon	Simnet Probe	Maintenance	1306
Set	LinkMon	Simnet Probe	Maintenance	819
Set	LinkMon	Simnet Probe	Maintenance	1023
Set	LinkMon	Simnet Probe	Maintenance	1021



---

# **Unit 15 Command-line tools and self-monitoring exercises**

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

## **Exercise 1 Encrypting a policy**

In this exercise, you experiment with policy encryption. Encryption serves two primary purposes:

- Software providers or business partners can use encryption to create policies and solutions for clients and customers without handing over intellectual property.
- An administrator can use encryption to lock modifications to policies in a production or change-sensitive environment.

1. Double-click the **Gnome Terminal** icon.



2. Type the following command to navigate to the `/opt/IBM/tivoli/impact/bin` directory.

```
cd /opt/IBM/tivoli/impact/bin
```

3. Using the following encryption parameters to construct the encryption command:

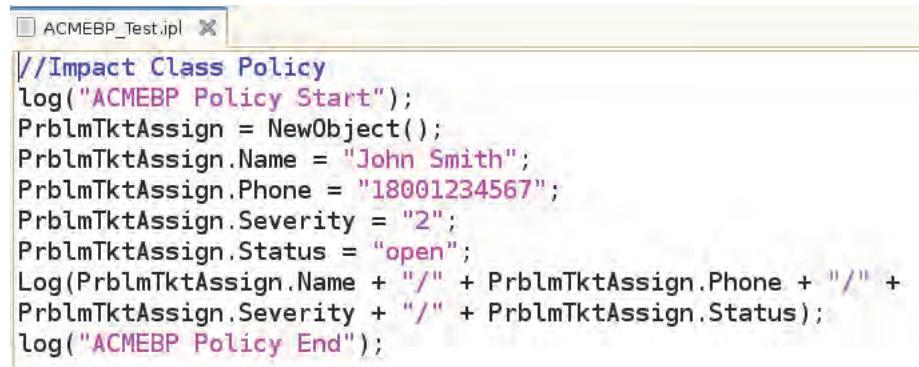
Netcool/Impact Server: **NCI**

Encryption Password: **ProdSecure1**

Input Unencrypted Policy: **ACMEBP\_Test.ipl**

Output Encrypted Policy: **ACMEBP\_Prod.ipl**

### Policy Location Directory: **labfiles**



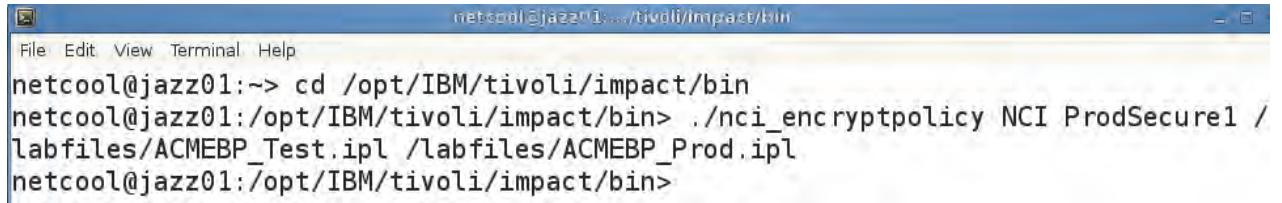
```
//Impact Class Policy
log("ACMEBP Policy Start");
PrblmTktAssign = NewObject();
PrblmTktAssign.Name = "John Smith";
PrblmTktAssign.Phone = "18001234567";
PrblmTktAssign.Severity = "2";
PrblmTktAssign.Status = "open";
Log(PrblmTktAssign.Name + "/" + PrblmTktAssign.Phone + "/" +
PrblmTktAssign.Severity + "/" + PrblmTktAssign.Status);
log("ACMEBP Policy End");
```

Figure 1 ACMEBP\_Test Policy Contents

#### 4. Run the **nci\_encryptpolicy** command.

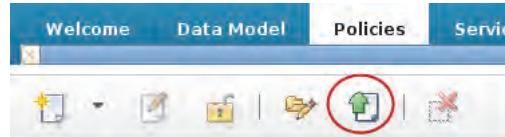
```
./nci_encryptpolicy NCI ProdSecure1 /labfiles/ACMEBP_Test.ipl
/labfiles/ACMEBP_Prod.ipl
```

The results look similar to the following example:



```
netcool@jazz01:~> cd /opt/IBM/tivoli/impact/bin
netcool@jazz01:/opt/IBM/tivoli/impact/bin> ./nci_encryptpolicy NCI ProdSecure1 /
labfiles/ACMEBP_Test.ipl /labfiles/ACMEBP_Prod.ipl
netcool@jazz01:/opt/IBM/tivoli/impact/bin>
```

#### 5. Upload the **ACMEBP\_Prod.ipl** policy into the policy editor using the **Upload a Policy File** icon



6. Locate the encrypted file. Click **Browse**.

### Upload Policy

Select a policy or parameters file to upload.

Policy file (.ipl or .js)

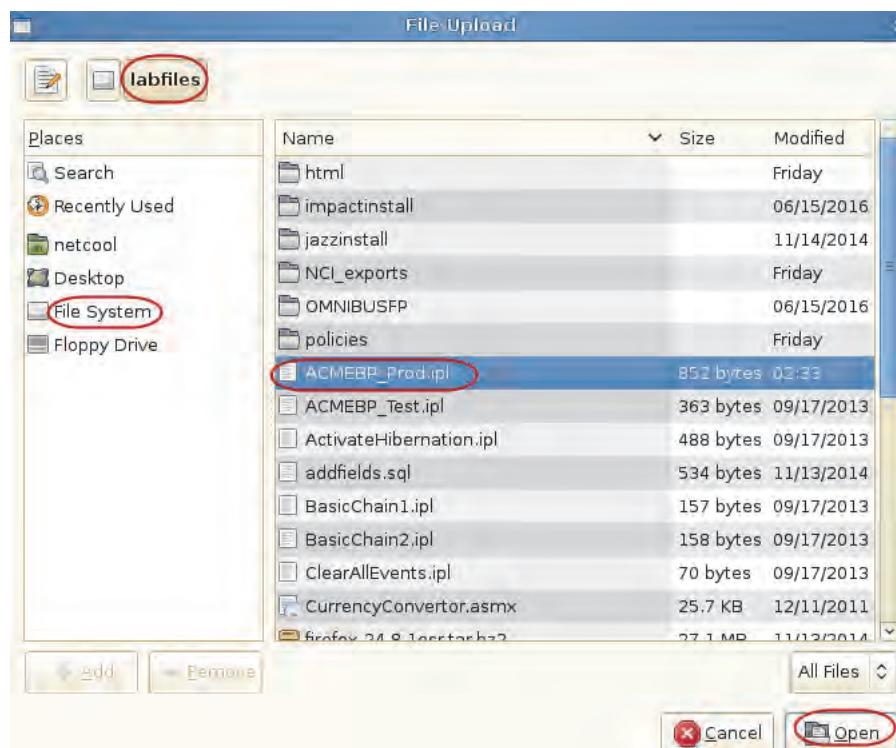
No file selected

Parameters file (.params)

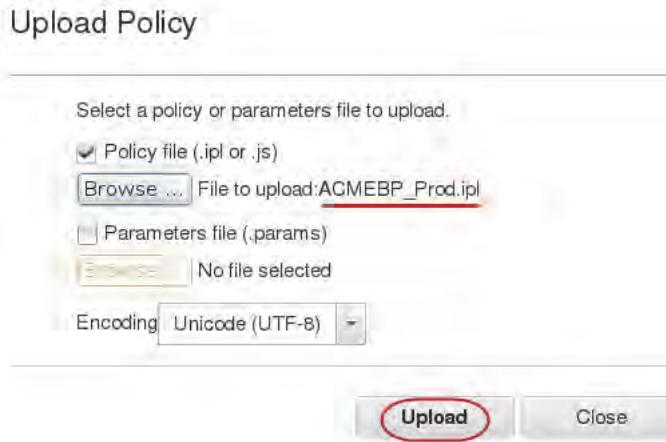
No file selected

Encoding Unicode (UTF-8)

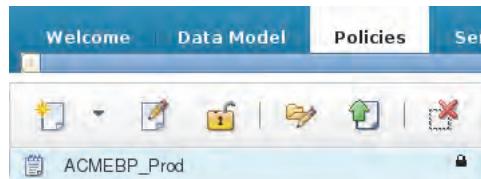
7. Click **File System>labfiles**. Select **ACMEBP\_Prod.ipl**, click **Open**.



8. Click **Upload**.



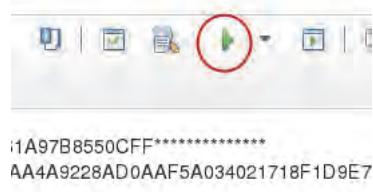
After the encrypted policy is imported, you can use it in services such as event readers the same way that unencrypted policies are used.



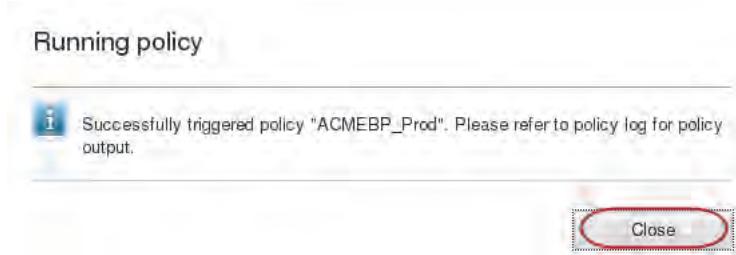
However, when viewing the policy, the user sees only encrypted code.

```
ACMEBP_Prod ✘
***** Encoded Policy *****
*****{AES}169BA82EDE12FCF47D661A97B8550CFF*****
'aes}38FFF7B8B463CF84C3333D1AF7F8B8AA4A9228AD0AAF5A034021718F1D9F7942f
```

9. Trigger the **ACMEBP\_Prod** policy from the policy editor. Click the **Run Policy** icon.



10. Click **Close** to exit the window.



11. View the results in the policy logger.

```
[PolicyLogger][ACMEBP_Prod][pool-3-thread-57]Parser log: ACMEBP Policy Start
[PolicyLogger][ACMEBP_Prod][pool-3-thread-57]Parser log: John Smith/18001234567/2/open
[PolicyLogger][ACMEBP_Prod][pool-3-thread-57]Parser log: ACMEBP Policy End
```

## Exercise 2 Enabling self-monitoring

In this exercise, you enable self-monitoring using the command-line manager and the GUI.

1. Double-click the **Gnome Terminal** icon on the desktop.



2. Ping the OMNIbus server.

```
cd /opt/IBM/tivoli/netcool/omnibus/bin
./nco_ping NCOMS &
```

The terminal window shows the command being run and its output. The output indicates that the server is available.

```
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ./nco_ping NCOMS &
[1] 6704
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> NCO_PING: Server available.

[1]+ Done                  ./nco_ping NCOMS
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin>
```

If the server is unavailable, use **nco\_objserv** to start the server.

```
cd /opt/IBM/tivoli/netcool/omnibus/bin
./nco_objserv &
```

```
File Edit View Terminal Help
netcool@jazz01:~> cd /opt/IBM/tivoli/netcool/omnibus/bin
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin> ./nco_objserv &
[1] 13556
netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin>
Netcool/OMNIbus Object Server - Version 8.1.0 64-bit

(C) Copyright IBM Corp. 1994, 2012

Server 'NCOMS' initialised - entering RUN state.

netcool@jazz01:/opt/IBM/tivoli/netcool/omnibus/bin>
```

- Start the simnet probe using the **nco\_p\_simnet** command.

```
cd /opt/IBM/tivoli/netcool/omnibus/probes
./nco_p_simnet &
```

- From a command prompt, start the command-line manager (telnet) using the following parameters:

- Server: **jazz01.tivoli.edu**
- Port: **2000**
- User: **impactadmin**
- Password: **object00**

```
File Edit View Terminal Help
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
```

- Enable Cluster Status Monitoring using the following command:

```
UPDATE Service SET EnableClusterStatus = true WHERE Name = 'SelfMonitoring';
```

The results resemble the following screen capture:

```
READY
UPDATE Service SET EnableClusterStatus = true WHERE Name = 'SelfMonitoring';
Command Executed
READY
```

6. Start the self-monitoring service in the command-line manager by typing the following command:

```
UPDATE Service SET Running = true WHERE Name = 'SelfMonitoring';
```

The results resemble the following screen capture:

```
-----+
UPDATE Service SET Running = true WHERE Name = 'SelfMonitoring';
Command Executed
READY
```

If the OMNIbus server is not running, the following error occurs:

```
READY
UPDATE Service SET Running = true WHERE Name = 'SelfMonitoring';
Execution Failed, check netcool.log for details.
READY
```

7. Enable Queue Status Monitoring using the following command:

```
UPDATE Service SET EnableQueueStatus = true WHERE Name = 'SelfMonitoring';
```

The results resemble the following screen capture:

```
UPDATE Service SET EnableQueueStatus = true WHERE Name = 'SelfMonitoring';
Command Executed
READY
```

8. Issue the following command to see if Queue Status Monitoring is enabled:

```
SELECT IsQueueStatusEnabled FROM Service WHERE Name = 'SelfMonitoring';
```

The results resemble the following screen capture:

```
READY
SELECT IsQueueStatusEnabled FROM Service WHERE Name = 'SelfMonitoring';

STARTRECORD
IsQueueStatusEnabled
-----
false
ENDRECORD

READY
```

9. Log in to OMNIbus to view self-monitoring events. Go to the following directory:

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

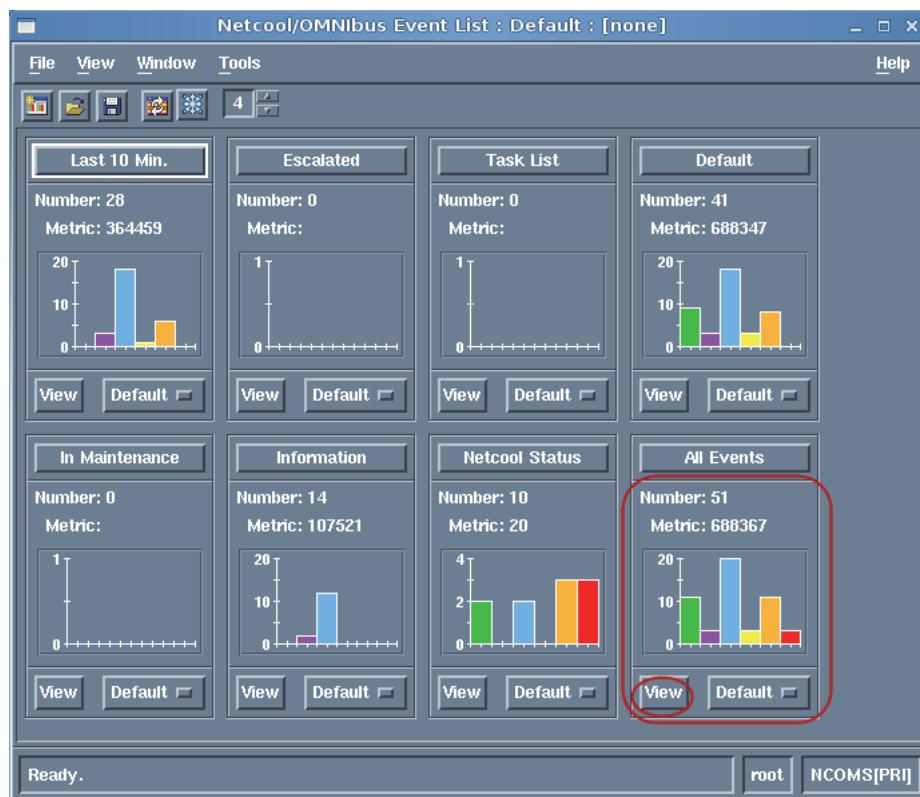
10. Start OMNIbus using the following command:

```
./nco_event
```

11. Log in with the user name **root**, password **object00**.



12. Locate the **All Events** box and click **View**.



13. Click the purple tab at the bottom of the window.

The screenshot shows a window titled 'Alerts' with a summary table. The table has columns for 'Node', 'Alert Group', and 'Summary'. Rows include: jazz01 (simnet probe), jazz01 (GET\_LOGIN\_TOKEN), jazz01 (Administrator), link1 (Link Down on port), London (Systems Machine has gone offline), link4 (Link Down on port), and Sydney (Systems Machine has gone offline). A red arrow points from the text in step 13 to the number '3' in the purple navigation bar at the bottom, which is circled in red.

