Netcool/Impact
Version 6.1.1.3

Integrations Guide

SC27-4854-00

Netcool/Impact
Version 6.1.1.3

Integrations Guide

SC27-4854-00

Note
Before using this information and the product it supports, read the information in "Notices".

Edition notice

This edition applies to version 6.1.1.3 of IBM Tivoli Netcool/Impact and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Integrations Guide

The Netcool/Impact *Integrations Guide* contains instructions on integrating Netcool/Impact with other IBM® software and other third party software.

Intended audience

This publication is for users who are responsible for integrating Tivoli® Netcool/Impact with other IBM software and other vendor software.

Publications

This section lists publications in the Netcool/Impact library and related documents. The section also describes how to access Tivoli publications online and how to order Tivoli publications.

Netcool/Impact library

- *Quick Start Guide*, CF39PML

  Provides concise information about installing and running Netcool/Impact for the first time.
- *Administration Guide*, SC14755900

  Provides information about installing, running and monitoring the product.
- *User Interface Guide*, SC27485100

  Provides instructions for using the Graphical User Interface (GUI).
- *Policy Reference Guide*, SC14756100

  Contains complete description and reference information for the Impact Policy Language (IPL).
- *DSA Reference Guide*, SC27485200

  Provides information about data source adaptors (DSAs).
- *Operator View Guide*, SC27485300

  Provides information about creating operator views.
- *Solutions Guide*, SC14756000

  Provides end-to-end information about using features of Netcool/Impact.
- *Integrations Guide*, SC27485400

  Contains instructions for integrating Netcool/Impact with other IBM software and other vendor software.
- *Troubleshooting Guide*, GC27485500

  Provides information about troubleshooting the installation, customization, starting, and maintaining Netcool/Impact.

Accessing terminology online

The IBM Terminology Web site consolidates the terminology from IBM product libraries in one convenient location. You can access the Terminology Web site at the following Web address:

http://www.ibm.com/software/globalization/terminology

Accessing publications online

Publications are available from the following locations:

- The *Quick Start* DVD contains the Quick Start Guide. Refer to the readme file on the DVD for instructions on how to access the documentation.
- Tivoli Information Center web site at http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcoolimpact.doc6.1.1/welcome.html. IBM posts publications for all Tivoli products, as they become available and whenever they are updated to the Tivoli Information Center Web site.

  Note: If you print PDF documents on paper other than letter-sized paper, set the option in the **File → Print** window that allows Adobe Reader to print letter-sized pages on your local paper.
- Tivoli Documentation Central at http://www.ibm.com/tivoli/documentation. You can access publications of the previous and current versions of Netcool/Impact from Tivoli Documentation Central.
- The Netcool/Impact wiki contains additional short documents and additional information and is available at https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/Tivoli%20Netcool%20Impact.

Ordering publications

You can order many Tivoli publications online at http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss.

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, contact your software account representative to order Tivoli publications. To locate the telephone number of your local representative, perform the following steps:

1. Go to http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss.
2. Select your country from the list and click **Go**.
3. Click **About this site** in the main panel to see an information page that includes the telephone number of your local representative.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see Appendix A, “Accessibility,” on page 49.

Tivoli technical training

For Tivoli technical training information, refer to the following IBM Tivoli Education Web site at http://www.ibm.com/software/tivoli/education.

Support for problem solving

If you have a problem with your IBM software, you want to resolve it quickly. This section describes the following options for obtaining support for IBM software products:

- "Obtaining fixes"
- "Receiving weekly support updates"
- "Contacting IBM Software Support" on page viii

Obtaining fixes

A product fix might be available to resolve your problem. To determine which fixes are available for your Tivoli software product, follow these steps:

1. Go to the IBM Software Support Web site at http://www.ibm.com/software/support.
2. Navigate to the **Downloads** page.
3. Follow the instructions to locate the fix you want to download.
4. If there is no **Download** heading for your product, supply a search term, error code, or APAR number in the search field.

For more information about the types of fixes that are available, see the *IBM Software Support Handbook* at http://www14.software.ibm.com/webapp/set2/sas/f/handbook/home.html.

Receiving weekly support updates

To receive weekly e-mail notifications about fixes and other software support news, follow these steps:

1. Go to the IBM Software Support Web site at http://www.ibm.com/software/support.
2. Click the **My IBM** in the toobar. Click **My technical support**.
3. If you have already registered for **My technical support**, sign in and skip to the next step. If you have not registered, click **register now**. Complete the registration form using your e-mail address as your IBM ID and click **Submit**.
4. The **Edit profile** tab is displayed.
5. In the first list under **Products**, select **Software**. In the second list, select a product category (for example, **Systems and Asset Management**). In the third list, select a product sub-category (for example, **Application Performance & Availability** or **Systems Performance**). A list of applicable products is displayed.
6. Select the products for which you want to receive updates.
7. Click **Add products**.
8. After selecting all products that are of interest to you, click **Subscribe to email** on the **Edit profile** tab.
9. In the **Documents** list, select **Software**.
10. Select **Please send these documents by weekly email**.
11. Update your e-mail address as needed.
12. Select the types of documents you want to receive.
13. Click **Update**.

If you experience problems with the **My technical support** feature, you can obtain help in one of the following ways:

Online

Send an e-mail message to erchelp@u.ibm.com, describing your problem.

By phone

Call 1-800-IBM-4You (1-800-426-4409).

World Wide Registration Help desk

For word wide support information check the details in the following link: https://www.ibm.com/account/profile/us?page=reghelpdesk

Contacting IBM Software Support

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. The type of software maintenance contract that you need depends on the type of product you have:

- For IBM distributed software products (including, but not limited to, Tivoli, Lotus®, and Rational® products, and DB2® and WebSphere® products that run on Windows or UNIX operating systems), enroll in Passport Advantage® in one of the following ways:

  Online

  Go to the Passport Advantage Web site at http://www-306.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm .

  By phone

  For the phone number to call in your country, go to the IBM Worldwide IBM Registration Helpdesk Web site at https://www.ibm.com/account/profile/us?page=reghelpdesk.
- For customers with Subscription and Support (S & S) contracts, go to the Software Service Request Web site at https://techsupport.services.ibm.com/ssr/login.
- For customers with IBMLink, CATIA, Linux, OS/390®, iSeries, pSeries, zSeries, and other support agreements, go to the IBM Support Line Web site at http://www.ibm.com/services/us/index.wss/so/its/a1000030/dt006.
- For IBM eServer™ software products (including, but not limited to, DB2 and WebSphere products that run in zSeries, pSeries, and iSeries environments), you can purchase a software maintenance agreement by working directly with an IBM sales representative or an IBM Business Partner. For more information about support for eServer software products, go to the IBM Technical Support Advantage Web site at http://www.ibm.com/servers/eserver/techsupport.html.

If you are not sure what type of software maintenance contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States. From other countries, go to the contacts page of the *IBM Software Support Handbook* on the Web at http://www14.software.ibm.com/webapp/set2/sas/f/handbook/home.html and click the name of your geographic region for phone numbers of people who provide support for your location.

To contact IBM Software support, follow these steps:

1. “Determining the business impact” on page ix
2. “Describing problems and gathering information” on page ix
3. “Submitting problems” on page ix

Determining the business impact

When you report a problem to IBM, you are asked to supply a severity level. Use the following criteria to understand and assess the business impact of the problem that you are reporting:

Severity 1
: The problem has a *critical* business impact. You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.

Severity 2
: The problem has a *significant* business impact. The program is usable, but it is severely limited.

Severity 3
: The problem has *some* business impact. The program is usable, but less significant features (not critical to operations) are unavailable.

Severity 4
: The problem has *minimal* business impact. The problem causes little impact on operations, or a reasonable circumvention to the problem was implemented.

Describing problems and gathering information

When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:

- Which software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can you re-create the problem? If so, what steps were performed to re-create the problem?
- Did you make any changes to the system? For example, did you make changes to the hardware, operating system, networking software, and so on.
- Are you currently using a workaround for the problem? If so, be prepared to explain the workaround when you report the problem.

Submitting problems

You can submit your problem to IBM Software Support in one of two ways:

Online
: Click **Submit and track problems** on the IBM Software Support site at http://www.ibm.com/software/support/probsub.html. Type your information into the appropriate problem submission form.

By phone
: For the phone number to call in your country, go to the contacts page of the *IBM Software Support Handbook* at http://www14.software.ibm.com/webapp/set2/sas/f/handbook/home.html and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the Software Support Web site daily, so that other users who experience the same problem can benefit from the same resolution.

Conventions used in this publication

This publication uses several conventions for special terms and actions, operating system-dependent commands and paths, and margin graphics.

Typeface conventions

This publication uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as **Tip:**, and **Operating system considerations:**)
- Keywords and parameters in text

Italic

- Citations (examples: titles of publications, diskettes, and CDs
- Words defined in text (example: a nonswitched line is called a *point-to-point line*)
- Emphasis of words and letters (words as words example: "Use the word *that* to introduce a restrictive clause."; letters as letters example: "The LUN address must start with the letter *L*.")
- New terms in text (except in a definition list): a *view* is a frame in a workspace that contains data.
- Variables and values you must provide: ... where *myname* represents....

`Monospace`

- Examples and code examples
- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This publication uses the UNIX convention for specifying environment variables and for directory notation.

When using the Windows command line, replace **$***variable* with **%***variable***%** for environment variables and replace each forward slash (**/**) with a backslash (\) in directory paths. The names of environment variables are not always the same in the Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX environments.

Note: If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Chapter 1. Integrating with Netcool/Impact

How to integrate Netcool/Impact with other IBM® software and other third party software.

Chapter 2. Integrating Netcool/Impact with Tivoli Enterprise Portal

Thanks to the integration of Netcool/Impact with Tivoli Enterprise Portal you can open the Netcool/Impact Operator View from Tivoli Enterprise Portal.

Overview

You use the Operator View in the Tivoli Enterprise Portal to see data from heterogeneous applications in a web-based view and to run Impact policies on this additional data. The accessibility of the Operator View from Tivoli Enterprise Portal means that you can jump quickly from an object in the Tivoli Enterprise Portal to associated data stored in other applications. For example, you can create an Operator View that shows the history of trouble tickets in an application acquired from another vendor, combined with the change management history from a change management application. You can include a button on the Operator View to open a new trouble ticket through an IPL policy.

The solution uses the ability of Tivoli Enterprise Portal to create right-click Launch definitions which run external processes. You create a Launch definition that runs a wrapper script. The Launch passes the wrapper script space-separated attributes from the Tivoli Enterprise Portal object that is tied to the Launch. The wrapper script then starts the browser with the full URL to the desired Operator View. For more information about creating wrapper scripts and about creating Launch definitions, see "Copying and editing OVWrapper script templates" on page 4, and "Creating Launch definitions" on page 5.

Supported operating systems

The integration with the Tivoli Enterprise Portal is compatible with the operating systems supported by Tivoli Netcool/Impact 6.1.1 and IBM Tivoli Enterprise Portal version 6.1 and 6.2.

The most up-to-date information about supported hardware, software, browsers, and operating systems is provided by the IBM® Software Product Compatibility Reports at: http://pic.dhe.ibm.com/infocenter/prodguid/v1r0/clarity/index.html

For more information about running the Software Product Compatibility Reports, see the *Overview and Planning* section of the Tivoli Netcool/Impact wiki at: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home/wiki/Tivoli%20Netcool%20Impact/page/Overview%20and%20Planning?lang=en

Setting up the integration

Follow this procedure to set up the integration with Tivoli Enterprise Portal.

Procedure

1. Create one or more Operator Views.

   For information about creating Operator Views, see Operator View Guide.

   There are also some examples of Operator Views in "Examples of operator views" on page 6.

2. Copy and edit the OVWrapper shell script templates.

   For more information about the script, see "Copying and editing OVWrapper script templates."
3. Create Tivoli Enterprise Portal Launch definitions.

   For more information about creating launch definitions, see "Creating Launch definitions" on page 5.

Copying and editing OVWrapper script templates

You use the OVWrapper script templates that you can find in the $IMPACT_HOME/impact/integrations/tep directory to set up the integration.

Procedure

1. Copy one the following templates, depending on your operating system, to the host or hosts that are running the Tivoli Enterprise Portal client:
   - OVWrapper.bat (Windows systems)
   - OVWrapper.sh (UNIX systems)
2. Make a copy of the template for each Operator View that you are going to view.

   You need one script for each Operator View being viewed.
3. Edit those copies to work with your Operator Views.

   Change the list of attribute names, *var1*, *var2*, and so on, and the line that starts the browser. See the attached templates for more details.

Example

The OVWrapper.bat script template for Windows platforms:

```
@echo off
rem YOUR url here
set OV_URL=http://9.15.165.121:16310/opview/displays/
WSCCLUSTERExampleOV_for_wrapper.html
rem path to YOUR browser here
set OV_BROWSER=c:\progra~1\mozill~1\firefox.exe
rem set BROWSER=c:\progra~1\Intern~1\IEXPLORE.EXE
rem YOUR attribute names here
set var1=FirstName
set var2=LastName
set var3=Age
rem example line to run browser and pass attributes
start %OV_BROWSER%
%OV_URL%?%var1%=%1^&%var2%=%2^&%var3%=%3
rem notes on example line:
rem     ^ before & is necessary to escape &
rem     ? seems ok without being escaped
```

The OVWrapper.sh script template for UNIX platforms:

```
#!/bin/sh
# YOUR url here
set OV_URL=http://9.15.165.121:16310/opview/displays/
WSCCLUSTERExampleOV_for_wrapper.html
# path to YOUR browser here
OV_BROWSER=/opt/firefox/firefox
# YOUR attribute names here
var1=FirstName
var2=LastName
var3=Age
# example line to run browser and pass attributes
```

```
$OV_BROWSER $OV_URL?$var1=$1"&"$var2=$2"&"$var3=$3
# notes on example line:
# quote around & are necessary to escape &
# ? seems ok without being escaped
```

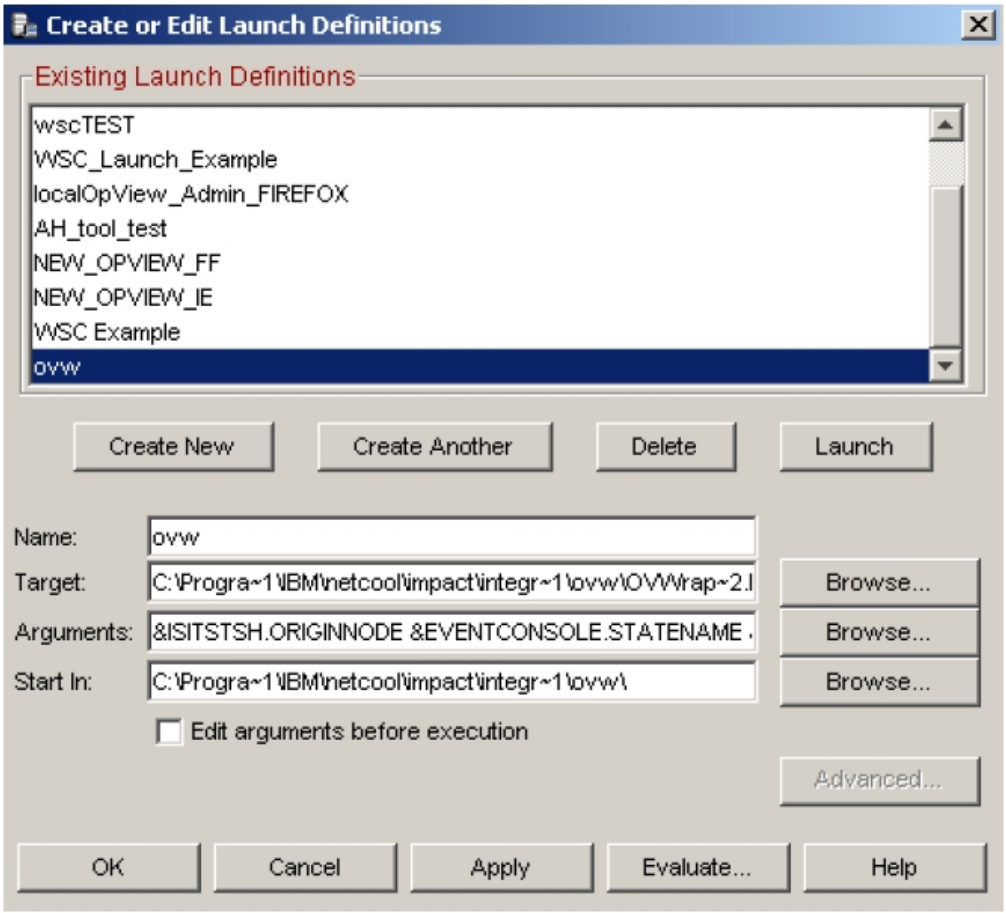
Creating Launch definitions

You can launch operator views from within Tivoli Enterprise Portalby creating a launch definition. Refer to the *IBM Tivoli Monitoring 6.2 User's Guide* for complete information and details about how to add launch definitions.

When starting an operator view fromTivoli Enterprise Portal, you can also choose to forward event data to the operator view using substitute arguments in the launch definition.

The following screen capture is an example of a launch definition for a Netcool/Impact operator view.

Important: You must use the 8.3 format for file names in Windows.



You must complete the following fields for your operator view in the Create or Edit Launch Definitions window:

Name

The name of the Launch definition, in this case **ovw**.

Target

The process to run. The full path in this example is `c:\ProgramFiles\IBM\netcool\impact\integrations\ovw\OVWrap-2.bat`. For this Launch to work (Windows 2000, ITM 6.2 FP 4) the long file names must be shortened to the DOS 8.3 name format. The 8.3 name format convention still applies in ITM 6.2.2 and Windows 2003.

Arguments

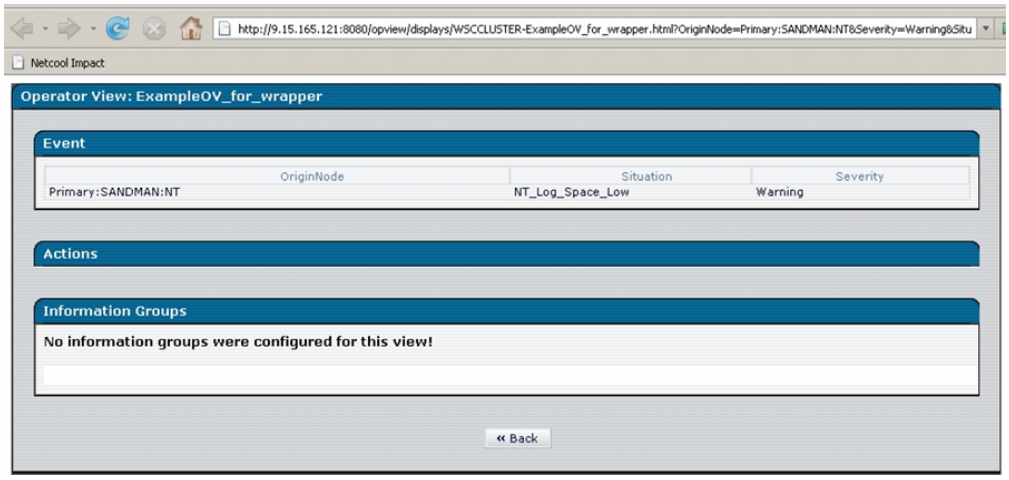
Substituted values from the object the Launch is run from. This example uses **`&ISITSTSH.ORIGINNODE &EVENTCONSOLE.STATENAME &ISITSTSH.SITNAME`** .

Start in

Specifies which folder to start the Launch from. Use the same folder as the batch file. The file path must use the 8.3 file name convention in Windows.

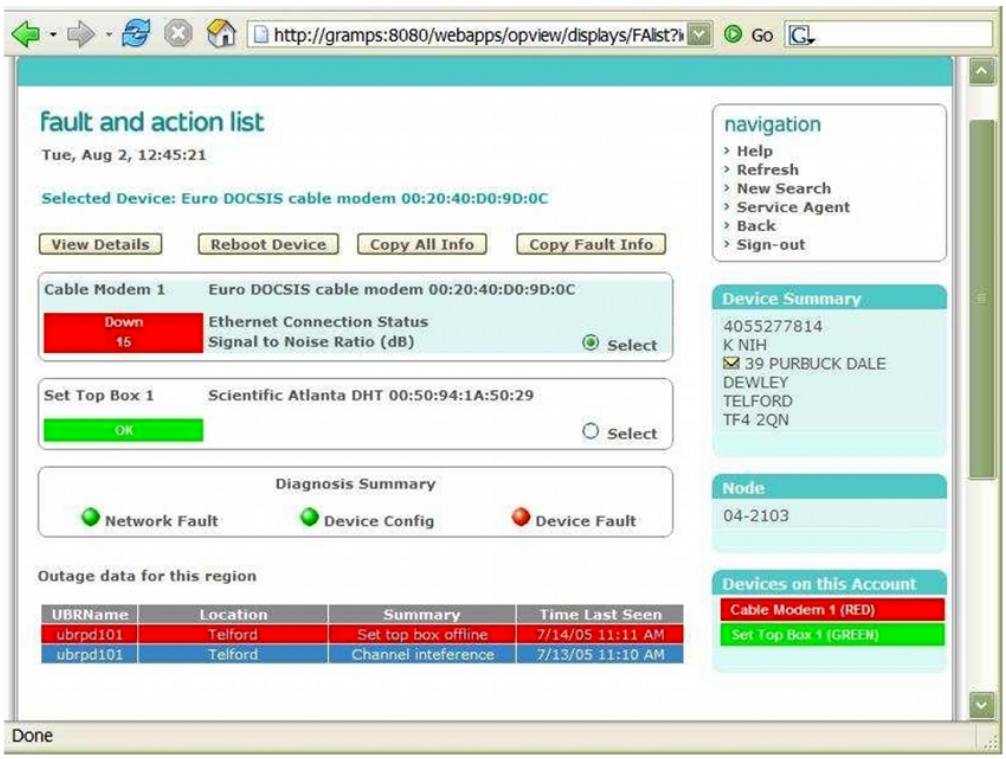
Example

An example of a simple Operator View that just shows the data that is passed to it.