Node	Alert Group	Summary
jazz01		simnet probe on jazz01: Rules file reread upon SIGHUP success
jazz01	GET_LOGIN_TOKEN	Attempt to login as secure-login from host jazz01 failed
jazz01	Administrator	Attempt to login as root from host jazz01 failed
link1	Link	Link Down on port
London	Systems	Machine has gone offline
link4	Link	Link Down on port
Sydney	Systems	Machine has gone offline

10      3      20      3      12      3      No Events

14. View the events, and locate the entry for **Queue Status** as displayed:

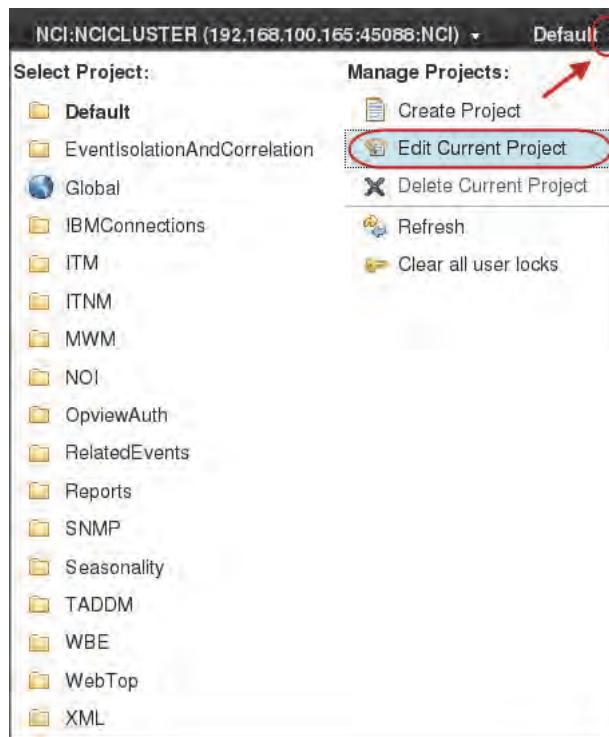
The screenshot shows a window titled 'Events' with a summary table. The table has columns for 'Node', 'Alert Group', and 'Summary'. Rows include: jazz01.tivoli.edu (QueueStatus EventProcessor : QueueSize: 0) and jazz01.tivoli.edu (QueueStatus ProcessNewSeasonalityReport : QueueSize: 0). The toolbar above the table includes icons for File, Edit, View, Alerts, Tools, and Help, along with various monitoring status indicators.

Node	Alert Group	Summary
jazz01.tivoli.edu	QueueStatus	EventProcessor : QueueSize: 0 (Min: 0 Max: 0) DeltaQueue: 0
jazz01.tivoli.edu	QueueStatus	ProcessNewSeasonalityReport : QueueSize: 0 (Min: 0 Max: 0) DeltaQueue: 0

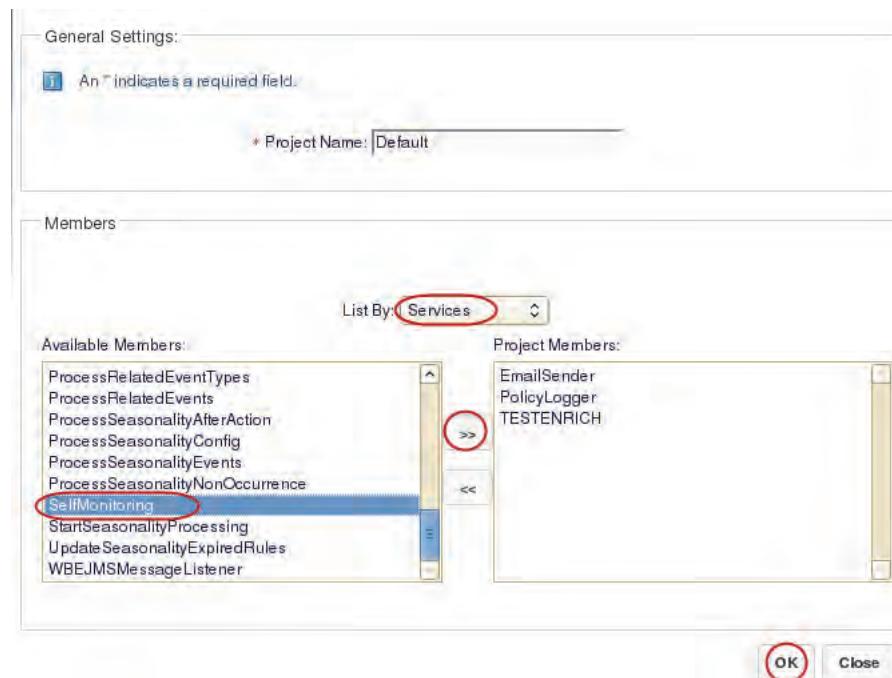
The self-monitoring service sends Cluster Status events to the ObjectServer under the following conditions:

- A secondary cluster member becomes unavailable.
- A secondary cluster member assumes the primary role.
- A secondary cluster member detects that the primary has become unavailable.
- A secondary cluster member accepts a new primary server.

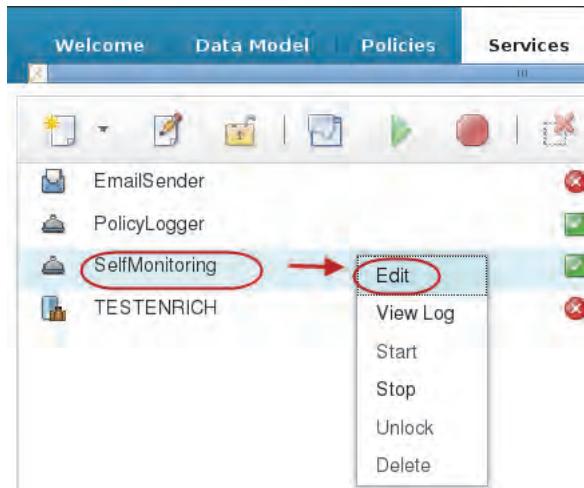
15. To add the **SelfMonitoring** service to the **Default** project, select **Edit Current Project**.



16. Add the **SelfMonitoring** service to the **Default** project. Select **List By Services** and select **SelfMonitoring** from **Available Members**. Click the arrows (>>) to add to **Project Members**. Click **Ok**.



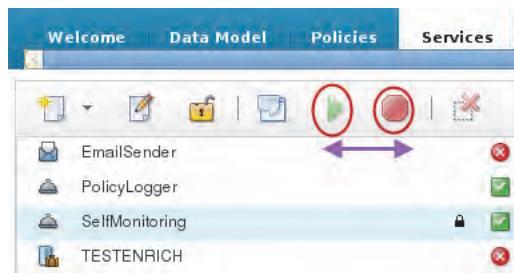
17. Click the **Services** tab. Edit **SelfMonitoring**.



18. Edit the **SelfMonitoring** service configuration screen as shown in the following screen capture.  
Click the **Save service** icon.

**Note:** The events are generated every 60 seconds.

19. Stop and start the **SelfMonitoring** service from the GUI.



20. View the changes in the OMNIbus event viewer. Notice the new events for **MemoryStatus**.

QueueStatus	EventProcessor : QueueSize: 0 (Min: 0 Max: 0) DeltaQueue: 0
MemoryStatus	Impact's Heap using 118M out of 1200M, Free System Memory Available: 3183M
QueueStatus	EventProcessor : QueueSize: 0 (Min: 0 Max: 0) DeltaQueue: 0
MemoryStatus	Impact's Heap using 233M out of 1200M, Free System Memory Available: 3181M
QueueStatus	EventProcessor : QueueSize: 0 (Min: 0 Max: 0) DeltaQueue: 0
MemoryStatus	Impact's Heap using 197M out of 1200M, Free System Memory Available: 3182M
QueueStatus	EventProcessor : QueueSize: 0 (Min: 0 Max: 0) DeltaQueue: 0

21. Type **exit** to close the telnet connection.

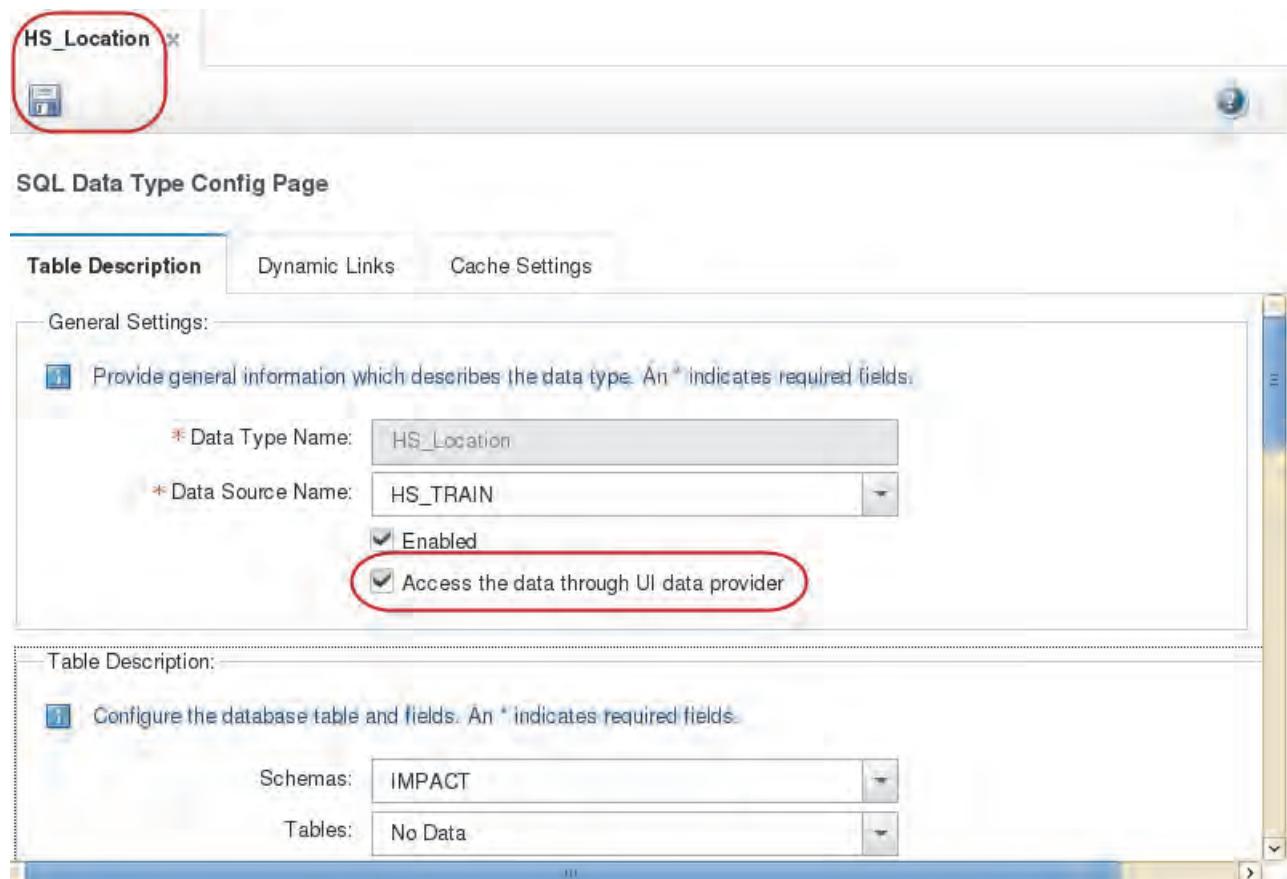
```
READY
exit
Connection closed by foreign host.
```

# Unit 16 The Netcool/Impact UI data provider exercises

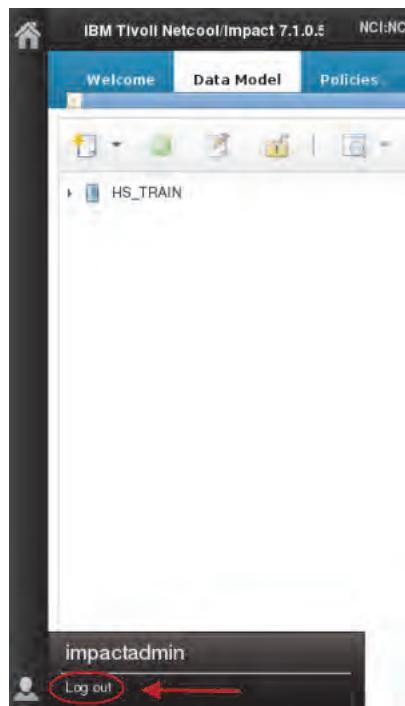
In this unit, a Netcool/Impact policy passes data to the IBM Service Management Hub to populate dashboard widgets.

## Exercise 1 Configuring a data source

1. Edit the **HS\_Location** Data Type (HS\_TRAIN Data Source) and select the check box for **Access the data through UI data provider**. Click **Save**.



2. Click **Log out** to exit Impact.



# Exercise 2 Building a widget

1. Open a GNOME Terminal on the desktop and type the startup command for JazzSM.



```
cd /home/netcool/IBM/JazzSM/profile/bin  
./startServer.sh server1
```

Wait for the JazzSM server to start.

2. Open a Firefox browser and connect to the following URL:

<http://jazz01.tivoli.edu:17310/ibm/console>

3. Accept the connection. Click **I understand the Risks**.



## This Connection is Untrusted

You have asked Firefox to connect securely to **jazz01.tivoli.edu:17311**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

### What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

[Get me out of here!](#)

### Technical Details

### I Understand the Risks

4. Click **Add Exception**.

### I Understand the Risks

If you understand what's going on, you can tell Firefox to start trusting this site's identification. Even if you trust the site, this error could mean that someone is tampering with your connection.

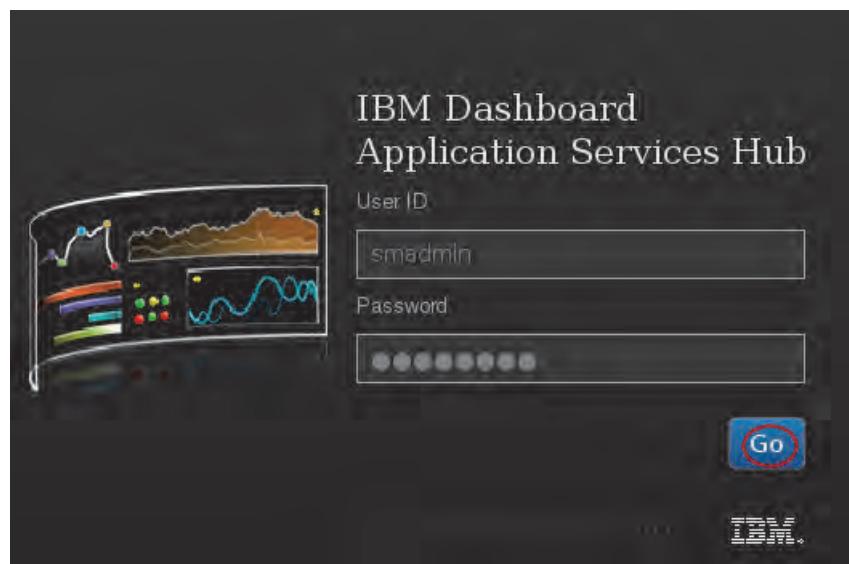
Don't add an exception unless you know there's a good reason why this site doesn't use trusted identification.

[Add Exception...](#)

5. Add security exception. Click **Confirm Security Exception**.



6. Type **smadmin** and **object00** as the user ID and password.



The welcome screen provides options for videos, tours, and community connections.