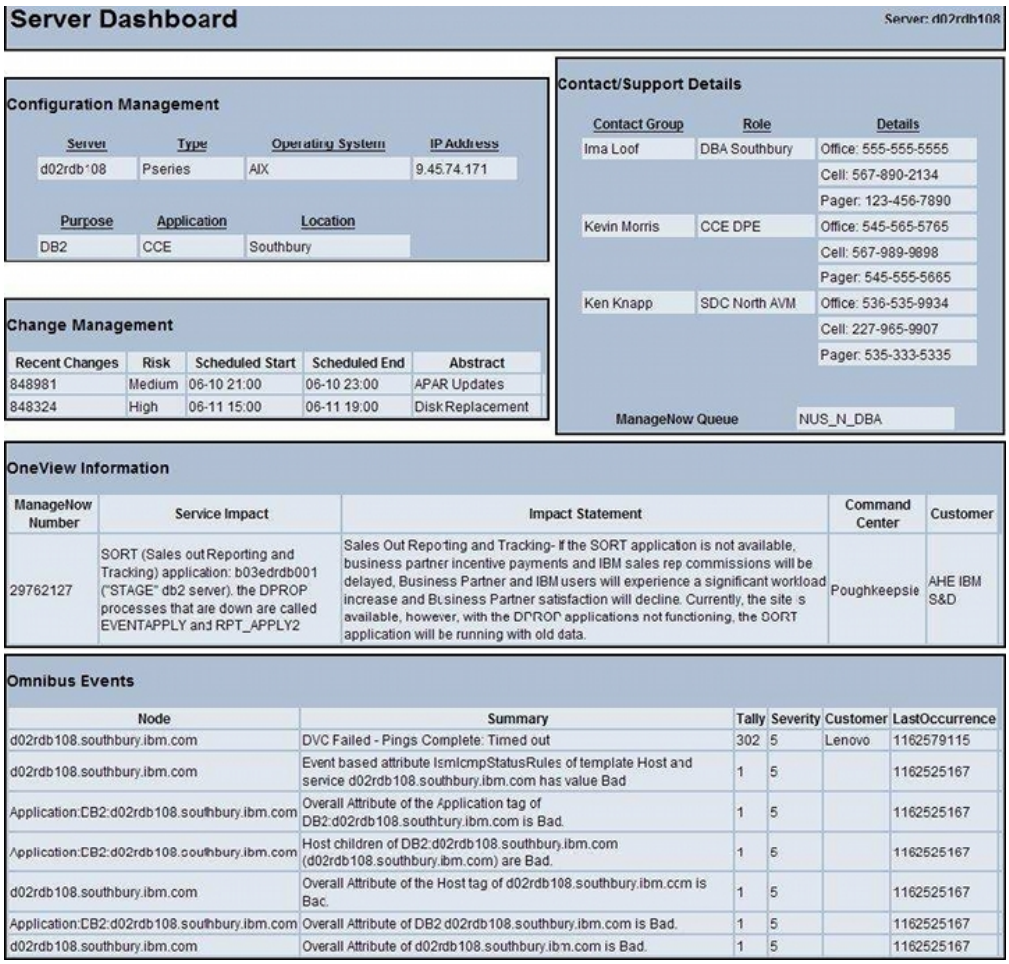


Examples of operator views

Here is an example of an operator views created in the field. Data from various applications is gathered and displayed in a single pane. The red and green color is used to analyze the obtained data. The buttons run other policies and some buttons open another page to display the results.



In another example of an operator view, various data is obtained through policies and then displayed in a web page.



Server Dashboard

Server: d02rdb108

Configuration Management

Server	Type	Operating System	IP Address
d02rdb108	Pseries	AIX	9.45.74.171

Purpose	Application	Location
DB2	CCE	Southbury

Contact/Support Details

Contact Group	Role	Details
Ima Loof	DBA Southbury	Office: 555-555-5555
		Cell: 567-890-2134
		Pager: 123-456-7890
Kevin Morris	CCE DPE	Office: 545-565-5765
		Cell: 567-989-9898
		Pager: 545-555-5665
Ken Knapp	SDC North AVM	Office: 536-535-9934
		Cell: 227-965-9907
		Pager: 535-333-5335

ManageNow Queue: NUS_N_DBA

Change Management

Recent Changes	Risk	Scheduled Start	Scheduled End	Abstract
848981	Medium	06-10 21:00	06-10 23:00	APAR Updates
848324	High	06-11 15:00	06-11 19:00	Disk Replacement

OneView Information

ManageNow Number	Service Impact	Impact Statement	Command Center	Customer
29762127	SORT (Sales out Reporting and Tracking) application: b03edrdb001 ("STAGE" db2 server). the DPROP processes that are down are called EVENTAPPLY and RPT_APPLY2	Sales Out Reporting and Tracking- If the SORT application is not available, business partner incentive payments and IBM sales rep commissions will be delayed, Business Partner and IBM users will experience a significant workload increase and Business Partner satisfaction will decline. Currently, the site is available, however, with the DPROP applications not functioning, the SORT application will be running with old data.	Poughkeepsie	AHE IBM S&D

Omnibus Events

Node	Summary	Tally	Severity	Customer	LastOccurrence
d02rdb108.southbury.ibm.com	DVC Failed - Pings Complete: Timed out	302	5	Lenovo	1162579115
d02rdb108.southbury.ibm.com	Event based attribute IsmIcmpStatusRules of template Host and service d02rdb108.southbury.ibm.com has value Bad	1	5		1162525167
Application:DB2:d02rdb108.southbury.ibm.com	Overall Attribute of the Application tag of DB2:d02rdb108.southbury.ibm.com is Bad.	1	5		1162525167
Application:DB2:d02rdb108.southbury.ibm.com	Host children of DB2:d02rdb108.southbury.ibm.com (d02rdb108.southbury.ibm.com) are Bad.	1	5		1162525167
d02rdb108.southbury.ibm.com	Overall Attribute of the Host tag of d02rdb108.southbury.ibm.com is Bad.	1	5		1162525167
Application:DB2:d02rdb108.southbury.ibm.com	Overall Attribute of DB2 d02rdb108.southbury.ibm.com is Bad.	1	5		1162525167
d02rdb108.southbury.ibm.com	Overall Attribute of d02rdb108.southbury.ibm.com is Bad.	1	5		1162525167

Chapter 3. Integrating Netcool/Impact with IBM Tivoli Monitoring

You use the integration with IBM Tivoli Monitoring to send messages from Netcool/Impact into the Tivoli Monitoring Universal Message Console.

Messages are sent from Tivoli Netcool/Impact into the Tivoli Monitoring Universal Message Console through Web Services.

Configuring Netcool/Impact to send messages to Tivoli Monitoring Universal Message Console

Use this procedure to configure Netcool/Impact to send messages to Tivoli Monitoring 6.1 and higher, Universal Message Console.

Procedure

1. Download the **ITMLibraryFunctions.zip** file from the Netcool/Impact devWorks wiki in the *Scenarios and Examples* page at the following URL: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/Tivoli%20Netcool%20Impact/page/Scenarios%20and%20examples
2. Extract the **ITMLibraryFunctions.zip** compressed file. It includes the following two policies:
   - **ITMLibraryFunctions.ipl**, which includes all function calls to Tivoli Monitoring by using the GetHTTP function.
   - **ITMFunctionsCaller.ipl**, which includes examples of how to call the functions.
3. Import the two policies to the Impact Server. For more information, see *Netcool/Impact User Interface Guide*, *Working with policies, Uploading policies*.
4. Update the **ITMFunctionsCaller** policy with Tivoli Monitoring specific information such as host name, port, user name, password, attribute, and object that is based on the function to be called.

   Remember: The example function is using default values.
5. Run the policy. The functions return an XML formatted string.

Obtaining data from Tivoli Monitoring 6.3 using the UI data provider

To obtain data from Tivoli Monitoring 6.3, use the UI data provider DSA to access the Tivoli Monitoring data provider.

Procedure

For more information, see the following link and information center reference:

- Netcool/Impact devWorks wiki, the *Scenarios and Examples* page at the following URL: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/Tivoli%20Netcool%20Impact/page/Scenarios%20and%20examples

- *Netcool/Impact Solutions Guide, Visualizing data from the UI data provider in the console, Visualizing a data mashup from two IBM Tivoli Monitoring sources.* This example shows how to visualize the data from Tivoli Monitoring sources in dashboard. You can use the same method to retrieve data for event management purposes. The only difference is that you omit the step that describes how to create the user output parameters for the policies.

Chapter 4. Integrating Netcool/Impact with Tivoli Enterprise Console

In the integration between Tivoli Enterprise Console and Netcool/Impact, Tivoli Enterprise Console can be used as an event source for Netcool/Impact much in the same way Netcool®/OMNIbusis.

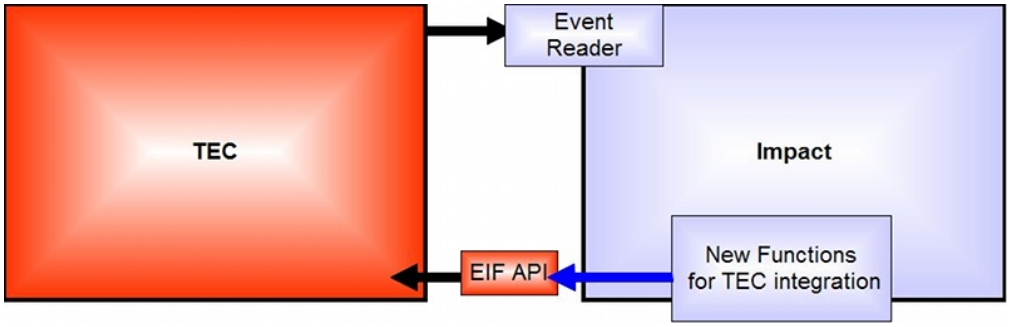
Overview of the integration

Tivoli Enterprise Console can be used as an event source for Netcool/Impact much in the same way Netcool/OMNIbus is. You can use this integration to hook Netcool/Impact into Tivoli Enterprise Console and process events in the same way that you process Netcool/OMNIbusevents.

You use the integration to complete the following tasks:

- Capture new or updated Tivoli Enterprise Console events for processing using an Event Reader
- Perform event enrichment on events that are caught with the Event Reader
- Insert new events into Tivoli Enterprise Console

Netcool/Impact pulls in new or updated events from the Tivoli Enterprise Console database using an Event Reader. Netcool/Impact functions are used to either enrich existing Tivoli Enterprise Console events or send new events to Tivoli Enterprise Console using Tivoli Enterprise Console EIF API.



Tivoli Enterprise Console was the Tivoli event management platform before Netcool/OMNIbus. If you are new to Tivoli Enterprise Console, you must familiarize yourself with the following concepts:

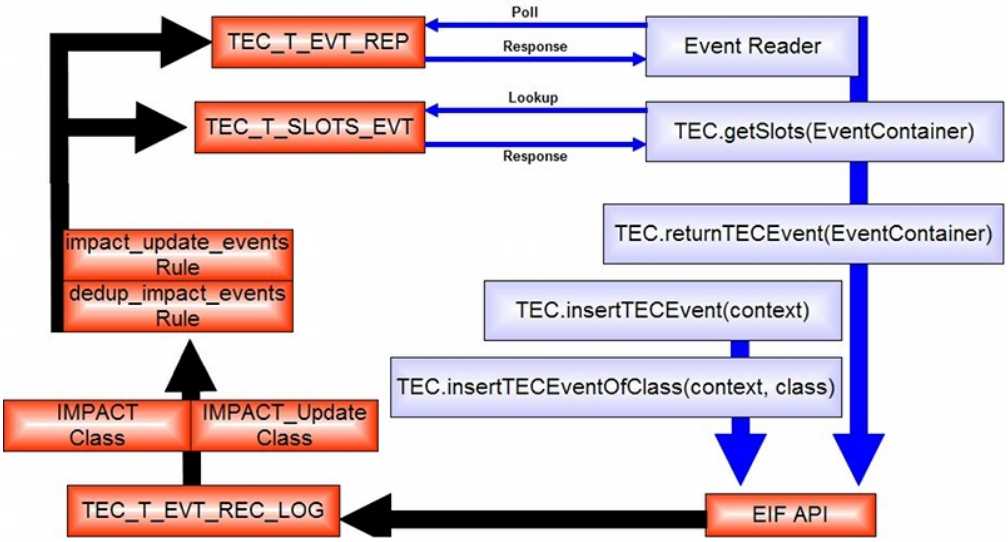
- Database Tables:
  - **TEC_T_EVT_REP** table
  - **TEC_T_SLOTS_EVT** table

  For more information about Tivoli Enterprise Console database tables, see "Tivoli Enterprise Console database tables" on page 12.
- TEC Classes: Every event in Tivoli Enterprise Console belongs to a Class. The Class defines how the raw **attribute=value** pairs in an incoming event are handled. For more information about classes, see "Tivoli Enterprise Console Classes" on page 14.
- TEC Rules: Tivoli Enterprise Console rules are run on incoming events that are based on class. For more information about rules, see "Tivoli Enterprise Console rules" on page 15.