## Learn. Explore. Connect.



7. Create the connection to Impact. Click the console settings icon. Click **Connections**.



Console Settings

**General**

- Catalogs
- Connections**
- Console Preference Profiles
- Export Wizard
- Pages
- Widgets
- Views
- WebSphere Administrative Console
- Console Integrations
- Console Properties

**Roles**

- Group Roles
- Roles
- User Roles

Computer IBM Dashboard Appl..

8. Establish a connection. Click the **Create a new Remote Provider** icon.

## Connections

The connection manager allows you to configure the local and remote connections for this computer. The list below displays the available connections.

To create a new remote connection, click on the 'Create new remote provider' icon. To edit an existing connection, either click on the 'Edit existing provider' button, or right-click on a connection and select the 'Edit' menu option. To delete a connection, select the connection and click on the 'Delete remote provider' button, or right-click on the connection and select the 'Delete' menu option.



Name	Type	Description	Connection	ID
Tivoli Directory Integrator	TDI	TDI Generic Data Provider (1.0.37)	Local	TDI
tip	tip	Tivoli Integrated Portal Data Provider	Static	tip

9. Enter the connection information as displayed in the example. Click **Search**.

- Host Name: jazz01.tivoli.edu
- Port:16310
- Name: impactadmin
- Password: object00
- Confirm Password: object00

## Connections

Change the name or description to modify this remote connection and click 'OK'.

### Server information

* Protocol:	* Host name:	* Port:
HTTP	jazz01.tivoli.edu	16310
* Path:	/ibm/tivoli/rest	
<input type="checkbox"/> Connection goes through a firewall		
Firewall address	Firewall port	

Use the following credentials to query the remote data providers

* Name:	* Password:
impactadmin	*****
* Confirm password:	
*****	

**Search**

10. Scroll down to select the table entry. Click **Impact\_NCICLUSTER**.

No filter applied			
Name	Description	Type	Provider ID
Impact_NCICLUSTER		Impact_NCICLUSTER	Impact_NCICLUSTER
Total: 1 Selected: 0			

The connection information will automatically display as shown in the example.

11. Click **OK**.

- Name: Impact\_NCICLUSTER
- Description: Impact\_NCICLUSTER

**Connection information**

\* Name:  
Impact\_NCICLUSTER

Description:  
Impact\_NCICLUSTER

\* Provider ID:  
Impact\_NCICLUSTER.jazz01.tivoli.edu

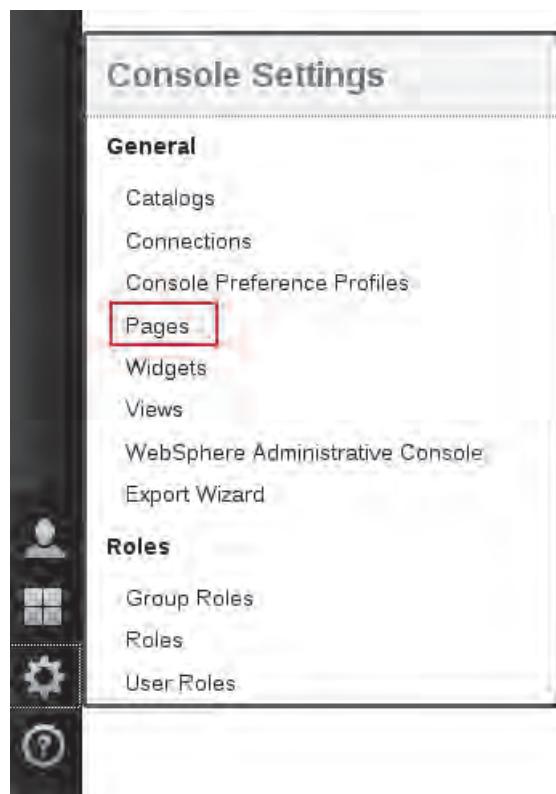
Use the credentials of the user (requires SSO Configuration)

**OK** **Cancel**

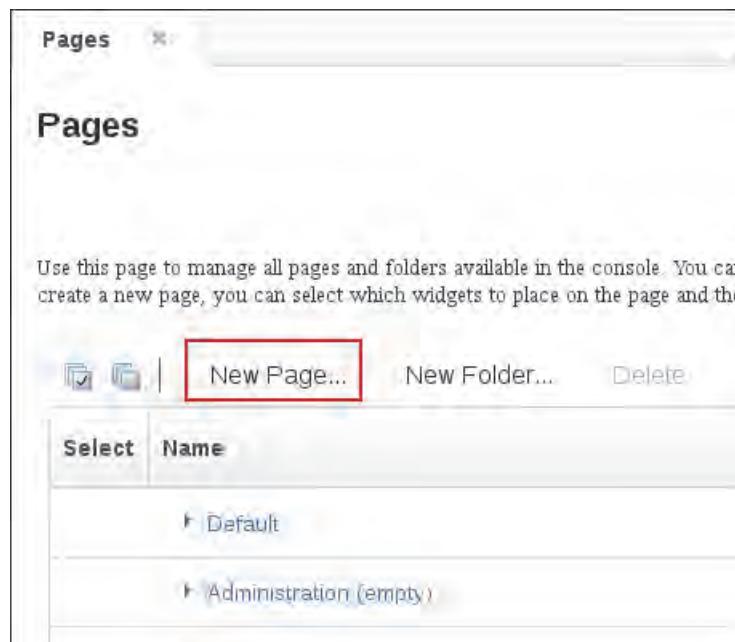
12. Examine the **Impact\_NCICLUSTER** table entry. Verify it has **Working** as status.

Name	Type	Description	Connection	ID	Status
Impact_NCICLUSTER	Impact_NCICLUSTER	Impact_NCICLUSTER	Remote	Impact_NCICLUSTER.jazz01.tivoli.edu	Working
Tivoli Directory Integrator	TDI	TDI Generic Data Provider	Local	TDI	No data received
tip	tip	Tivoli Integrated Portal	Static	tip	Working

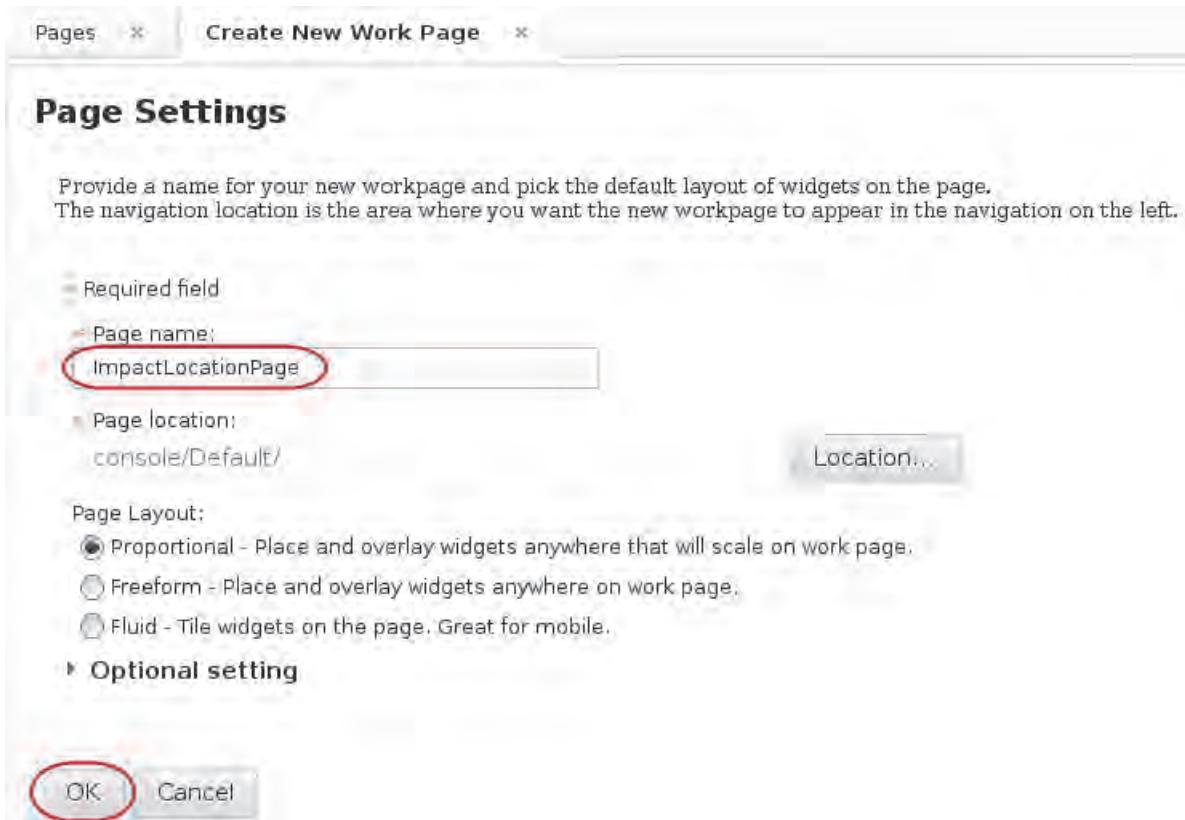
13. Build a new page by clicking **Console Settings > Pages** as shown.



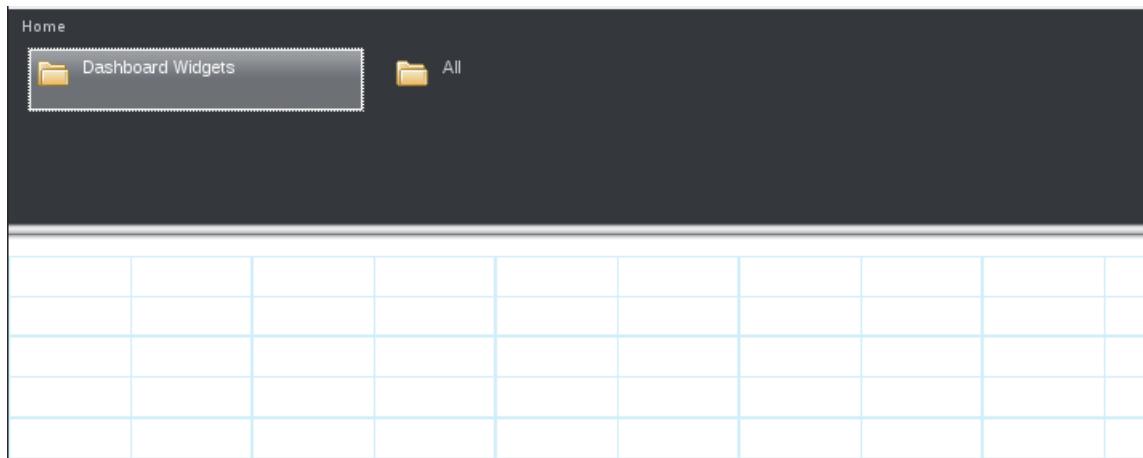
14. Click **New Page**.



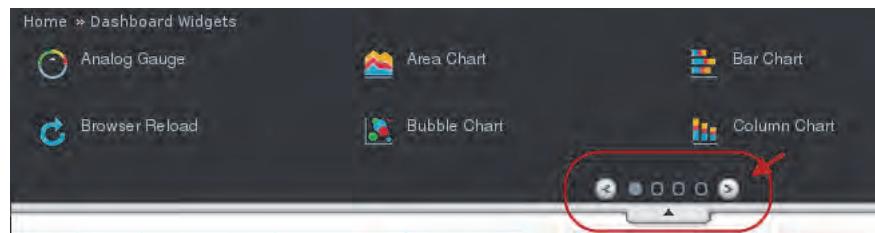
15. Type **ImpactLocationPage** in the **Page name** field and click **OK**.



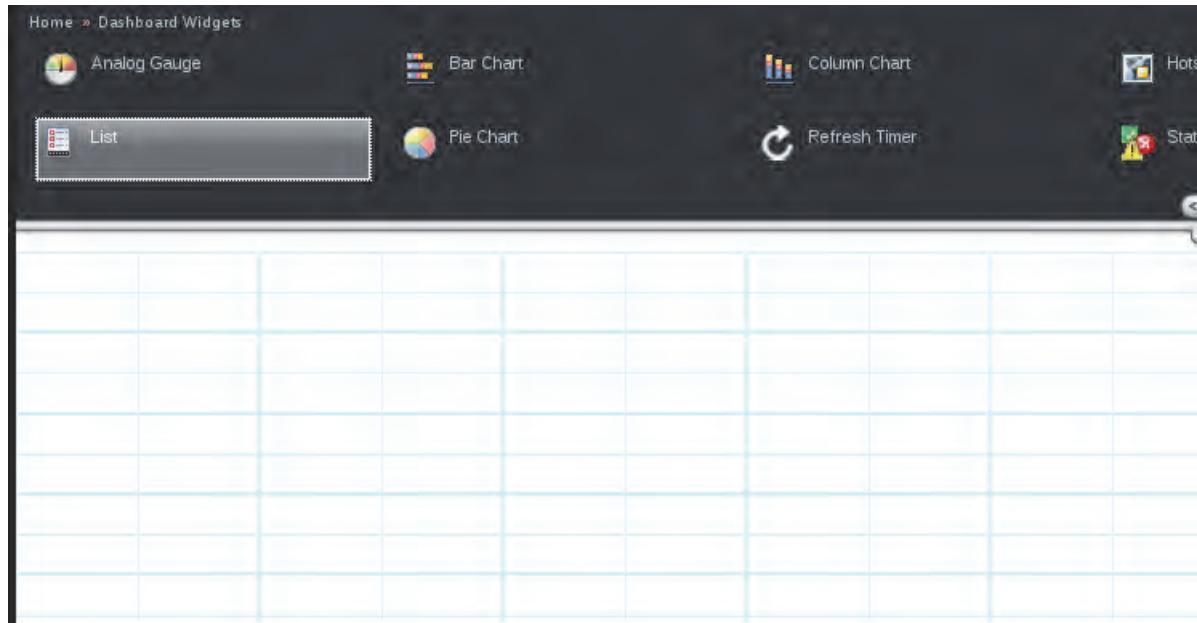
16. Click **Dashboard Widgets**.



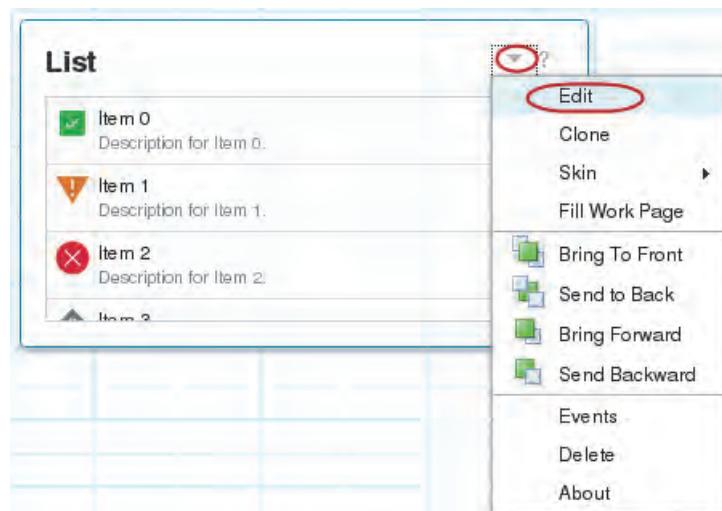
17. Find the **List** widget. Use the arrow to scroll right.