- EIF API: The EIF API is used to move Tivoli Enterprise Console events around. The EIF API is used in an Netcool/OMNIbus Tivoli Enterprise Console probe. For more information about EIF API, see "EIF API to Tivoli Enterprise Console" on page 14.

Integration architecture

Netcool/Impact pulls in new or updated events from the Tivoli Enterprise Console database using an Event Reader. New Netcool/Impactfunctions are used to either enrich existing Tivoli Enterprise Console events or send new events to Tivoli Enterprise Console through the Tivoli Enterprise Console EIF API.



The Event Reader is used to catch new or updated rows in **TEC_T_EVT_REP**.

Tivoli Enterprise Console functions. For more information about Tivoli Enterprise Console functions, see "Tivoli Enterprise Console functions" on page 16.

EIF API. The various Tivoli Enterprise Console functions in Netcool/Impact use the EIF API to get their new or updated events over to the Tivoli Enterprise Console. Events sent over to the Tivoli Enterprise Console are first dumped into the table **TEC_T_EVT_REC_LOG**. For more information about the EIF API, see "EIF API to Tivoli Enterprise Console" on page 14.

The Tivoli Enterprise Console Classes. For more information about Tivoli Enterprise Console classes, see "Tivoli Enterprise Console Classes" on page 14.

The Tivoli Enterprise Console Rules. For more information about Tivoli Enterprise Console rules, see "Tivoli Enterprise Console rules" on page 15.

Tivoli Enterprise Console database tables

The integration uses the following Tivoli Enterprise Console database tables:

- **TEC_T_EVT_REP**
- **TEC_T_SLOTS_EVT**

TEC_T_EVT_REP database table

The **TEC_T_EVT_REP** is the core table where events are held and the table that the Tivoli Enterprise Console GUI console displays data from. This table is similar to the Netcool/OMNIbus `alert.status`, with the following differences:

- The **TEC_T_EVT_REP** table is not extensible. You can define extra **`attribute=value`** pairs called slots to hold data, but these slots do not have their own fields and are stored in a separate table.
- The **TEC_T_EVT_REP** table has three key fields:
  - **SERVER_HNDL**
  - **EVENT_HDNL**
  - **DATE_RECEPTION**

Refer to the Tivoli Enterprise Console documentation for a full list of fields in **TEC_T_EVT_REP**.

TEC_T_EVT_REP and Impact

TEC_T_EVT_REP is the Tivoli Enterprise Console table that is polled by the Netcool/Impact Event Reader to catch new or updated events. The timestamp field that is used when working with updated events is **LAST_MODIFIED_TIME**. The **Key** field can be anything. Events are not returned to the Tivoli Enterprise Console using the Netcool/Impact Event Reader but instead are sent through the EIF API.

The field **CREDIBILITY**, is normally 0 and not used by customers. It can be used as a Netcool/Impact field to signify that a Tivoli Enterprise Console row has been processed by Netcool/Impact. **CREDIBILITY** is set to 5 in events that are updated by Netcool/Impact by the default Tivoli Enterprise Console rules shipping with this integration. The **STATUS** field has the values 0 (open), 20 (acknowledged), and 30 (closed). Use as required in your Event Reader filters.

Table TEC_T_SLOTS_EVT database table

The core event table, **TEC_T_EVT_REP** is not extensible; instead, new slots are defined and stored in **TEC_T_SLOTS_EVT**.

The name of a slot is stored in the field **slot_name**. The value of a slot is stored in either **SHORT_SLOT_VALUE** or **LONG_SLOT_VALUE**, depending on the length of the value. Tivoli Enterprise Console adds the data to the right value field. Rows in **TEC_T_SLOTS_EVT**are linked back to rows in **TEC_T_EVT_REP** by the key fields **SERVER_HNDL, EVENT_HNDL**, and **DATE_RECEPTION**. You can view the slot values for an event in the Tivoli Enterprise Console GUI, select the event, and click the **Details** button. **TEC_T_SLOTS_EVT** displays the following fields:

- **SLOT_NAME**
- **SHORT_SLOT_VALUE**
- **LONG_SLOT_VALUE**
- **SERVER_HNDL**
- **EVENT_HNDL**
- **DATE_RECEPTION**

TEC_T_SLOTS_EVT and Impact

Additional slot data for an event can be stored in **`TEC_T_SLOTS_EVT`**. The Event Reader polls **`TEC_T_EVT_REP`**. However, this means that a policy started by the Event Reader does not include the slot data. If you need this slot data for your

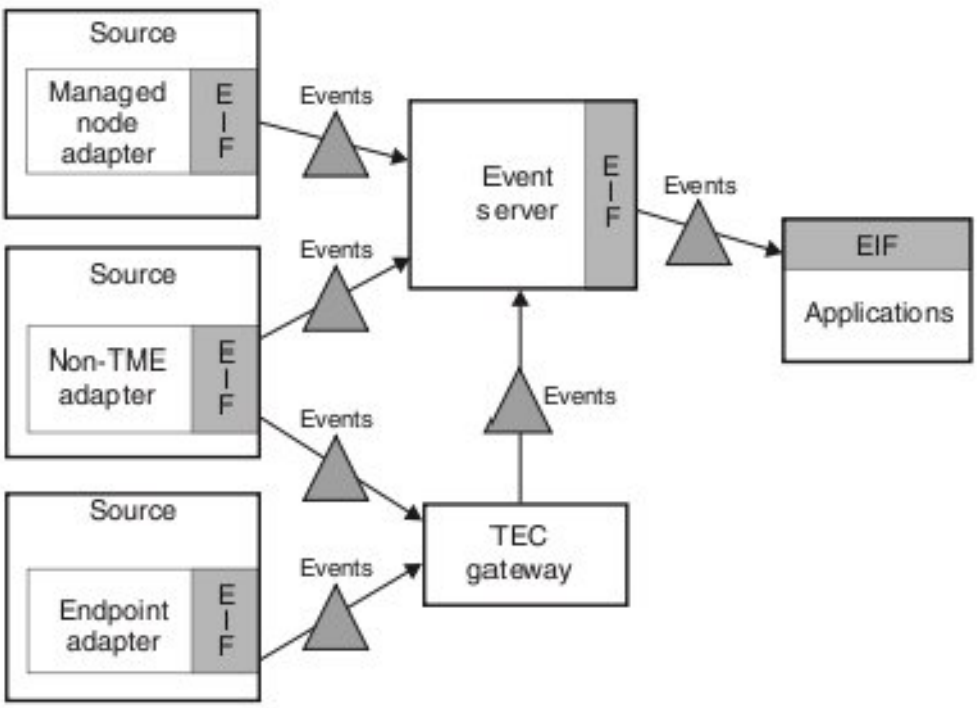
policy, there is an IPL function, **TEC.getSlots()**, for retrieving the extra slot data and adding it to the Event Container. You can change the values of these slots as you would normal Event Container (@) fields. The Tivoli Enterprise Console selects where the values go after the event is returned to the Tivoli Enterprise Console using the IPL function **TEC.returnTECEvent()**.

EIF API to Tivoli Enterprise Console

The various Tivoli Enterprise Console functions in Netcool/Impact uses the EIF API to get their new or updated events over to the Tivoli Enterprise Console. The Tivoli Enterprise Console server communication details that are required by the API are stored in the file `ImpactTECAgent.conf` which is normally found in the `$IMPACT_HOME/impact/etc` directory.

Events sent over to Tivoli Enterprise Console are first dumped into the table **TEC_T_EVT_REC_LOG** before the events are evaluated by the TEC classes or rules. The command **wtdumprl** dumps the contents of the **TEC_T_EVT_REC_LOG** table out to the command line. The **wtdumprl** command is useful for trying to determine why new or updated events that being sent over from Netcool/Impactare not making it all the way through to the table **TEC_T_EVT_REP**.

The EIF API is used to move Tivoli Enterprise Console events around. It is used in an Netcool/OMNIbus Tivoli Enterprise Console probe. Netcool/Impact uses the API to send events to the Tivoli Enterprise Console.



Tivoli Enterprise Console Classes

Events in **TEC_T_EVT_REC_LOG** first have their Classes checked. If an event has an invalid Class or **attribute=value** pairs that are not listed in the indicated Class, then the event is not processed any further.

The `TEC.returnTECEvent()` function uses the Class **IMPACT_Update**.

The `TEC.insertTECEvent()` function uses the Class **`IMPACT`**.

The `TEC.insertTECEventOfClass()` function uses the Class passed to the function.

Every event in Tivoli Enterprise Console belongs to a Class. The Class defines how the raw **`attribute=value`** pairs in an incoming event are handled. The Class defines what fields in **TEC_T_EVT_REP** that values from the incoming event go into. If there are extra slots defined in a Class, the Class also specifies how the remaining **`attribute=value`** pairs are put into **TEC_T_SLOTS_EVT**. Tivoli Enterprise Console Classes support inheritance. New Classes inherit all the field assignments and slots of their parents. If a raw event contains **`attribute=value`** pairs that are not defined in the Class, the event is dropped.

Note: You can view these dropped events by running the **`wtdumprl`** command on the Tivoli Enterprise Console host.

- IMPACT_Data
- IMPACT_Base
- IMPACT
- IMPACT_Update

Tivoli Enterprise Console Classes and Netcool/Impact

There are four new Tivoli Enterprise Console Classes defined for this integration.

- IMPACT_Data is derived from the base Tivoli Enterprise Console Class EVENT and adds the slots **`impact_Service_Desk_Ticket`**, **`impact_OnCall_Technician`** and **`impact_EventOperatorView`**. IMPACT_Data is the class you add additional slots to for event enrichment.
- IMPACT_Base is derived from IMPACT_Data and adds the slots **`impact_instance`** and **`impact_status`**. The slot **`impact_instance`** can be used to identify the Netcool/Impact cluster that updated the event. If you want to set a value for the **`impact_instance`** slot you must manually set the value. The slot **`impact_status`** is used by the Tivoli Enterprise Console Rules to set the STATUS value in an updated event. You do not have to set **`impact_status`**. The **`TEC.returnTECEvent()`** function sets **`impact_status`** to the value of **`@STATUS`** for you.
- IMPACT is derived from IMPACT_Base and adds the slot identifier. The identifier needs to set and is used to deduplicate events of Class IMPACT inserted by Impact. Exactly what identifier is set to is up to you. Populate it in the same manner you would populate the Netcool/OMNIbus identifier field.
- IMPACT_Update is derived from IMPACT_Base and adds the slots **`server_handle`**, **`event_handle`**, and **`date_reception`**.

These slots are all set by the **`returnTECEvent()`** function. The values are copied from @SERVER_HNDL, @EVENT_HNDL, and @DATE_RECEPTION. The values are used by the Tivoli Enterprise Console Rules. The slots shipping with the integration are examples. Add slots as necessary, that make sense for your environment.

Tivoli Enterprise Console rules

`Dedup_impact_events` is run on events with a Class of **`IMPACT`**. This rule uses the identifier slot in the IMPACT Class to deduplicate events that are inserted from Impact.

Impact_update_events is run on events with a Class of **IMPACT_Update**. **IMPACT_Update** is the Class assigned to all updated events sent back to the Tivoli Enterprise Console from Netcool/Impact. This rule uses the **server_handle**, **event_handle**, and **data_reception** slots defined in the **IMPACT_Update** Class to find and update the original Tivoli Enterprise Console event. The credibility field in the original event is set to 5. The event of **Class IMPACT_Update** sent over by Impact is then dropped.

Tivoli Enterprise Console rules are run on incoming events that are based on class. These rules can correlate across events in the Tivoli Enterprise Console server, update values in the Tivoli Enterprise Console server, delete events, and run external process. Tivoli Enterprise Console rules combine functionality that is found in OMNIbus rules files and OMNIbus procedural SQL.

Tivoli Enterprise Console Rules and Netcool/Impact

There are two new Tivoli Enterprise Console rules that are defined in this integration.

- **Dedup_impact_events** is run on events with a class of IMPACT. This rule uses the identifier slot in the IMPACT class to deduplicate events that are inserted from Impact.
- **Impact_update_events** is run on events with a class of IMPACT_Update.

IMPACT_Update is the class that is assigned to all updated events sent back to Tivoli Enterprise Console from Netcool/Impact. This rule uses the **server_handle, event_handle**, and **data_reception** slots that are defined in the **IMPACT_Update** class to find and update the original Tivoli Enterprise Console event. The event of class **IMPACT_Update** sent over by Netcool/Impact is then dropped.

Tivoli Enterprise Console functions

Netcool/Impact functions are used to either enrich existing Tivoli Enterprise Console events or send new events to Tivoli Enterprise Console using Tivoli Enterprise Console EIF API.

getSlots

This function takes an Event Container from a Tivoli Enterprise Console event, retrieves the extra slots from the **TEC_T_SLOTS_EVT** table, and adds them to the Event Container.

The function syntax:

```
TEC.getSlots(TEC_T_SLOTS_EVT EventContainer)
```

It should be used with events that have data in the slots table, which you want to work with. This function is optional.

returnTECEvent

This function is used to send an updated event back to the Tivoli Enterprise Console with a TEC Class of IMPACT_Update.

The function syntax:

```
TEC.returnTECEvent(EventContainer)
```

This function should be called instead of the returnevent() function.

insertTECEvent

This function takes a context and sends it over to the Tivoli Enterprise Console as an event of Class IMPACT.

The function syntax:

```
TEC.insertTECEvent(context)
```

Only context attributes, for example myContext.paintColor, whose name is in the list in the TEC_Insert_ValidSlots variable at the top of the Tivoli Enterprise Console Policy will be sent over to Tivoli Enterprise Console. Additional attributes of the context will be dropped by Impact. If the new event is sent over to the Tivoli Enterprise Console with slots not supported by the IMPACT Class, then the event will be dropped by the Tivoli Enterprise Console. The list in the IPL variable TEC_Insert_ValidSlots needs to be accurate.

insertTECEventOfClass

This function takes a context and send it over to Tivoli Enterprise Console as an event of the Class that is given in the function call.

The function syntax:

```
TEC.insertTECEventOfClass(context, class)
```

Note: If the context has attributes (for example, myContext.paintColor) not supported by the Class, then the event will be dropped by the Tivoli Enterprise Console.

Setting up integration with Tivoli Enterprise Console

To set up the integration, you complete a series of steps on Netcool/Impact and Tivoli Enterprise Console.

Procedure

1. Set up the integration on the Tivoli Enterprise Console:
   a. Add provided Classes to your TEC Rulesbase.

      For more information, see "Adding provided Classes to TEC Rulesbase" on page 18.
   b. Add provided Rules to TEC Rulesbase.

      For more information, see "Adding provided rules to TEC Rulesbase" on page 18.
   c. Confirm that your Tivoli Enterprise Console server is using a fixed communication port.

      For more information, see "Configuring the Tivoli Enterprise Console server to use fixed communication port" on page 18.
2. Set up the integration on Netcool/Impact.
   a. Update the `ImpactTECAgent.conf` file.

      For more information, see "Updating ImpactTECAgent.conf file" on page 18.
   b. Update the **TEC** policy.

      For more information, see "Updating the TEC policy" on page 19.
   c. Make the database connections.

For more information, see “Making database connections” on page 20.

d. Configure the event reader.
   For more information, see “Setting up Event Reader” on page 21.

Adding provided Classes to TEC Rulesbase

The new Classes are in the `IMPACT.baroc` file in `$IMPACT_HOME/impact/integrations/tec/rules` on your Impact Server.

Procedure

Follow the instructions in the Tivoli Enterprise Console documentation for adding these Classes to your TEC Rulesbase see the following link: http://www.ibm.com/tivoli/documentation.

Adding provided rules to TEC Rulesbase

The new rules are in the `IMPACT_tec.rules` file in `$IMPACT_HOME/impact/integrations/tec/rules` on your Impact Server.

Procedure

Follow the instructions in the Tivoli Enterprise Console documentation for adding these Rules to your TEC Rulesbase, see the following link: http://www.ibm.com/tivoli/documentation.

Configuring the Tivoli Enterprise Console server to use fixed communication port

The Tivoli Enterprise Console instance may need to be configured to use a fixed IP port.

Procedure

Edit the `/usr/local/Tivoli/bin/<platform>/TME/TEC/.tec_config` file.
If necessary, uncomment `"tec_recv_agent_port"`and restart your Tivoli Enterprise Console server. On Windows platforms this may not need to be done. Refer to the Tivoli Enterprise Console manuals for more information see the following link, http://www.ibm.com/tivoli/documentation.

Updating ImpactTECAgent.conf file

About this task

The Tivoli Enterprise Console Agent configuration file is called `ImpactTECAgent.conf` and can be found in `$IMPACT_HOME/impact/etc`.

```
BufEvtPath=/home/netcool/data/tecLogs/tecplus.log
BufferEvents=YES
TransportList=t1_
t1_Type=SOCKET
t1_Channels=c1_
c1_ServerLocation=tiv
c1_Port=5529
LogLevel=ALL
TraceLevel=ALL
LogFileName=/home/netcool/data/tecLogs/teclog.log
TraceFileName=/home/netcool/data/tecLogs/tectrace.log
```

You will need to update the following parameters:

- **BufEvtPath**: This is the path to the buffer file. The buffer file is used to hold inserts and updates when there is a communication problem with the Tivoli Enterprise Console server. If there is a problem communicating with the TEC server, you will see events buffered in this file.
- **c1_ServerLocation**: This is the IP address or hostname of the host running the Tivoli Enterprise Console server.
- **c1_Port**: This is the port on which the Tivoli Enterprise Console server will be listening for communications from Impact. Your Tivoli Enterprise Console instance may need to be configured to use a fixed port. Edit the file `/usr/local/Tivoli/bin/<platform>/TME/TEC/.tec_config` and uncomment `tec_recv_agent_port`. Restart your Tivoli Enterprise Console server. On Windows platforms this may not need to be done; refer to the Tivoli Enterprise Console manuals for more information see http://www.ibm.com/tivoli/documentation.
- **LogFileName**: Log file for the EIF API calls. If there is a communication problem with the Tivoli Enterprise Console server you will see `"Failed to connect to port <tec_port> on server "<tec_server>"."`

  Note: If everything is working you will see an error about a problem with the buffer file in the log file.
- **TraceFileName**: This contains detailed information about communications with the Tivoli Enterprise Console server. If there is communication problem with the Tivoli Enterprise Console server you will see a Connection Refused exception in this log.

Note: In a clustered Netcool/Impact environment, this file must be configured on each Netcool/Impact server in the cluster.

Updating the TEC policy

Five variables at the top of the **TEC** policy need to have their values updated to match your environment.

1. In the Tivoli Integrated Portal GUI, select the **TEC** project.
2. Select **Event Automation** > **Polices**.
1. Select the **TEC** policy and click the **Edit** icon. Look for these lines:

```
//TEC Class used to identify new inserts from Impact
TEC_InsertClass = "IMPACT";

//TEC Class used to identify that updates to an existing event are being passed
TEC_UpdateClass = "IMPACT_Update";

//valid slots for the TEC class named in TEC_InsertClass
TEC_Insert_ValidSlots = "severity, msg, hostname,
impact_MRO_Service_Desk_Ticket, impact_instance, identifier";

//valid slots for the TEC class named in TEC_UpdateCLass
TEC_Update_ValidSlots = "impact_MRO_Service_Desk_Ticket, severity, msg";

// full path to the ImpactTECAgent.conf file
ImpactTECAgentConfFileLocation = "/opt/ibm/netcool/impact/etc/ImpactTECAgent.conf";
```

- The first four variables are TEC-related. The default values work with the TEC Classes and Rules that ship with this integration.
- If you expand the slots that are supported by those classes and rules; you will need to update the variables **TEC_Insert_ValidSlots** and **TEC_Update_ValidSlots**.

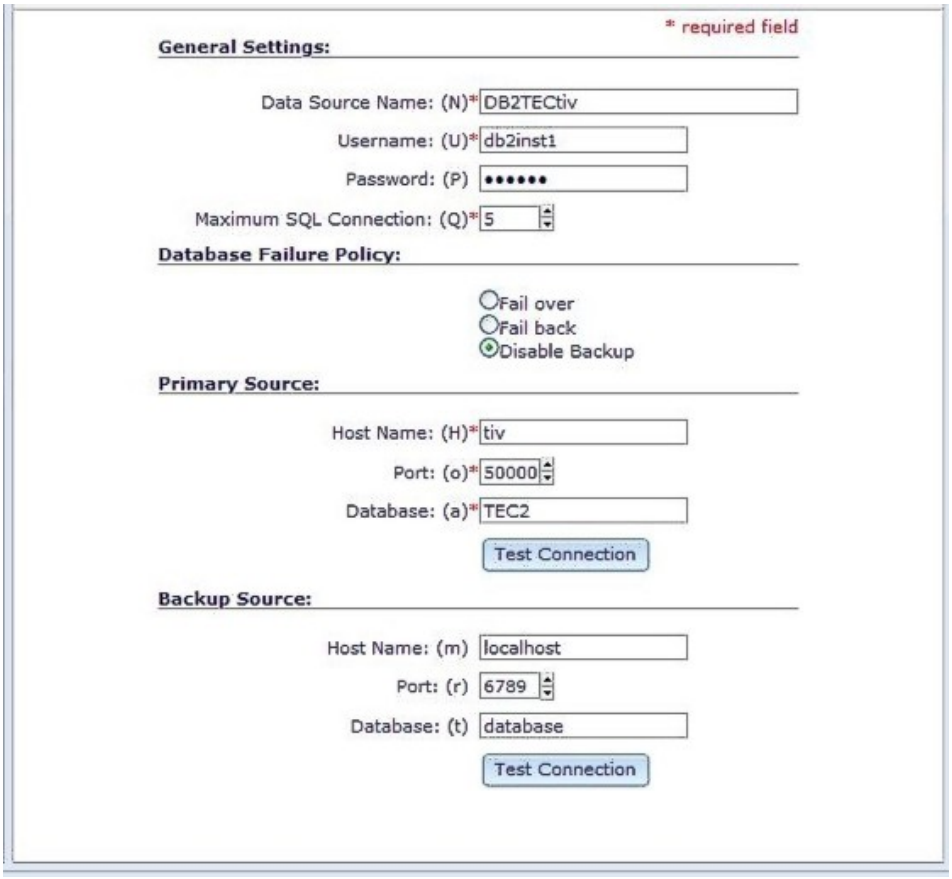
- The final variable, **ImpactTECAgentConfFileLocation**, is the full path to the configuration file used by Impact to communicate with your Tivoli Enterprise Console server.