18. Drag the **List** widget to the page.



19. Click the arrow to edit the widget.



20. Type **HS\_Location** in the search field. Click **HS\_Location** in the list to select the data set.

### List

**Select a Dataset**

Enter a term and press Search to see datasets. To see all, just press Search:

After searching, select a dataset from the search results below to continue:

**Provider: Impact\_NCICLUSTER > Datasource: HS\_TRAIN**

HS_Location	>
Datatype from Datasource: HS_TRAIN, name of the datatype: HS_Location, with table name: IMPACTLOCATION	

Datasets Found: 1

- Messages: 0

21. Map the visualization attributes to the data set columns as shown. Click **OK**.

### List

**Visualization Settings**

Messages: 0

**\*Label**  
You can choose any value

**Status**  
Status type expected

**Description**  
You can choose any value

**Timestamp**  
Date or Date/Time expected. If selected, will be used as default group-by

LOCATION
None
LOCATIONID
None

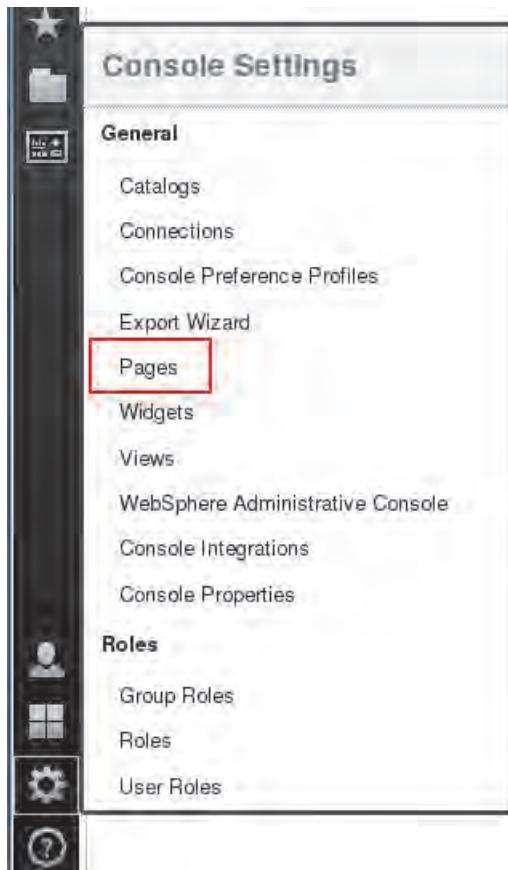
The resulting data is as shown in this screen capture.

List

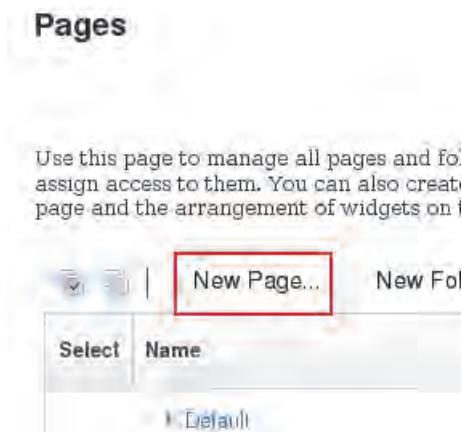
US:New York:Wall Street	1
UK:London:Bridas	2
US:San Francisco:Thompson	3
US:Dallas:Disraeli	4
US:Chicago:Gladstone	5
Australia:Sydney:Watershed	6

# Exercise 3 Opening the Operator View using the Web Widget

1. Build a new page by clicking **Console Settings > Pages** as shown.



2. Click **New Page**.



3. Type **OperatorViewData** in the **Page name** field and click **OK**.

## Page Settings

Provide a name for your new workpage and pick the default layout of widgets on the page.  
The navigation location is the area where you want the new workpage to appear in the navigation on the left.

\* Required field

\* Page name:  
  

Page location:  
console/Default/ Location...

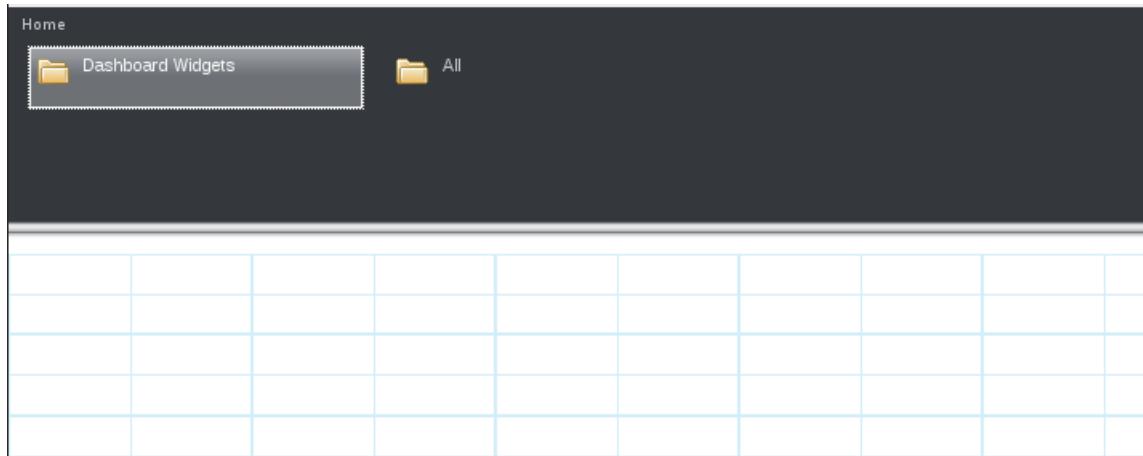
Page Layout:

Proportional - Place and overlay widgets anywhere that will scale on work page.  
 Freeform - Place and overlay widgets anywhere on work page.  
 Fluid - Tile widgets on the page. Great for mobile.

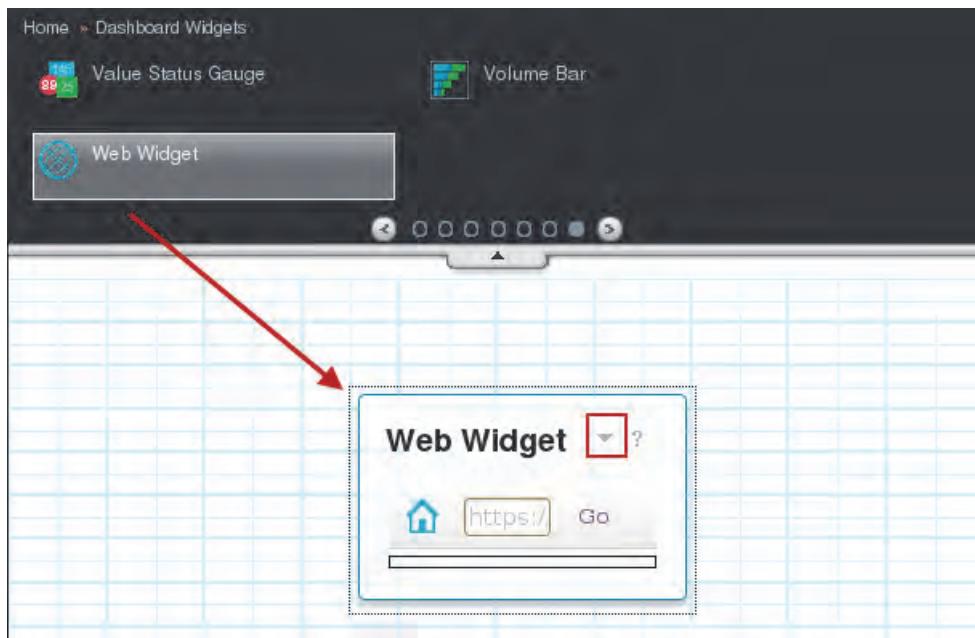
▶ **Optional setting**

OK Cancel

4. Click **Dashboard Widgets** to expand it.



5. Scroll to the right. Drag the **Web Widget** to the page.



6. Click the arrow, and select **Edit**.



7. Enter the following data:

Widget Title: Operator View

Home Page (Operator View URL) :

<https://jazz01.tivoli.edu:16311/opview/displays/NCICLUSTER-MyView.html>

## Web Widget

Use this page to edit shared settings for Web Widget users.

You can set the widget title, the default Web address, custom help page URL, and the HTML iFrame name.

The HTML iFrame name is an advanced option to allow a displayed page to be modified using a scripting language.

Widget title:

Home page (use http:// for remote hosts or relative URLs for the dashboard server):

Help page:

HTML iFrame name:

Note: An iFrame name must be unique. Do not use the same iFrame name for multiple Web widget iFrames.

8. Scroll down to click **Save**.



The Web Widget shows in the Operator View URL.

The screenshot displays the "Server Administration Dashboard". At the top right, the date and time are shown as Jun 27, 2016 04:25 AM. The dashboard is divided into several sections:

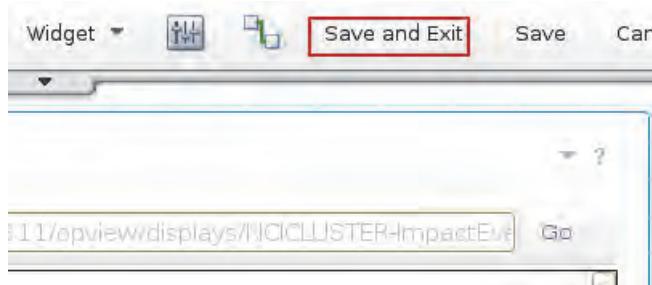
- Server Info:** Contains details like Server Name (impactserver1), IP Address (192.168.1.119), MAC Address (00:A0:C9:14:C8:27), Location (Atlanta - Peachtree), Server Type (Sun), Operating System (Solaris 10), Disk (2 @ 73 GB), Memory (4 GB), and Processors (2 @ 800 GHz).
- Contact Info:** Shows Customer (ACME TELCOM) and Customer Contact (saadmin@vacmtel.com) details.
- Ticket Info:** Lists two tickets:

Ticket Number	Short Description	Last Modified	Status	Action
CPT2007-0000188	Device is unreachable. Device is unplugged.	1/25/2012 03:25:06 PM	CLOSED	<a href="#">view</a>
CPT2007-0000169	Attempts to ping server have been unsuccessful.	1/25/2011 03:25:06 PM	OPEN	<a href="#">view</a>
- Application Info:** Lists installed applications: Netcool Object Server, Netcool Mtrrpad Probe, Netcool Syslog Probe, and Apache Tomcat.
- Full Ticket View:** Provides detailed information for the first ticket (CPT2007-0000188):

Ticket Number:	CPT2007-0000188	Device:	192.168.1.119
Status:	CLOSED	Category:	Server
Last Update:	1/25/2012 03:25:06 PM	Opened By:	Support Desk
Assigned To:	Server Adminperson		

**Full Description:** Manual ping was attempted with no success.

9. Click **Save and Exit**.

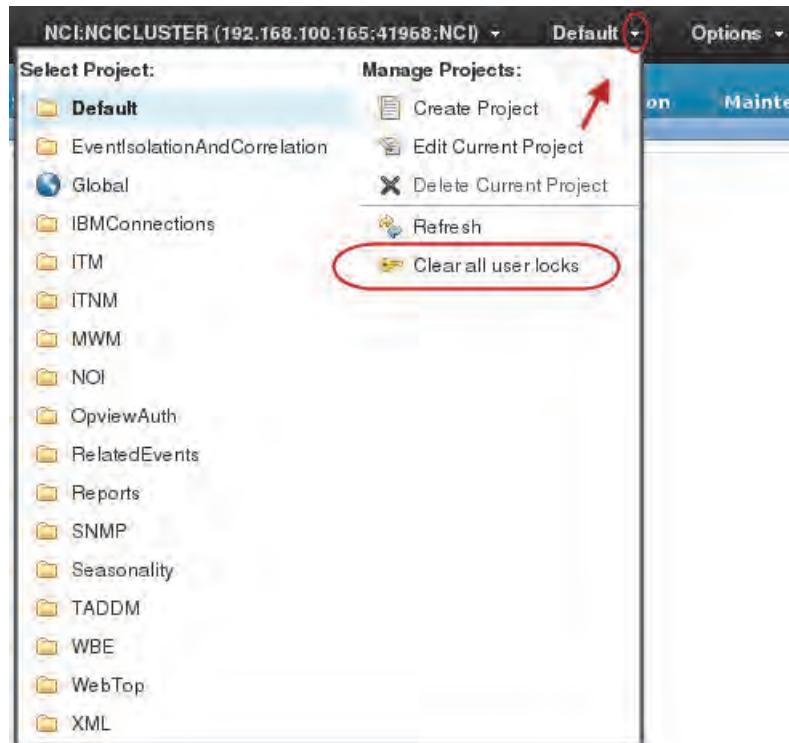


# Unit 17 Server utilities exercises

In these exercises, you work with server utilities.

## Exercise 1 Exporting Netcool/Impact server

1. Click the arrow next to **Default**. Select **Clear All user locks** to unlock the objects in Netcool/Impact server.



2. Click **OK** to close the window.



3. To export the Netcool/Impact objects from the server, open a Gnome Terminal and type the following commands.

```
cd /opt/IBM/tivoli/impact/bin/  
.nci_export NCI /labfiles/NCI_exports/
```

```
netcool@jazz01:/opt/IBM/tivoli/impact/bin> ./nci_export NCI /labfiles/NCI_exports  
s/  
nci_export of NCI to /labfiles/NCI_exports completed.  
netcool@jazz01:/opt/IBM/tivoli/impact/bin>
```



**Note:** **NCI\_exports** is a directory. Make sure that the **NCI\_exports** directory does not exist before you run the **export** command; otherwise the command will fail.

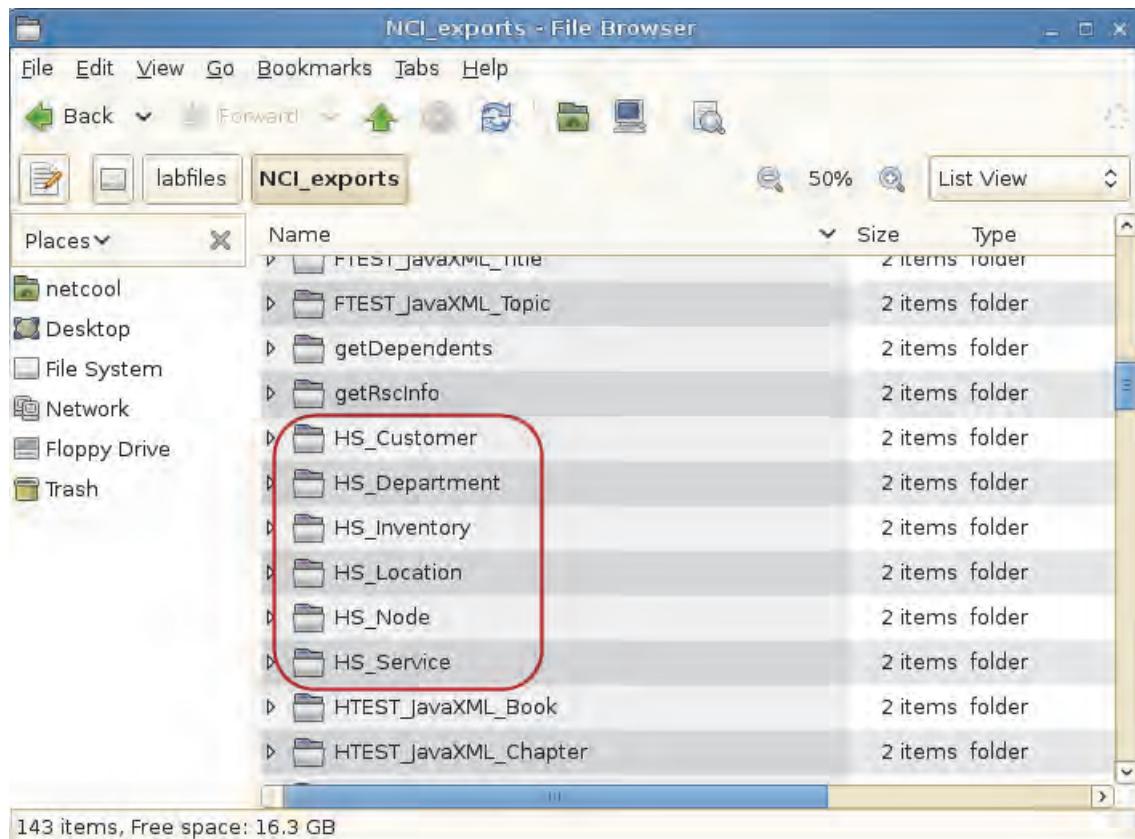
4. Open a file browser on your desktop.



5. View the contents of the labfiles/NCI\_export directory.



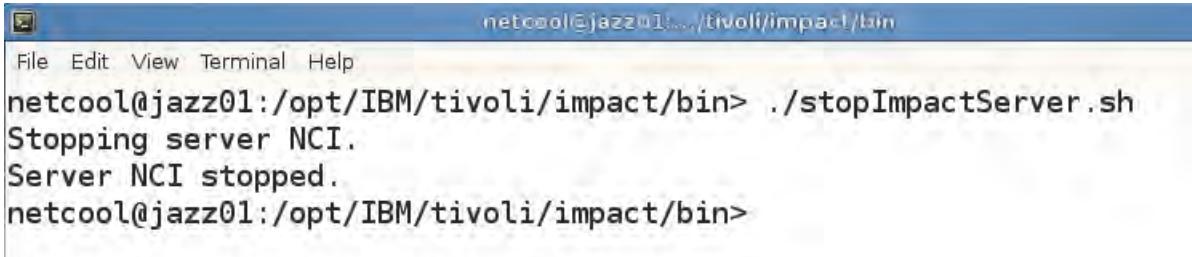
6. Verify that your previously created policies, data sources, and other elements exist in the directory.



7. Stop the Impact server using the following commands.

```
cd /opt/IBM/tivoli/impact/bin/  
../stopImpactServer.sh
```

8. Look for the following messages in the output:



A screenshot of a terminal window titled "netcool@jazz01:/opt/IBM/tivoli/impact/bin". The window contains the following text:

```
File Edit View Terminal Help  
netcool@jazz01:/opt/IBM/tivoli/impact/bin> ./stopImpactServer.sh  
Stopping server NCI.  
Server NCI stopped.  
netcool@jazz01:/opt/IBM/tivoli/impact/bin>
```

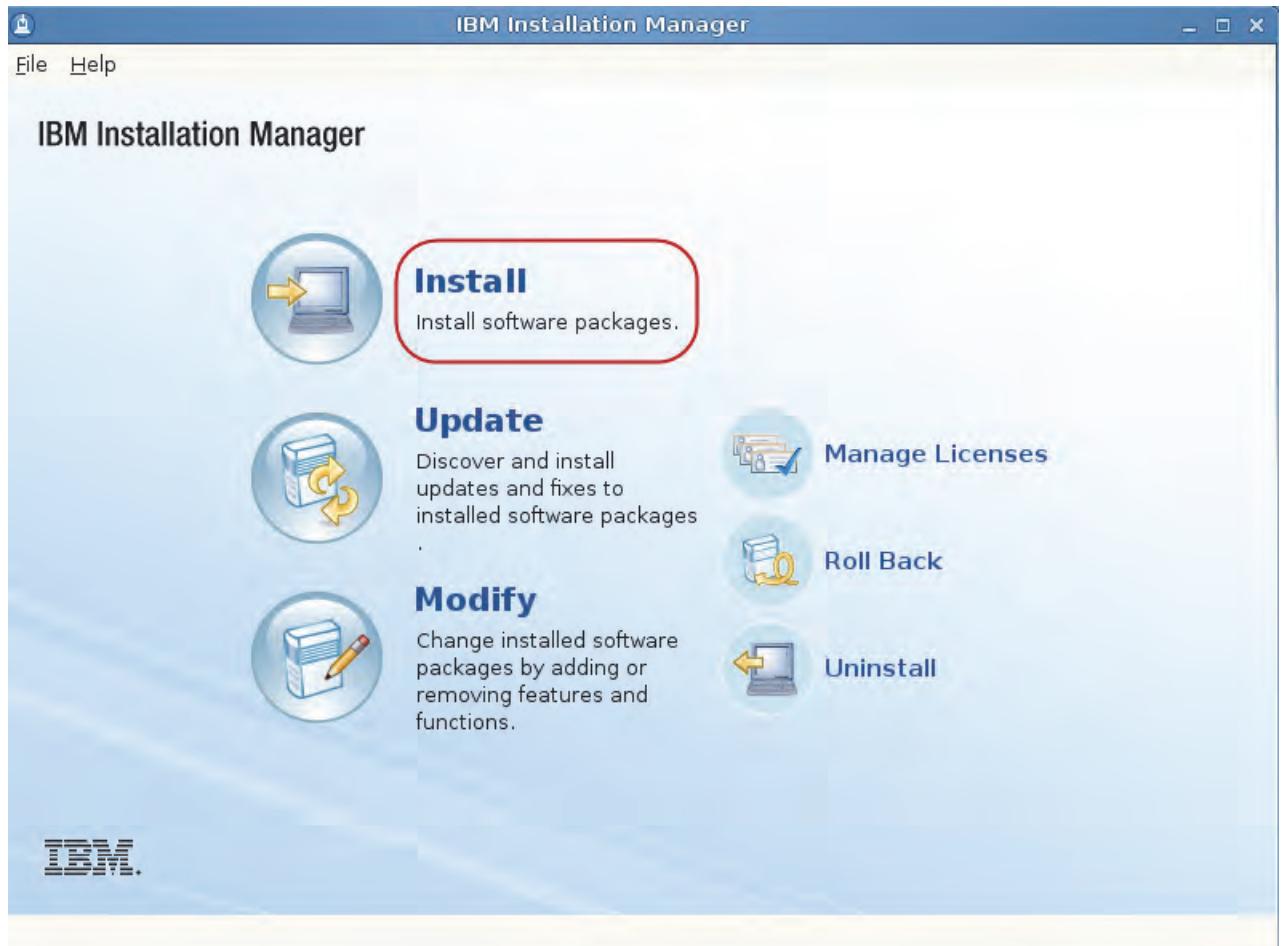
# Exercise 2 Creating a second server instance

1. Use the following command to start the Installation Manager tool:

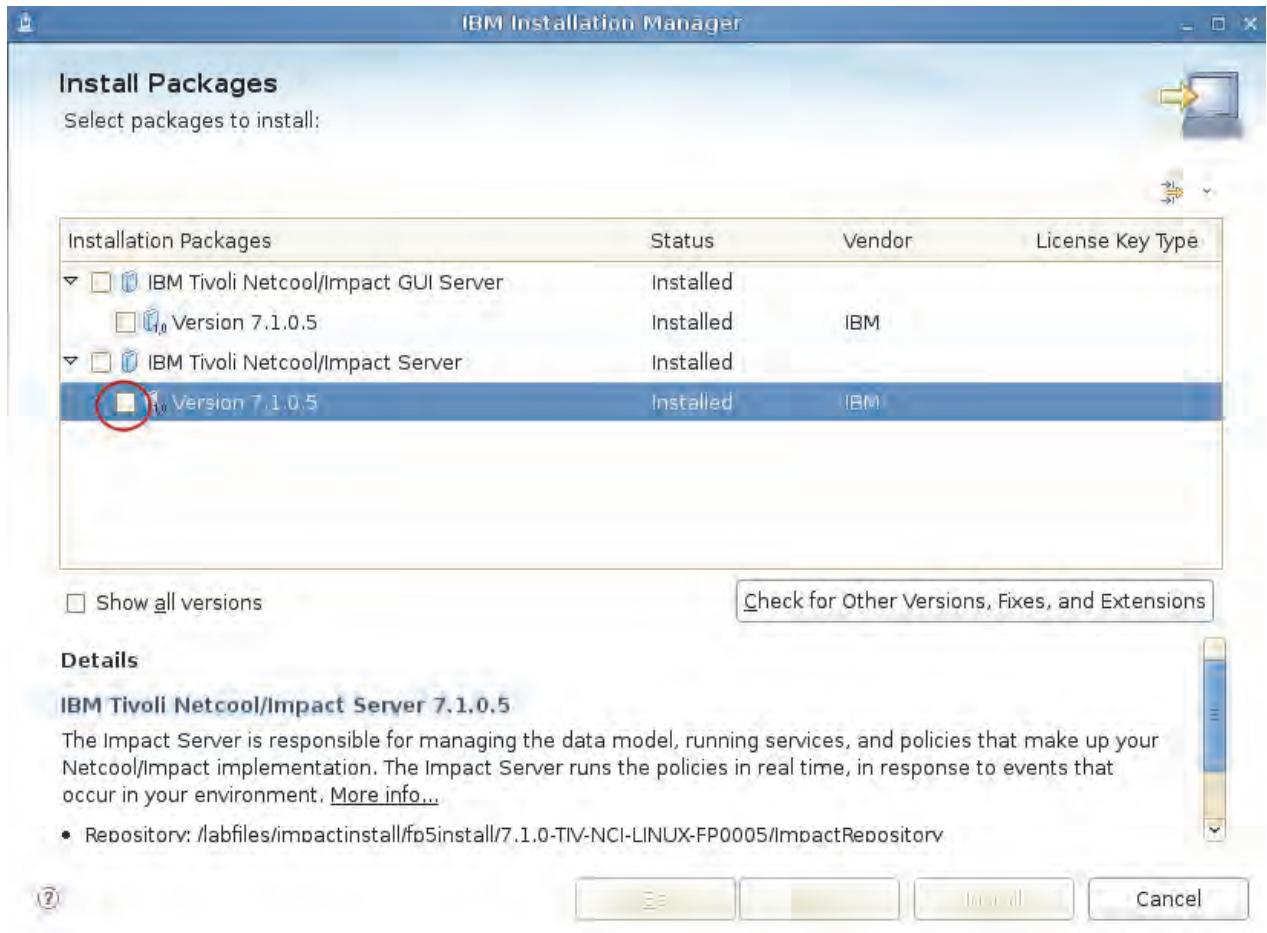
```
cd /home/netcool/IBM/InstallationManager/eclipse/  
.IBMIM
```

2. Add the installation repositories. Select **File > Preferences** add the following package:

```
/labfiles/impactinstall/ImpactRepository/disk1/diskTag.inf
```



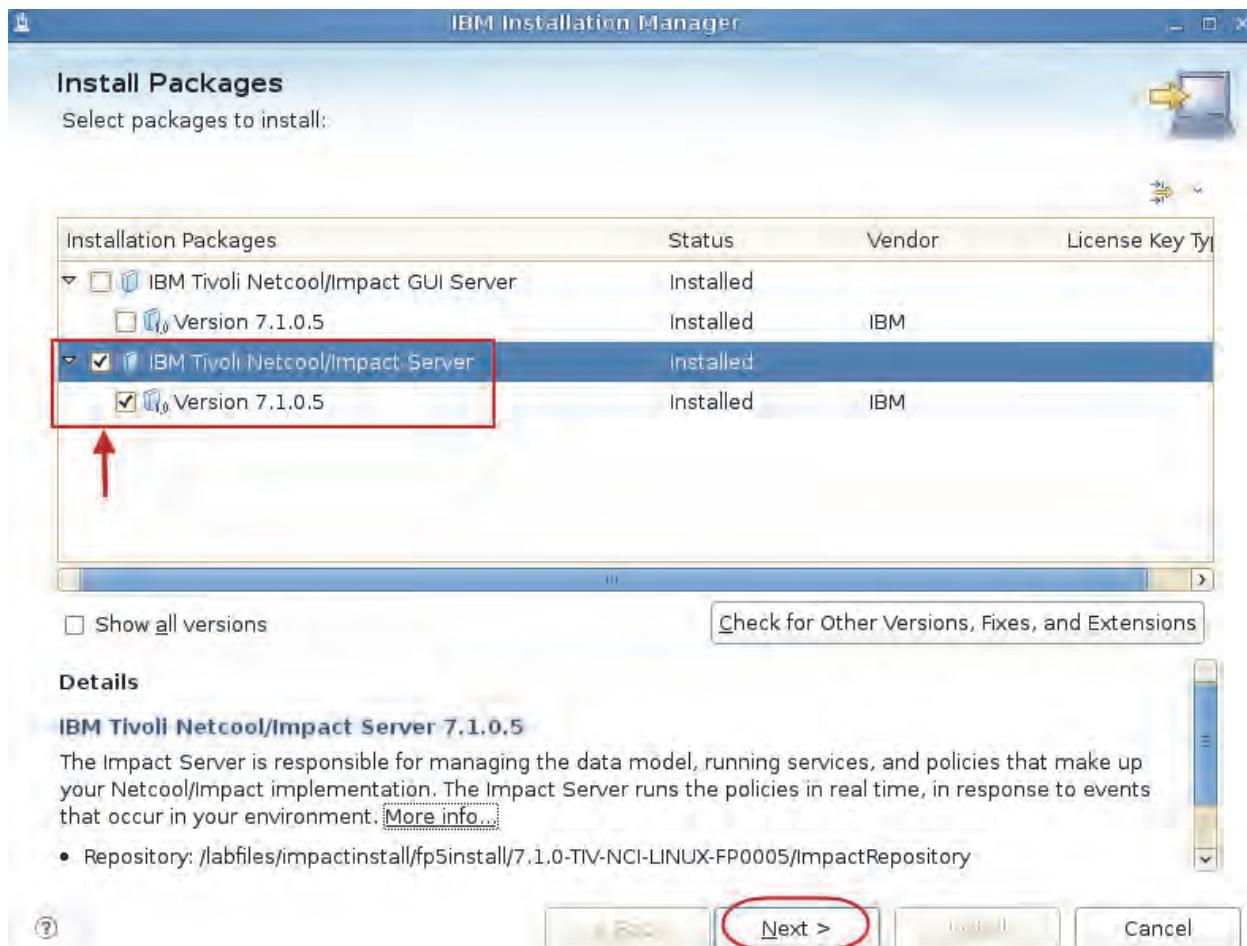
3. Select IBM Tivoli Netcool/Impact Server version 7.1.0.5.



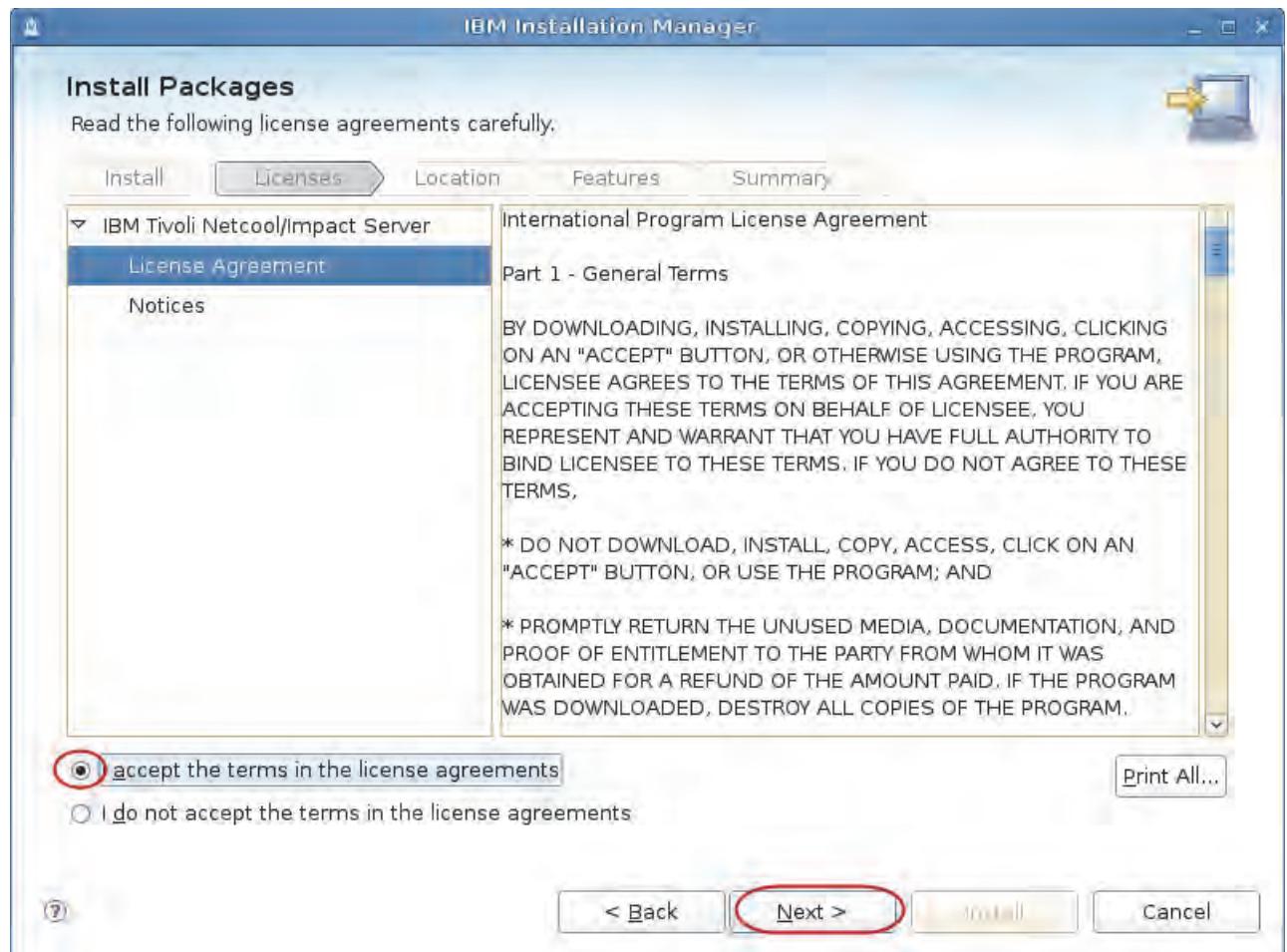
4. Click **Continue** to install the package to a new group.