Making database connections

Procedure

1. Configure a data source connection to the database used by your Tivoli Enterprise Console server.

   You need to configure two data types named **TEC_evt_rep** and **TEC_slots_evt** that use the **DB2TECtiv** data source. Your Tivoli Enterprise Console or database administrator can provide the required connection information.



   Note: If you are running Tivoli Enterprise Console on DB2, you may need to adjust your Impact Server depending on your Netcool/Impact and DB2 versions. See the Netcool/Impact documentation for more details.

2. Create a data type called **TEC_evt_rep**.
3. In the **Data Type Name** field, create the following data type `TEC_evt_rep`.

   The **Data Source Name** field is autopoulated with the data source `DB2TECtiv`. The **State** check box is **Enabled**.
4. The **TEC_evt_rep** data type connects to the table **TEC_T_EVT_REP**
5. Click the **New** button next to the **New Field** field.
6. Add the **DATE_RECEPTION**, **SERVER_HNDL**, and **EVENT_HNDL** as the key fields.
7. Click **OK** and then click **Save** to create the data type.

8. Create a data type called **TEC_slots_evt**. The name and typecase of the data type is important.
9. The **TEC_slots_evt** data type connects to the table **TEC_T_SLOTS_EVT**.
10. Click the **New** button next to the **New Field** field.
11. Add **DATE_RECEPTION**, **SERVER_HNDL**, **EVENT_HNDL**, and **SLOT_NAME** as the key fields.
12. Click **OK** and then click **Save** to create the data type.

Setting up Event Reader

New events or updated events are captured from Tivoli Enterprise Console using an event reader.

The event reader must point at the **TEC_evt_rep** data type:

1. In the navigation tree, expand **System Configuration** > **Event Automation** click **Services** to open the **Services** tab.
2. In the **Services** tab, select the **DatabaseEventReader** service.
3. In the **Service Name** field, type `TECEventReader`.
4. In the **Data Source** field, select **DB2TECtiv**.
5. In the **Data Type** field, select **TEC_evt_rep**.
6. Enable the **Start up** and **Service log** check boxes.
7. Click the **Event Mapping** tab.
8. If you want Netcool/Impact to process updated events, select the **Get updated events** check box.
9. Select the field **LAST_MODIFIED_TIME** as the **TimeStamp** field.
10. The **Key Field** is irrelevant as the Tivoli Enterprise Console integration does not write back to the Tivoli Enterprise Console database directly. You can use **DATE_RECEPTION** or really anything.
11. Click the **New** button next to **New Mapping** to open the Create a New Event Filter window.
12. In the Filter Expression field, type **CREDIBILITY**.

    The **CREDIBILITY** field can be used as an "impact" flag. **CREDIBILTITY** is **`0`** by default and events updated by Netcool/Impact have this value set to **`5`** by the new Tivoli Enterprise Console rules provided with the integration. *STATUS* values in Tivoli Enterprise Console are as follows:

    - 0- opened
    - 20- acknowledged
    - 30- closed
13. In the **Policy Name** list, select, **TECexample_update**.
14. Select the **Active** check box. Click **OK** to add the event mapping to the event reader.
15. Click **Save**.

Testing the setup

After you complete all of the Tivoli Enterprise Console and Netcool/Impact setup steps, you are ready to test the integration.

About this task

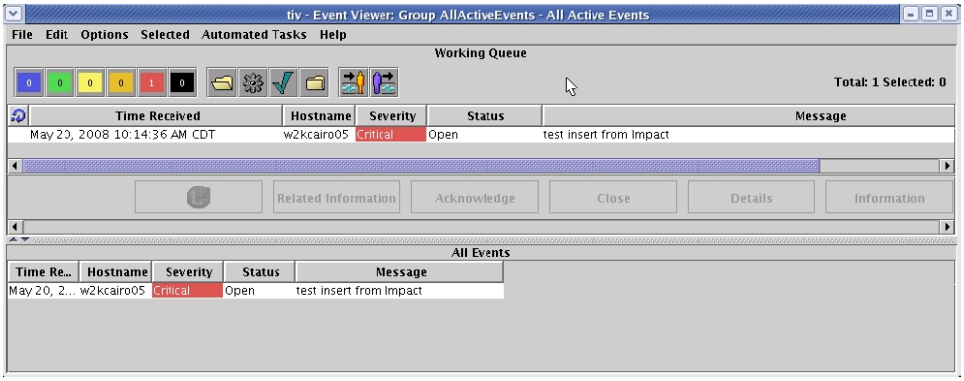
There are three test policies, TECexample_insert, TECexample_insert_2, and TECexample_update.

- TECexample_insert will insert a new event into the Tivoli Enterprise Console server. The new event will have a host name of w2kc insario05 and a message of “test insert from Impact”. The extra slot impact_MRO_Service_Desk_Ticket will read “none.”
- TECexample_insert_2 will also insert a new event into the Tivoli Enterprise Console but uses the TEC.insertTECEventOfClass() function, which can be used to insert an event of any Class.
- TECexample_update policy will update a Tivoli Enterprise Console event brought in by the Event Reader. The updated event will have “updated by Impact” added to the message field and the impact_MRO_Service_Desk_Ticket will be changed to “Ticket # 12345”.

Procedure

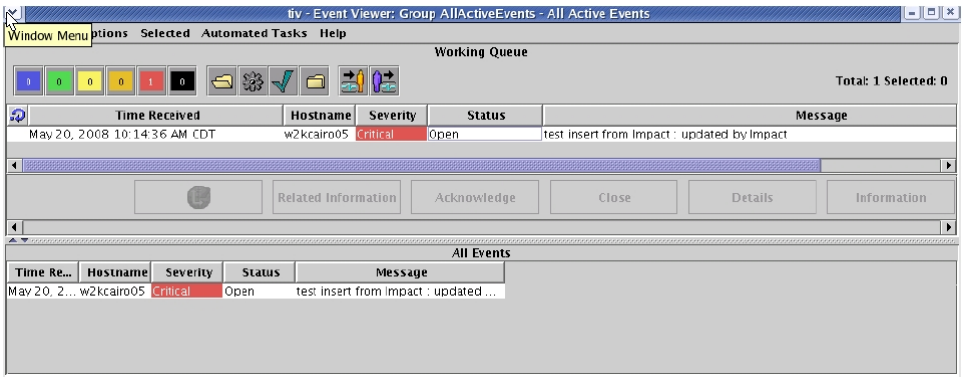
1. Testing Inserts.

   Run the Impact policy TECexample_insert. This policy will insert a new event into TEC, which will look like this:



2. Testing the reader.

   Now configure the Event Reader to run the policy TECexample_update on events where CREDIBILITY < 5 and STATUS < 30 and HOSTNAME = ‘w2kcario05'. Then start the Event Reader. The Reader processes the event. After your Tivoli Enterprise Console refreshes, the event should look like this:



   The Message field has been updated.

Results

Your integration is working!

Troubleshooting

Use the following checklist to verify that you have completed all the installation steps correctly.

Installation checklist

ImpactTECAgent.conf

- Are you using the correct hostname/IP for the TEC server?
- Are you using the correct port for TEC server?
- Is the TEC server set to use a fixed port?
-

TEC Policy updated to match environment?

- Do the slots listed in the IPL variable TEC_Insert_ValidSlots match those in the IMPACT Class?
- Do the slots listed in listed in the IPL variable TEC_Update_ValidSlots match those in the IMPACT_Update Class?
- Is the full path in the IPL variable ImpactTECAgentConfFileLocation correct?

Data Source to TEC Database

Try the Test Connection button.

Data Type TEC_evt_rep

- Can you view data items from the table from TEC_T_EVT_REP?
- Is the Data Type correctly named TEC_evt_rep?

Data Type TEC_slots_evt

- Can you view data items from the table TEC_T_SLOTS_EVT?
- Is the Data Type correctly named TEC_slots_evt?

Troubleshooting TEC Side

Class and Rule confirmation

1. Determine the Rulesbase in use: `[root@tiv ~]# wrb -lscurrb`.
2. List all of the classes whose names contain IMPACT: `wrb -lsrbclass -detailed IMPACT <Rulesbase name> | more`. You should see **`IMPACT_Data, IMPACT_Base, IMPACT, and`** and **`IMPACT_Update`**.
3. List all of the rules in use whose names contain IMPACT: `wrb -lsrbrule IMPACT <Rulesbase name>`. You should see **`IMPACT_tec.rls`**.

Logs to check

- `$NCHOME/log/impactserver.log` If everything was installed/created correctly there should not be much action in the `impactserver.log` file beyond the java calls from the Java DSA.
- `$IMPACT_HOME/log/tecbuf.log` If there is a communication problem with the TEC server you will see events from Impact being buffered in this file.

- $IMPACT_HOME/log/tectrace.log If there is communication problem with the TEC server you will see a Connection Refused exception. There is a lot of activity in this log– scroll way down to find the exception.
- $IMPACT_HOME/log/teclog.log If there is a communication problem with the TEC server you will see

  ```
  Failed to connect to port <tec_port> on server "<tec_server>.
  ```

  If everything is working you may see an error about a problem with the buffer file. This is run on the TEC server at the command prompt. It dumps out all of the events that came into the TEC server.
- **wtdumprl** command. If bad slots are being passed, the events will not make it into the TEC Console but **wtdumprl**will show a parsing failed error:

  ```
  PARSING_FAILED~'Line 1: Slot myBadSlot not defined in class
  ```

Customizing the integration

You can customize your integration by:

- Adding new slots to existing TEC Classes. For more information, see "Adding new slots to existing Tivoli Enterprise Console."
- Updating the IMPACT Class. For more information, see "Updating IMPACT Class" on page 25.
- Running a New Slot Check List. For more information, see "New Slot Check List" on page 25.

Adding new slots to existing Tivoli Enterprise Console

Adding new slots to existing TEC Classes can be easily done using multiple inheritance. Example

```
TEC_CLASS :
   alphaOne ISA EVENT;
END

TEC_CLASS:
 IMPACT_Data ISA EVENT
 DEFINES {
  impact_MRO_Service_Desk_Ticket: STRING;
  impact_OnCall_Technician: STRING;
  impact_EventOperatorView: STRING;
 };
END

TEC_CLASS :
   BetaTwo ISA [alphaOne, IMPACT_Data];
END
```

In the example, BetaTwo inherits from both alphaOne and IMPACT_Data. Due to Class Inheritance, any Class derived from BetaTwo will inherit all slots from alphaOne and IMPACT_Data. In this way, slots for use with Impact can quickly be added to your Class structure.