5. Install packages. Click **Next**.

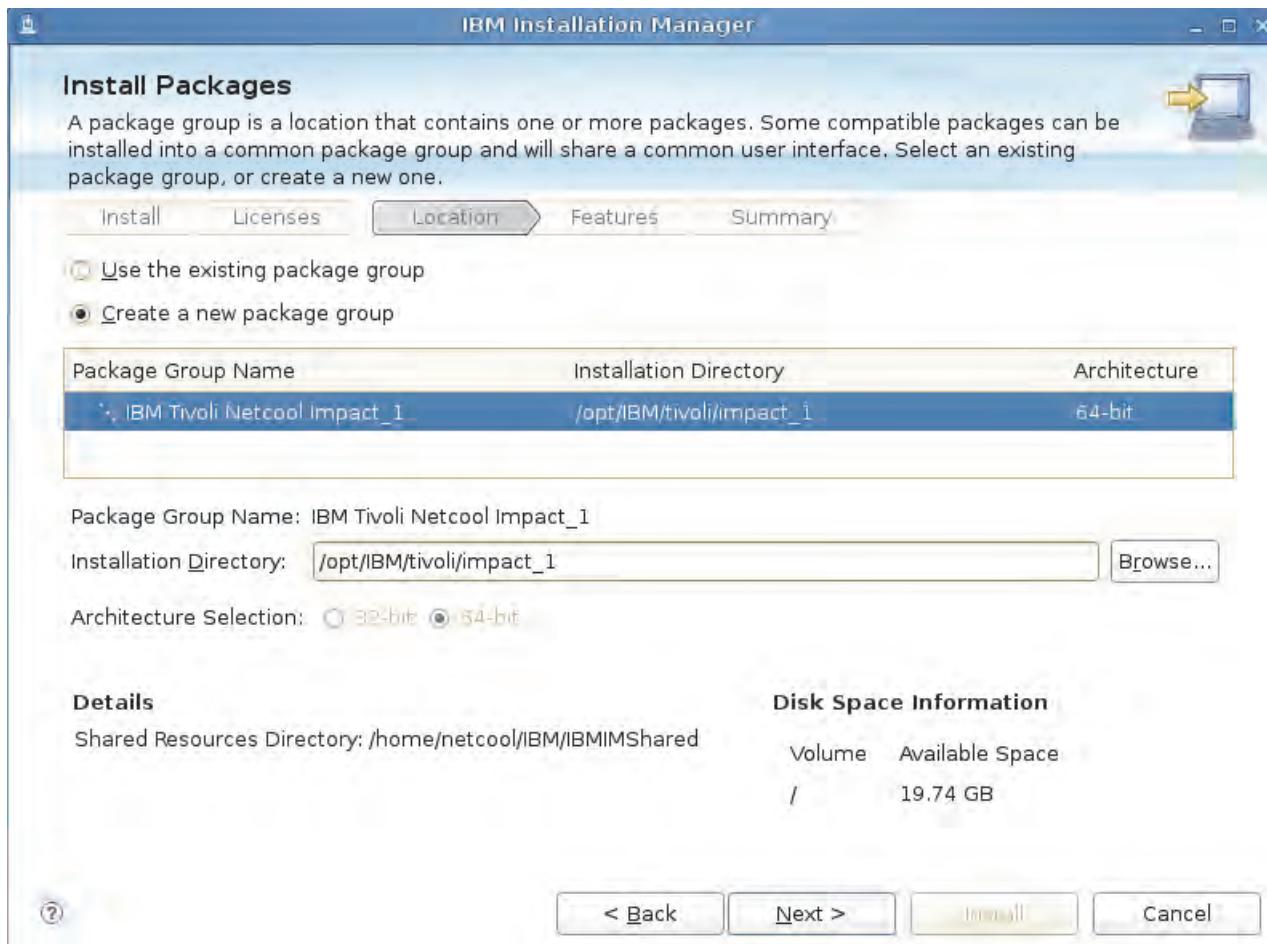


6. Accept the license agreement. Click **Next**.

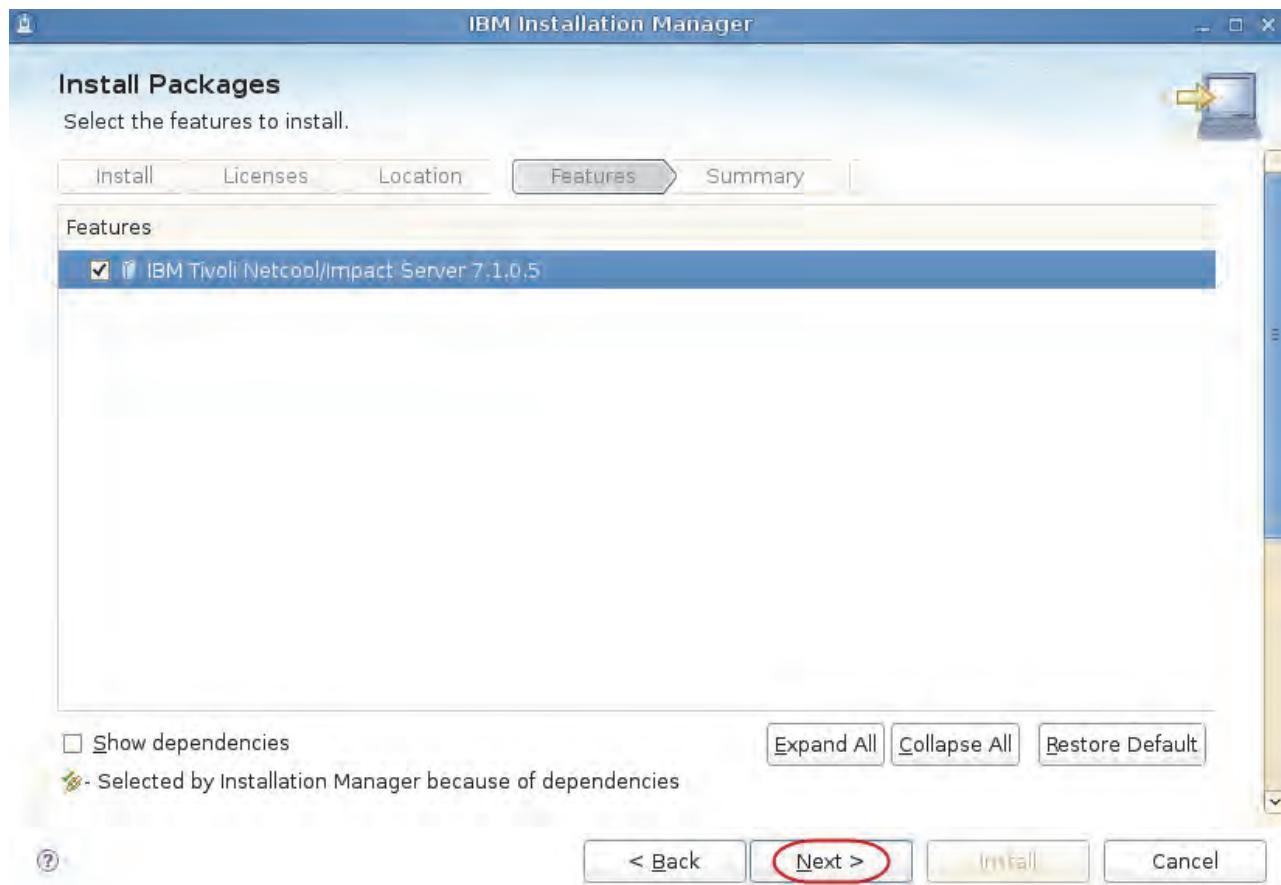


Notice the Installation Directory.

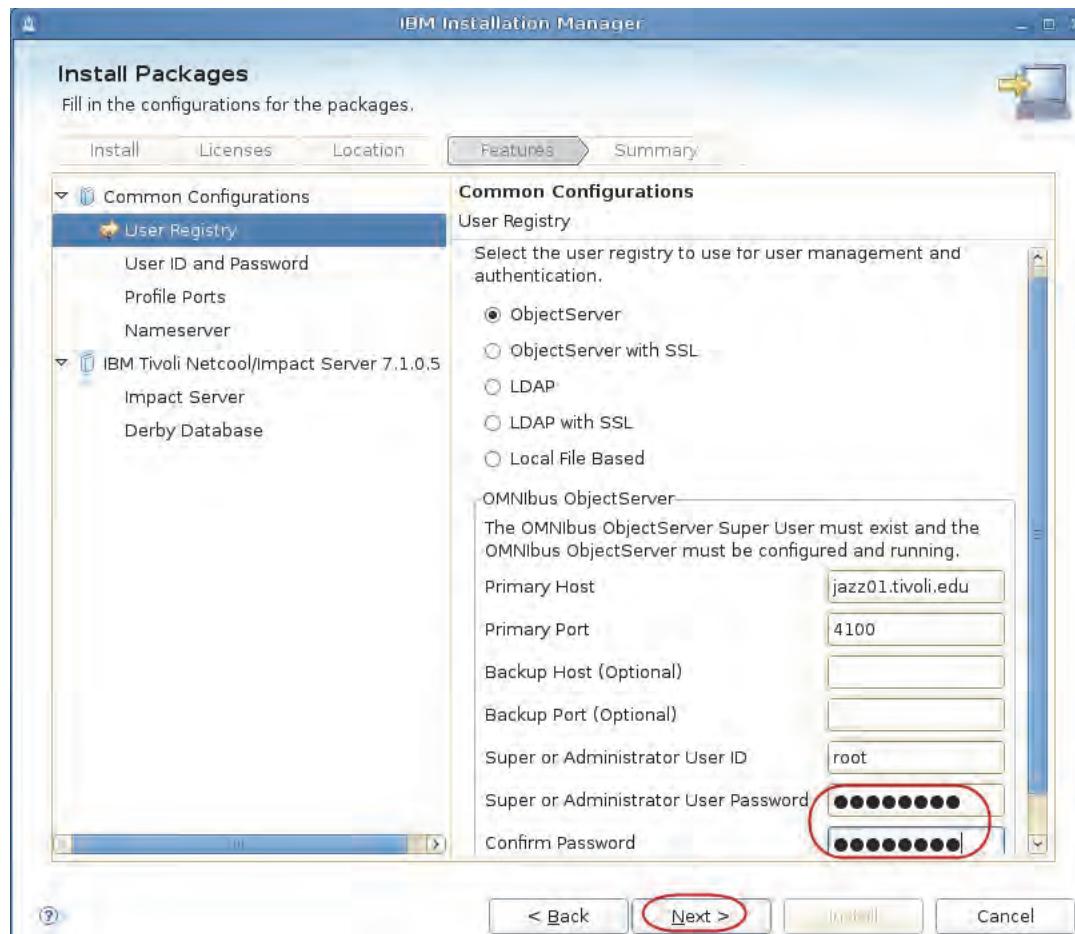
7. Click Next.



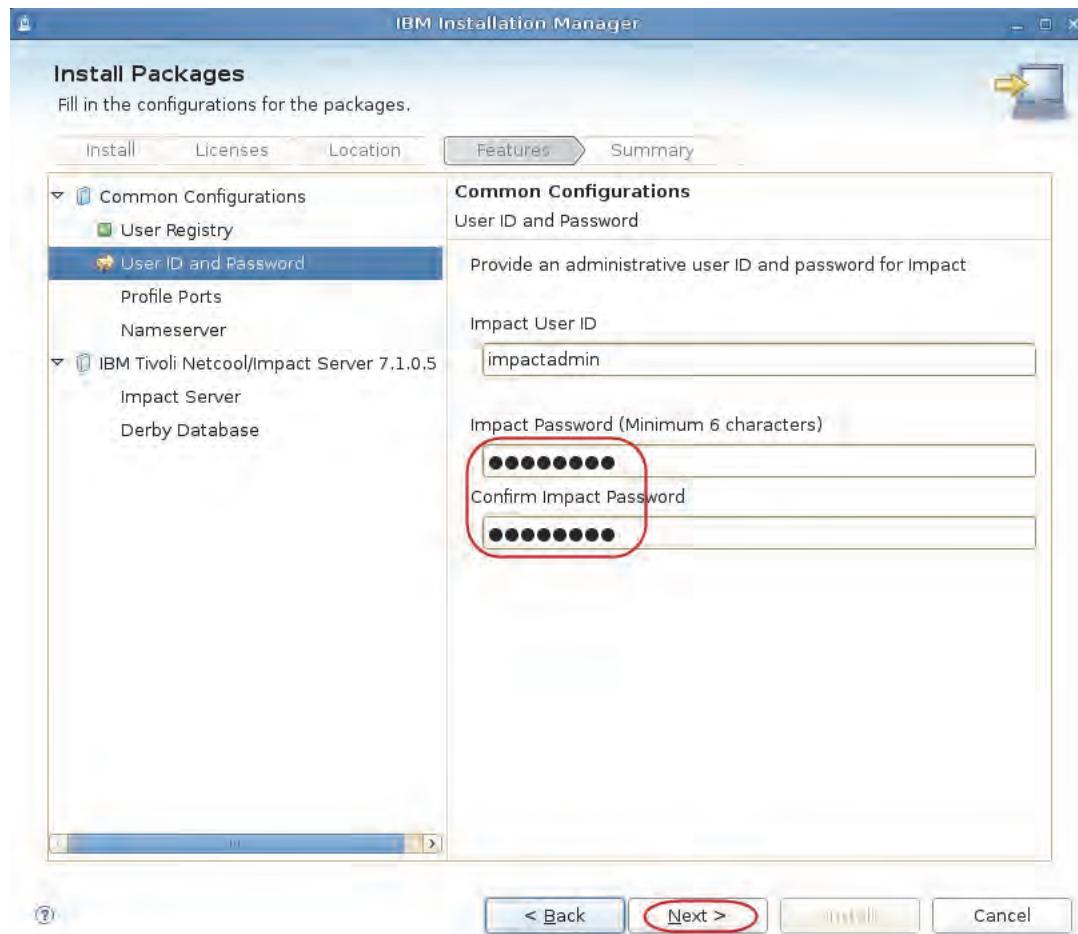
8. Accept the defaults and click **Next**.



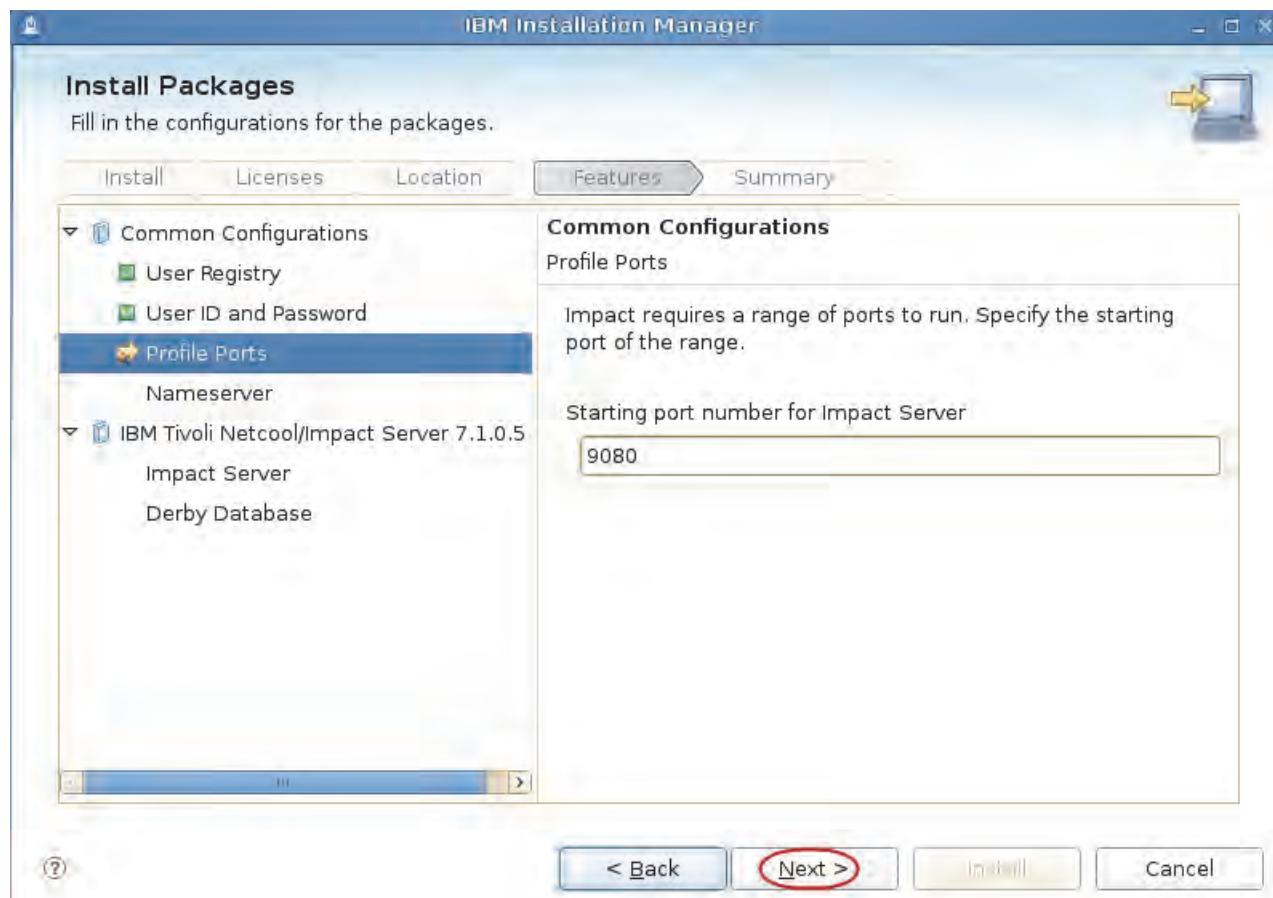
9. Type the password **object00**. Click **Next**.



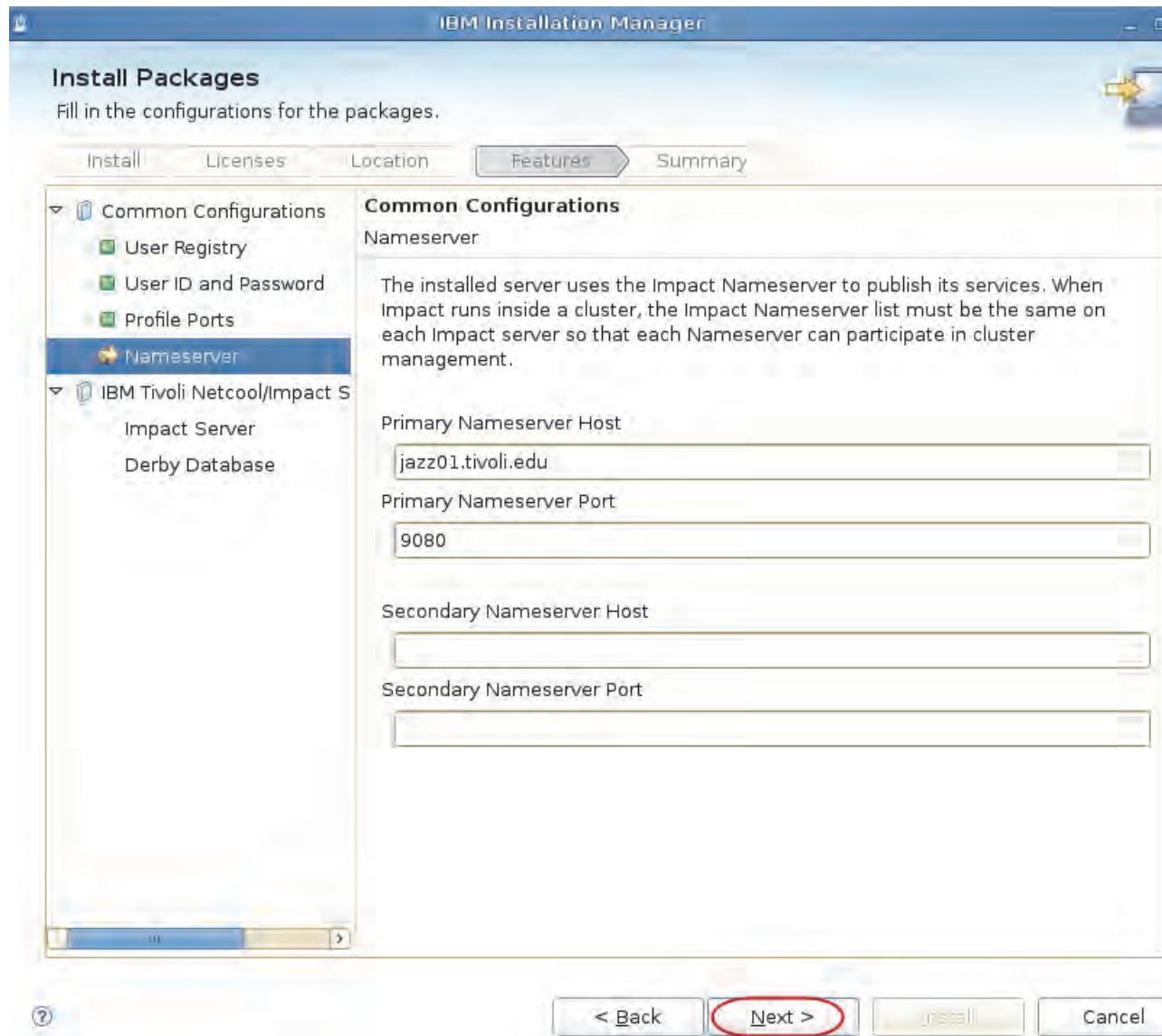
10. Type **object00** as the Impact password.



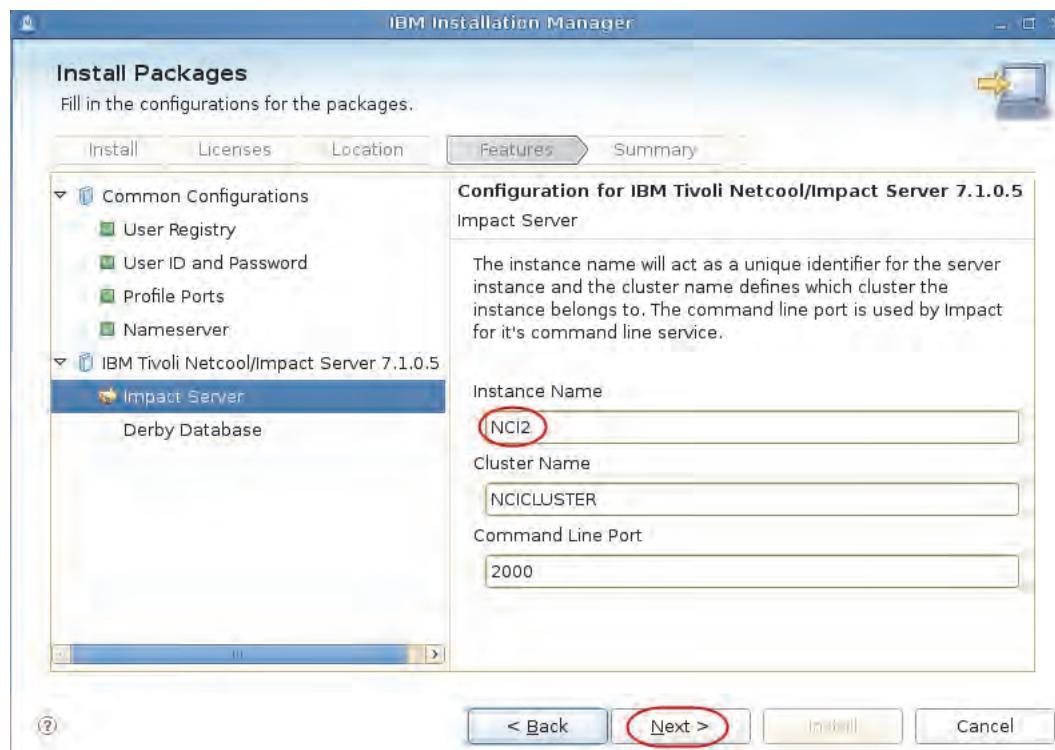
11. Accept the default port and click **Next**.



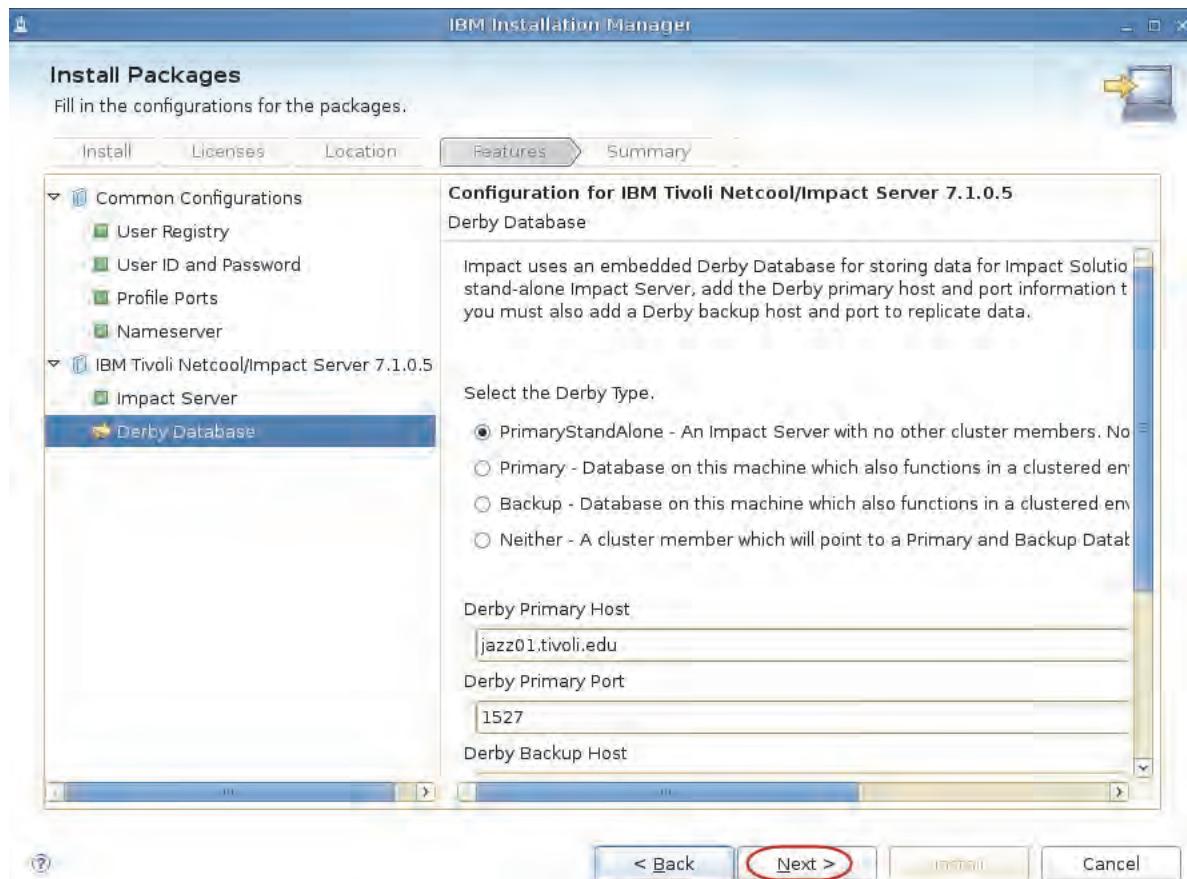
12. Accept the defaults. Click **Next**.



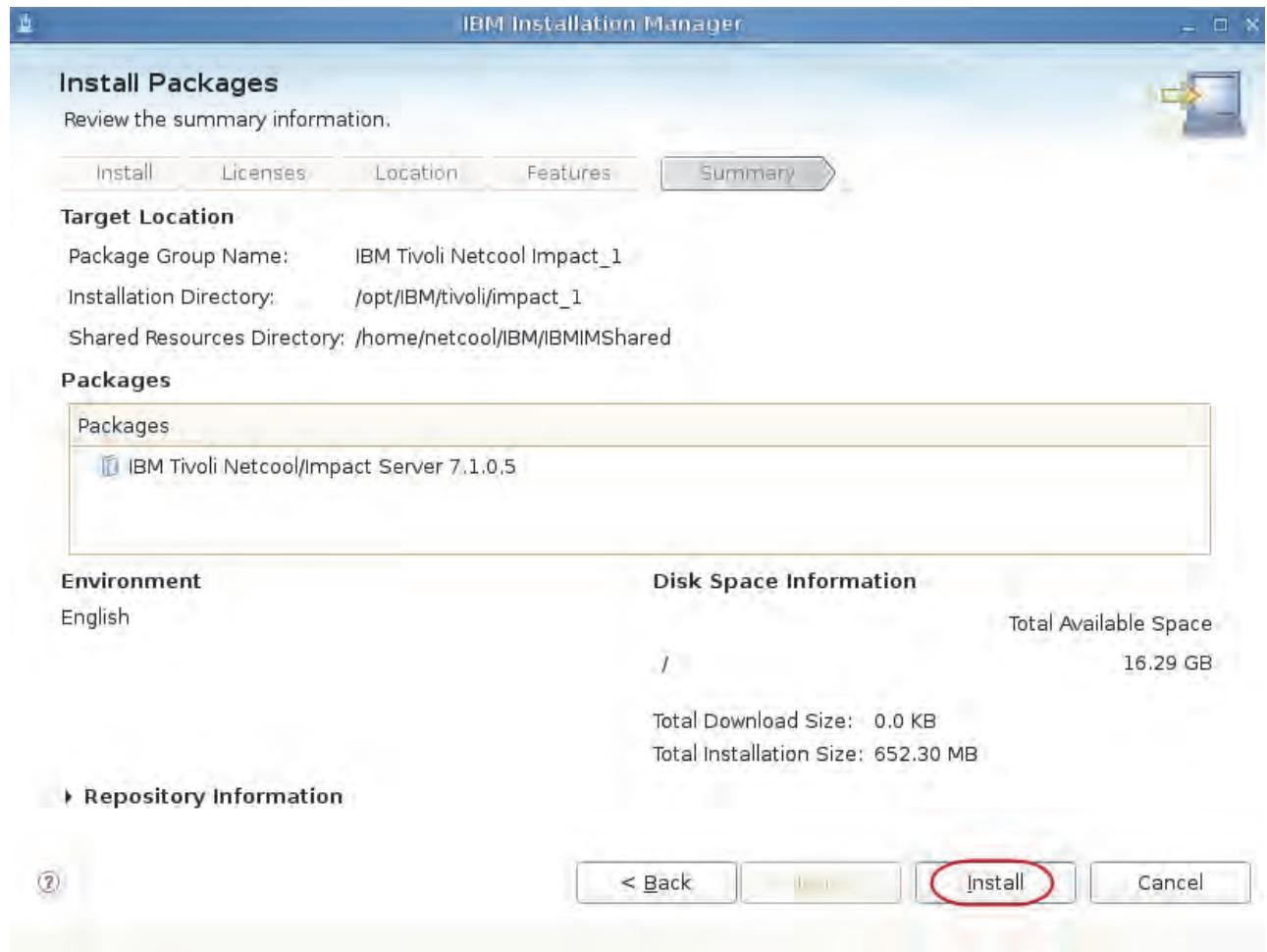
13. Type **NCI2** as the new Instance Name. Click **Next**.