Via the IPL function returnTECEvent(), Impact sends updated events to TEC with a Class of IMPACT_Update. This happens regardless of the Class of the original event. If you add slots to a Class and want Impact to be able to update them, ensure the slots are added to the Class IMPACT_Update. It is suggested that you add all of your Impact-related slots to the IMPACT_Data Class. IMPACT_Update inherits from IMPACT_Data.

Updating Impact to handle the new slots

After adding the slots to TEC, you will need to update the TEC Policy in Impact. The IPL function returnTECEvent() sends updated events to TEC with a Class of IMPACT_Update. This happens regardless of the Class of the original event. Only slots listed in the policy variable TEC_Update_ValidSlots will be sent to TEC as part of the updated event. All other slots are dropped by the Policy. If you add slots to TEC Classes and you want these slots to be updated by Impact, you must update the variable TEC_Update_ValidSlots in the TEC Policy.

Updating the Impact_update_events Rule to handle new slots

When Impact sends an updated event to TEC, the Class used is IMPACT_Update. The Rule Impact_update_events runs on events of Class IMPACT_Update. This Rule uses the IMPACT_Update slots server_handle, event_handle, and date_reception to locate the original event and update all of the slots specified in the Rule.

It then drops the event of Class IMPACT_Update. If you add new slots to a Class and you want Impact to be able to update those slots in TEC then you need to make sure that the Rule Impact_update_events is updated to handle the new slots.

Updating IMPACT Class

New events sent to TEC via the insertTECEvent() function will have a Class value of IMPACT. The integration provides 3 example slots to work with

- impact_MRO_Service_Desk_Ticket
- mpact_instance
- identifier

If you add additional slots to the IMPACT Class, you must remember to update the TEC_Insert_ValidSlots variable at the top of the TEC Policy or those new slots will not be sent to TEC.

The default value for TEC_Insert_ValidSlots is:

```
TEC_Insert_ValidSlots =
"severity, msg, hostname, impact_MRO_Service_Desk_Ticket, impact_instance,
identifier";
```

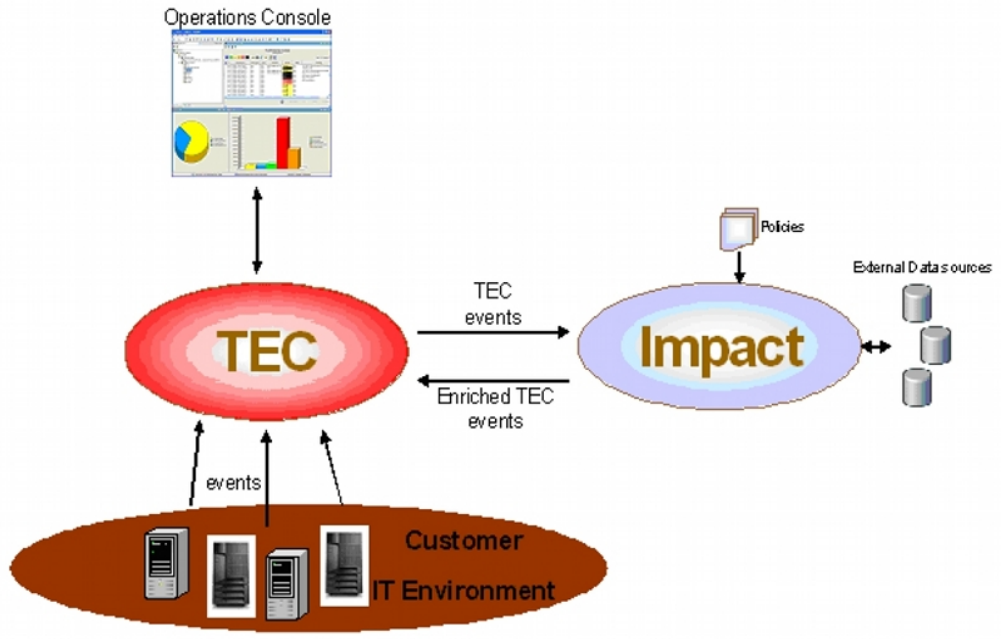
New Slot Check List

1. Update the TEC Class(es) that you want to support this new slot and ensure the IMPACT_Update Class has this slot (using IMPACT_Data and multiple inheritance makes this easy).
2. Update the impact_update_event Rule so that the new slot is updated by the Rule
3. Run all the necessary TEC commands and restart the Tivoli Enterprise Console server
4. Update the values in the variable TEC_Update_ValidSlots in the TEC Policy

Examples of integration architectures

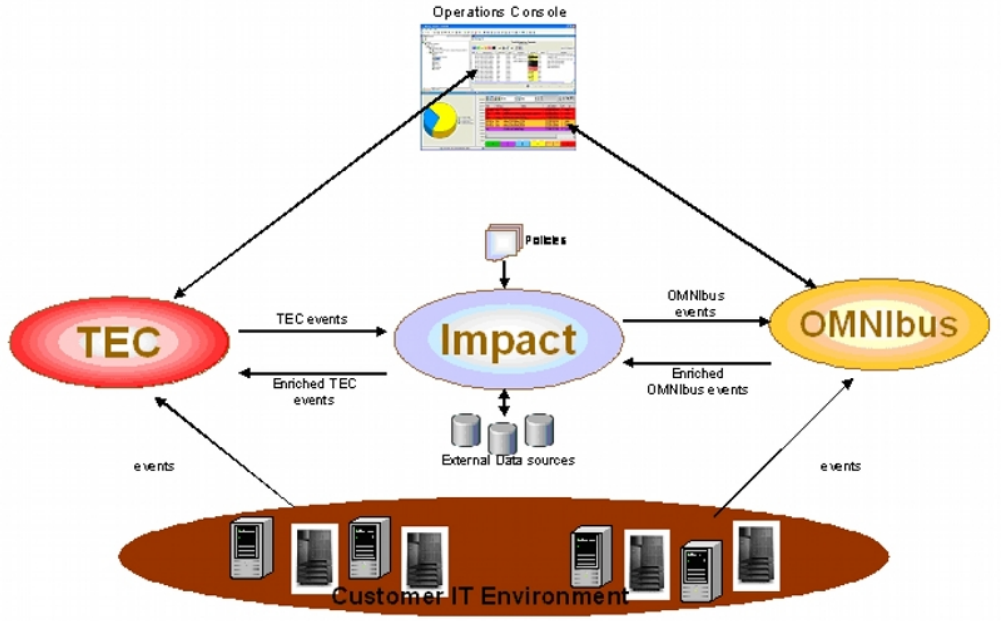
Impact and Tivoli Enterprise Console

Impact is being used to correlate and enrich Tivoli Enterprise Console events with external data. Impact policies fetch data in real time from the trusted source. There are no fact files to maintain.



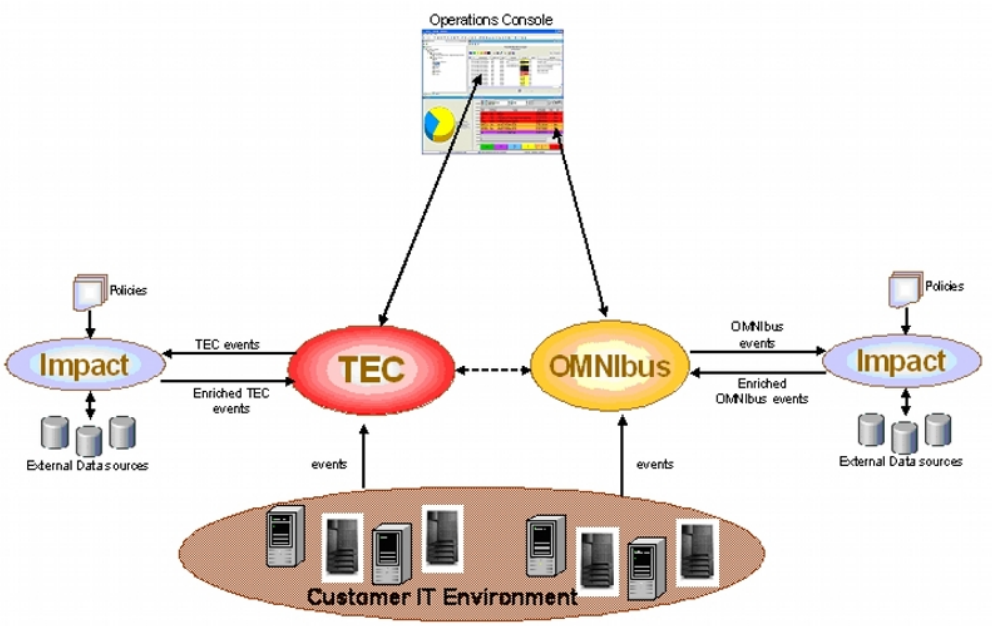
Tivoli Enterprise Console and OMNIbus sharing an Impact Server

Tivoli Enterprise Console and OMNIbus sharing an Impact Server



Tivoli Enterprise Console and OMNIbus with dedicated Impact Servers

Tivoli Enterprise Console and OMNIbus with dedicated Impact Servers



Chapter 5. Integrating Netcool/Impact with Webtop

Using the Netcool/Impact functions in this integration you can use Netcool/Impact to complete the following tasks:

- Provision Webtop content such as maps, entities, and tools.
- Administer Webtop access control in either Webtop 1.3 or Webtop 2.0/2.1/NGF.
- Create NGF pages that containWebtop content.

Netcool/Impact functions

This solution provides functions that you can use to complete the following tasks:

- Create complex XML structures as IPL objects and turn those objects into XML strings
- Save those XML strings into files locally using JRExec or remotely through SSH command/response
- Push the files to run waapi or the ngf_apilocally through JRExec or remotely through SSH command/response
- Optionally delete the files afterwards

These functions are stored in several example policies, which can be found in the **WebTop** project.

There is example code showing how to create Webtop content, administer Webtop, configure NGF, and create NGF pages (`psml` files).

Supported operating systems

The Webtop integration is supported by Tivoli Netcool/Impact 6.1.1.

The most up-to-date information about supported hardware, software, browsers, and operating systems is provided by the IBM® Software Product Compatibility Reports at: http://pic.dhe.ibm.com/infocenter/prodguid/v1r0/clarity/index.html

For more information about running the Software Product Compatibility Reports, see the *Overview and Planning* section of the Tivoli Netcool/Impact wiki at: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home/wiki/Tivoli%20Netcool%20Impact/page/Overview%20and%20Planning?lang=en

Note: Note for *nix users, the integration was not tested on any *nix other than CentOS 4.5. The only complications might be the behavior of the echo command on different UNIX operating systems. Tweak the function `cleanTextForEcho(text)` on the **UTIL** policy if you must. WAAPI requires `$WAAPI_HOME` to be set and ngf_api requires `$NCHOME` to be set for the user who is running the Netcool/Impact commands.

Note: For Windows users, all interaction with `WAAPI` or NGF_API on a Windows box is through SSH Command/Response. The Windows policy code was developed against a Windows 2003 server running openSSH that provides Windows style pathing, (`c:\blah`). openSSH can be downloaded from http://sshwindows.sourceforge.net/. `$NCHOME` and `$WAAPI_HOME` must be set on the

Windows box before you start the openSSH Service. Always use short file names in your paths when you work with Windows (`c:\progra~1\ibm\netcool`).