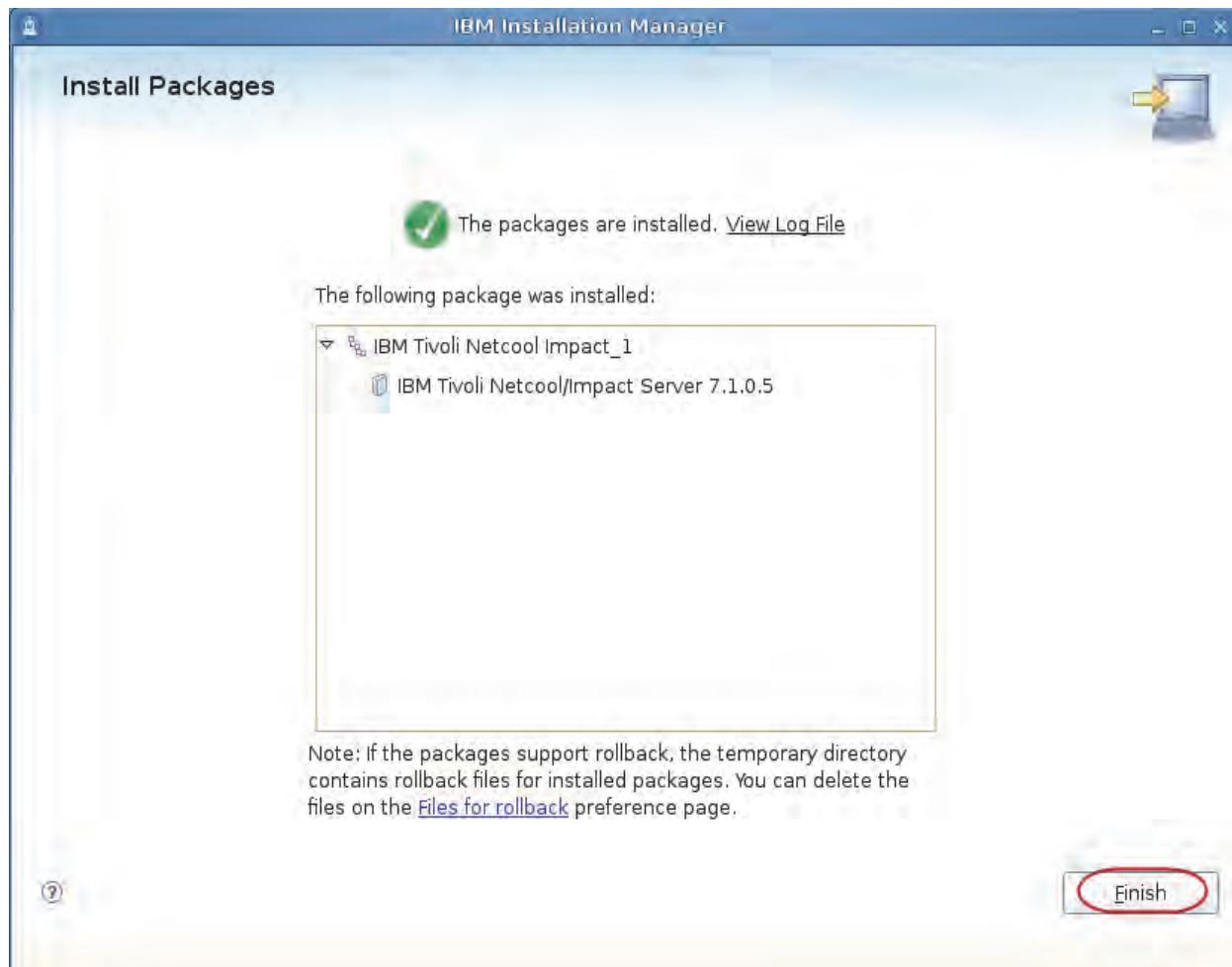
14. Accept the defaults and click **Next**.



15. Click Install.



16. Click **Finish** to complete the install.



17. Verify OMNIbus is running, type the following commands:

```
cd /opt/IBM/tivoli/netcool/omnibus/bin/  
./nco_objserv&
```

18. To verify the GUI server is running, type the following commands:

```
cd /opt/IBM/tivoli/impact/bin/  
./startGUIServer.sh
```

```
netcool@jazz01:/opt/IBM/tivoli/impact/bin> ./startGUIServer.sh  
Starting server ImpactUI.  
Server ImpactUI started with process ID 6457.  
netcool@jazz01:/opt/IBM/tivoli/impact/bin>
```

19. To verify the NCI2 instance has started, type the following commands:

```
cd /opt/IBM/tivoli/impact_1/bin/  
./startImpactServer.sh
```

```
netcool@jazz01:/opt/IBM/tivoli/impact_1/bin> ./startImpactServer.sh  
Server NCI2 is already running with process ID 31733.  
netcool@jazz01:/opt/IBM/tivoli/impact_1/bin>
```

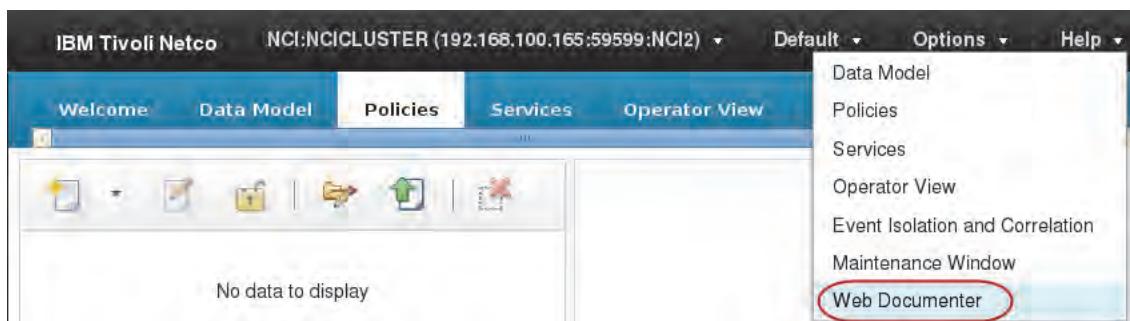
20. Click the IMPACT icon on the desktop.



21. Type the User name **impactadmin**, and Password **object00**.



22. Click the **Help** tab and select **Web Documenter** to open the **Configuration Documenter**.



23. Notice the **Primary Server** name. Close the Configuration Documenter.

The screenshot shows the Configuration Documenter interface for the NCICLUSTER cluster. The primary server, NCI2 (Current Instance), is highlighted with a red circle. Other sections visible include Server Status (NCI2) and Event Status (NCI2).

**Cluster Status for NCICLUSTER**

Primary Server	Host
NCI2 (Current Instance)	jazz01.tivoli.edu

**Server Status**

NCI2

Memory Status	Current Usage (in MB)	Max Limit (in MB)
Heap Memory Utilization	127	2400

**Event Status**

NCI2

Service Name	Number of Events in Queue	Event Source
EventProcessor	0	

**Data Sources**

Data Source Name	Data Source Type
EIC_alertsdb	ObjectServer

24. To verify the primary server using the command line, enter the following commands:

```
cd /opt/IBM/tivoli/impact_1/bin/  
.nci_get_primary http jazz01.tivoli.edu 9080 NCICLUSTER
```

```
Press [ENTER] for options  
http://jazz01.tivoli.edu:9080/nameserver/services-->  
Get Binding
```

```
Enter product: Enter service: *** Getting binding  
> getBinding(productbase, servicebase)  
    productbase: Impact  
    servicebase: NCICLUSTER  
< Result returned  
    binding: 192.168.100.165:59599 NCI2  
DONE
```

```
Press [ENTER] for options  
http://jazz01.tivoli.edu:9080/nameserver/services-->  
Quit
```

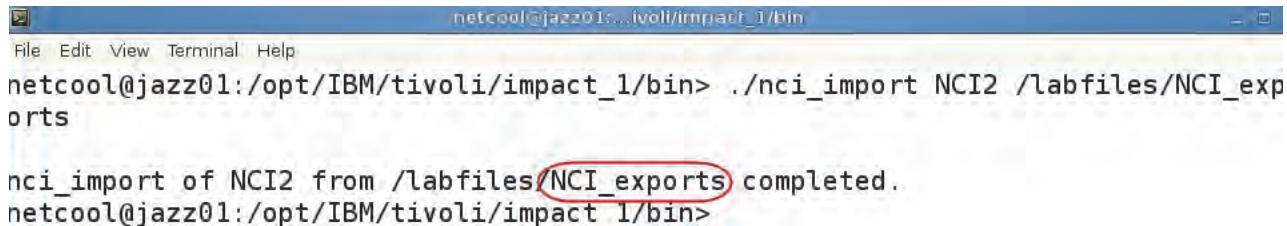
```
*** Exiting
```

```
netcool@jazz01:/opt/IBM/tivoli/impact_1/bin>
```

# Exercise 3 Importing Netcool/Impact Server

1. Use the import command to load the policies, data model, and services using the following command.

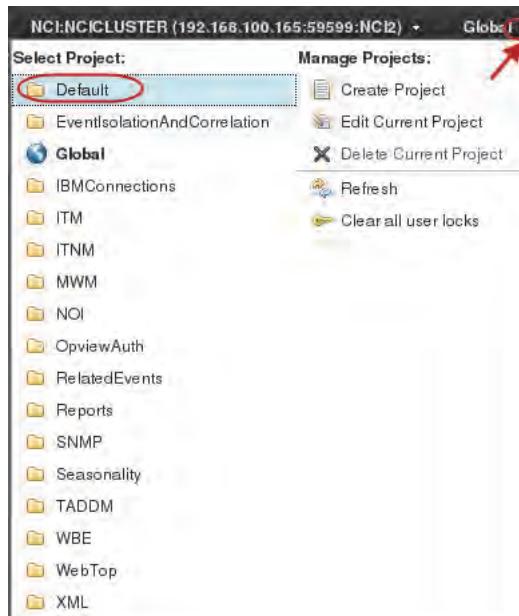
```
cd /opt/IBM/tivoli/impact_1/bin/  
.nci_import NCI2 /labfiles/NCI_exports  
This process will take a few minutes to complete.
```



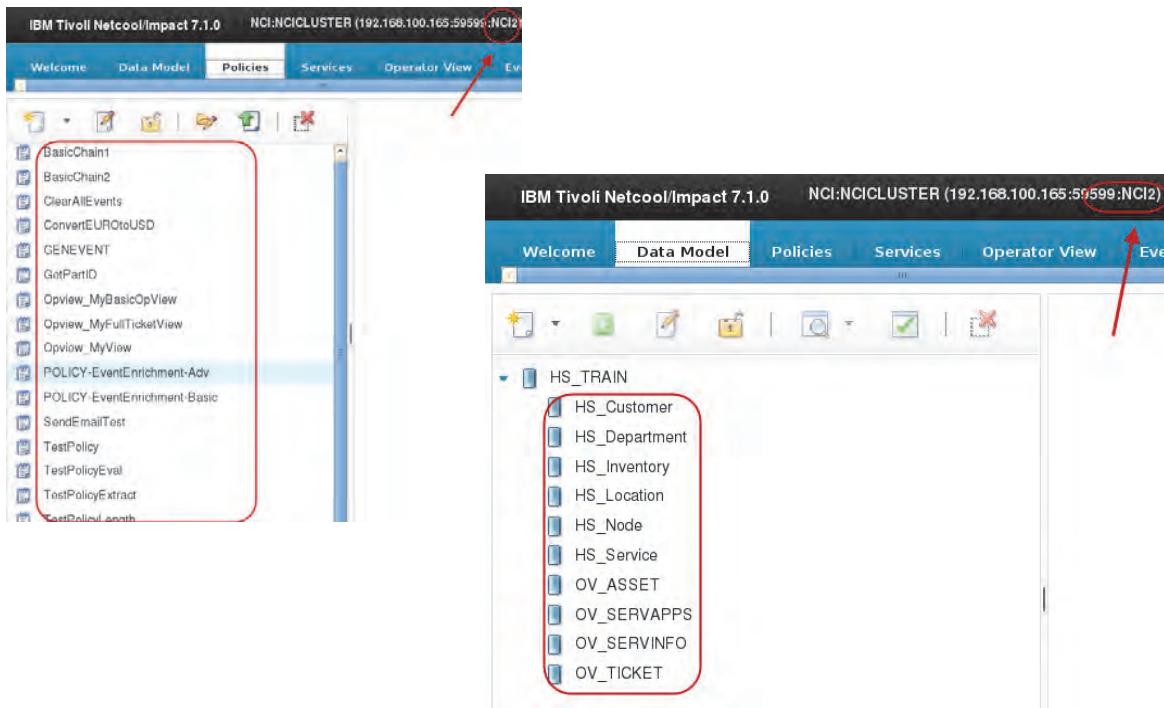
A terminal window titled "netcool@jazz01:/opt/IBM/tivoli/impact\_1/bin". The user runs the command ".nci\_import NCI2 /labfiles/NCI\_exports". The output shows the import process has completed successfully.

```
netcool@jazz01:/opt/IBM/tivoli/impact_1/bin> .nci_import NCI2 /labfiles/NCI_exports  
nci_import of NCI2 from /labfiles/NCI_exports completed.  
netcool@jazz01:/opt/IBM/tivoli/impact_1/bin>
```

2. Switch to the **Default** project, if the **Global** project is showing.



3. Verify that the Policies, Data Model, Services, and Operator Views have been imported properly from the backup.



4. Restart the NCI server instance using the following commands.

```
cd /opt/IBM/tivoli/impact/bin/  
. ./startImpactServer.sh
```

```
netcool@jazz01:/opt/IBM/tivoli/impact/bin> ./startImpactServer.sh  
Starting server NCI.  
Server NCI started with process ID 14210.  
netcool@jazz01:/opt/IBM/tivoli/impact/bin>
```

5. Click the **Help** tab and select **Web Documenter** to display the **Configuration Documenter**. Notice that **NCI** is now the secondary server.

**IBM Tivoli Netcool/Impact 7.1.0.5**      Refresh

**Configuration Documenter**

Server Name: NCI2 Version: 7.1.0.5 (201603221218) Generated on: Sat-Jun 25 02:54:57

Status | Data Sources | Data Types | Policies | S

### Cluster Status for NCICLUSTER

Primary Server	Host
NCI2 (Current Instance)	jazz01.tivoli.edu

Secondary Server	Host
NCI	jazz01.tivoli.edu

**Server Status**

NCI2		
Memory Status	Current Usage (in MB)	Max Limit (in MB)
Heap Memory Utilization	177	2400

NCI		
Memory Status	Current Usage (in MB)	Max Limit (in MB)
Heap Memory Utilization	100	1200

**Event Status**

NCI2		
Service Name	Number of Events in Queue	Event Source
ProcessSeasonalityAfterAction	0	
EventProcessor	0	ProcessSeasonalityAfterAction ProcessSeasonalityEvents
ProcessSeasonalityEvents	0	



IBM Training



© Copyright IBM Corporation 2017 All Rights Reserved.