Note: This integration has not been tested with Webtop 2.2 running in Tivoli Integrated Portal.

Setting up the integration

Use this procedure to set up the integration with Webtop.

Procedure

1. Import the policies into your Netcool/Impact cluster.
2. Configure the policies for your environment by editing the `getENV` function in the **WEBTOP**, **NGF**, and **PSML** policies.

   If you are running Webtop or NGF on Windows, you need to install openSSH on the Windows host. See, http://sshwindows.sourceforge.net/.

Chapter 6. Integrating Netcool/Impact and WebSphere Business Events

When installing Netcool/Impact and WebSphere Business Events on the same computer, install WebSphere Business Events first and make sure it is running before installing Netcool/Impact.

Configuring WebSphere Business Events

Procedure

1. Load the Websphere Administrative Console.
2. Create a new jms Topic Connection Factory named `ImpactTopicConnectionFactory` with a jndi name of `jms/ImpactTopicConnectionFactory` associated with the WbeBus.
3. Create a new jms topic to receive messages from Netcool/Impact called `impactTopic` with a jndi name of `jms/impactTopic`. Associate this topic with the `ImpactTopicConnectionFactory`.
4. Create a new jms topic to send messages to Netcool/Impact called `wbeTopic` with a jndi name of `jms/wbeTopic`. Associate this topic with the `ImpactTopicConnectionFactory`.
5. Load the WBE Design Data tool.
6. Load the Netcool/Impact **`Integration.xml`** file.
7. Add all of the Objects to the WebSphere Business Events Repository.

Integrating Netcool/Impact with the external JMS queue

You must install the resource adapter on the eWAS where Netcool/Impact is running before you can integrate Netcool/Impact with the external JMS queue.

About this task

Use the **`sibc.jmsra.rar`** file from the eWAS version 7.0.0.17 or later.

Procedure

1. Obtain the **`sibc.jmsra.rar`** file from this URL: ftp://l3.hursley.ibm.com/PMC/fixes/eWAS-SIBRA/sibc.jmsra-7.0.0.15+PM32130.rar.
2. You can also use the **`sibc.jmsra.rar`** file from the WebSphere Application Server 7.5 installation, from the `<WAS_INSTALL>/runtimes` directory.
3. Copy the **`sibc.jmsra.rar`** file to the scripts folder, `<TEMP>/scripts`.
4. Next, create the **`updateRA_patch_On_Impact_eWASServer.jacl`** file.
   a. Open a text editor.
   b. Copy and paste the following text into the text file.

```
##########
#
# This script has to be run on the Netcool Impact eWAS server.

##########
# Here we refer to the SIB JMS rar file. provide the full details and
# ensure the file name is proper, for example,
the directory where you have saved the file.
```

```
set rarFile "C:/temp/sibc.jmsra.rar"

##########
# Here we refer to the server , cell and node details of Impact eWAS
# server this will not change unless there is any other version of
# Impact is installed. Ensure the the details before running the script.
set serverName "server1"
set theCell "ImpactCell"
set theNode "ImpactNode"

##########
puts "Uninstall any existing RA"
set ra [$AdminConfig getid "/Cell:$theCell/Node:$theNode/J2CResourceAdapter:
Default messaging provider/"]
if { $ra != "" } { $AdminConfig uninstallResourceAdapter $ra }

##########
puts "Save the configuration"
$AdminConfig save

##########
puts "Uninstall any existing RA"
set ra [$AdminConfig getid
"/Cell:$theCell/Node:$theNode/J2CResourceAdapter:
WebSphere Application Server JMS Resource Adapter/"]
if { $ra != "" } { $AdminConfig uninstallResourceAdapter $ra }

##########
puts "Save the configuration"
$AdminConfig save

##########
puts "Install the RA"
set option { -rar.name "WebSphere Application Server JMS Resource Adapter"
-rar.desc
"WebSphere Application Server service integration bus
JMS resource adapter" }
$AdminConfig installResourceAdapter $rarFile $theNode $option

##########
puts "Get a reference to the newly created RA"
set ra [$AdminConfig getid
"/Cell:$theCell/Node:$theNode/J2CResourceAdapter:
Default messaging provider/"]

##########
puts "Save the configuration"
$AdminConfig save

puts "Done."
```

c. Save the text file and rename it as, **updateRA_patch_On_Impact_eWASServer.jacl**.

5. From the <TEMP>/scripts folder, open the **updateRA_patch_On_Impact_eWASServer.jacl** file.
6. Update the **rarFile** property to the complete path of the scripts folder using the following declaration: `set rarFile "<TEMP>/scripts/sibc.jmsra.rar"`

   The path within the inverted commas must be the complete path to the `sibc.jmsra.rar` file.

   Note: Ensure that you use the forward slash (/) to specify the file path.

7. Go to the <TEMP>/scripts/sibc.jmsra.rar folder, and run the following command from the command line: <IMPACT_PROFILE>/bin/wsadmin -f updateRA_patch_On_Impact_eWASServer.jacl -username wasadmin -password netcool
8. Restart the Impact server after running the script.

Using the integration

Included in the Integration on the Netcool/Impact side is a policy library called **WBE**. This policy includes two functions. One is SendEventToWBE and the other is ParseWBEMessage.

SendEventToWBE takes a Netcool/Impact object and creates a WebSphere Business Events JMS message and sends it to WebSphere Business Events using the configured **SendToWBE** data type.

ParseWBEMessage takes a JMS message received from WebSphere Business Events and converts it to a Netcool/Impact object.

On the WebSphere Business Events side there are two touch points. One is Netcool/Impactt Event and the other is Netcool/Impact Action.

Netcool/Impact Event listens for the Netcool/Impact event over the JMS bus and creates an Intermediate Object called Netcool Omnibus Event.

Netcool/Impact Action sends data to Netcool/Impact over the JMS bus.

Chapter 7. Working with IPL to XML functions

Data source adapters allow for access to data from a wide variety of sources. In many cases, the data is retrieved from SQL data sources and delivered as Impact Policy Language (IPL) objects (contexts). One challenge for policy writers has been converting these IPL contexts into XML strings for applications with interfaces that expect data in XML format. To facilitate this task, a set of IPL to XML functions has been developed that can be used to generate an XML string from an IPL object.

IPL to XML functions overview

You use IPL to XML functions to generate an XML string from an IPL object. What effectively happens is a top-level IPL object, referred to as the XML document object, is transformed into XML. IPL objects nested within the document object, referred to as element objects, become XML elements. The functions are used to create the XML document and element objects and to set XML attributes, content, and comments.

XML document object

The XML document object is the base object. XML element objects, attributes, content, and comments are added to the XML document object. It is the XML document object that is converted into an XML string.

For information about how to create the XML document object, see "Creating the XML document object" on page 36.

XML element objects

XML element objects are added to the XML document object or to each other (for nested XML elements). After you created and added an XML element object you can use other functions to add XML attributes, content, and comments to it. For more information about the IPL to XML functions used to create and add XML element objects, see "Adding a sub element" on page 36 and "Creating an unassociated element" on page 37.

Adding XML attributes to element objects

You can use on of three methods to add attributes to XML element objects. The simplest method is to pass an element object, attribute name, and attribute value to the `IPLtoXML.addAttribute()` function. For more information about this method, see "Adding XML attributes to element objects, simple approach" on page 37.

The second method requires creating a separate XML attribute object and adding that XML attribute object to the XML element object using the `IPLtoXML.addAttributeObject()` function. For more information about how to use this function, see "Adding XML attributes to element objects that use Attribute objects" on page 38. This method is useful in cases where you expect to use the same attribute and value in many places in your XML.

You use the third method to add several attributes at once using the `IPLtoXML.addOrgNodeAttributes()` function. Use this function to quickly take data

from a data type lookup and add all the fields to an XML element object as attributes. For more information about this method, see "Adding XML attributes to element objects adding attributes from an OrgNode" on page 39.

Adding XML content to element objects

You can add content to any element and append additional content to it later. For more information about how to add and append content to element objects, see "Adding the content to an XML element object" on page 39 and "Appending content to XML element objects" on page 40.

XML comments

Using the `addCommentToElement` function you can add comments to any XML element. The function also puts in the XML commenting code (`<!-- -->`) for you so do not have to add it manually.

For more information about using this function, see "Adding XML comments to element objects" on page 40.

Nesting XML elements

In cases where you created a stand-alone XML element object using the `newElement` function you can still nest that XML element object using the `addElement` function. In most cases you, will not be creating stand-alone objects using the `newElement` function but instead will be creating and nesting XML element objects at the same time using the `newSubElement` function.

For more information about adding XML element objects to each other, see "Adding XML element objects to each other (nesting)" on page 41.

Creating the XML document object

Procedure

Use this function to create the XML document object:

```
IPLtoXML.newDocument(myXMLDocumentObject)
```

The `myXMLDocumentObject` variable becomes the new XML document object.

Example

```
newDocument(REM_Album);
```

Adding a sub element

This function creates an XML element object and nests it within the parent element object in one step.

Procedure

To add a sub element, use this function:

```
IPLtoXML.newSubElement(parentElement, myElement, elementType)
```

where

parentElement
This element or document object must exist.

myElement
This variable becomes the new XML element of type `elementType` nested within the `parentElement`.

elementType
The type of XML element to create.

Example

IPL:

```
newDocument(myCars);
newSubElement(myCars, myElement, "Honda");
newSubElement(myElement, driver1, "susi");
```

Generated XML:

```
<Honda><susi/></Honda>
```

Creating an unassociated element

Rather than using this function to create an unassociated element you can use the `newSubElement` function to create and nest an XML element object in one step.

Procedure

To create an unassociated element, use this function:

```
IPLtoXML.newElement(myElement, elementType)
```

where

myElement
This variable becomes the new XML element of type `elementType`.

elementType
The type of XML element to create.

Example

IPL:

```
newElement(myElement, "Honda");
```

Generated XML:

```
<Honda/>
```

Adding XML attributes to element objects, simple approach

Procedure

To add XML attributes to an element object, use this function:

```
IPLtoXML.addAttribute(myElementObject, attributeName, attributeValue)
```

where

myElementObject
The element to add the attribute to.

attributeName
The string name of the attribute to add.

attributeValue
The value of the attribute.

Example

IPL:

```
newElement(myForester, "Subaru");
addAttribute(myForester, "year", 2003);
```

Generated XML:

```
<Subaru year="2003"/>
```

Note: Use `addAttribute` when you want to avoid creating the attribute object.

Adding XML attributes to element objects that use Attribute objects

To add XML attributes to an element object that Attribute objects follow this procedure.

Procedure

1. Create the attribute object. Use the following function:

   ```
   IPLtoXML.newAttributeObject(attributeObject, attributeName, attributeValue)
   ```

   where

   attributeObject
   This variable becomes the new XML attribute object.

   attributeName
   The name of the attribute.

   attributeValue
   The value for the attribute.

   IPL:

   ```
   newAttributeObject(carYear, "year", 2003);
   ```

   Generated XML:

   ```
   year="2003"
   ```

   Note: Attribute objects are useful because they can then be reused and added to multiple elements in your code.
2. Add the attribute object to an element object. Use the following function:

   ```
   IPLtoXML.addAttributeObject(myElementObject, attributeObject)
   ```

   where

   myElementObject
   The XML element object to which the attribute is added

   attributeObject
   The XML attribute object that is added to `myElementObject`.

   IPL:

   ```
   newElement(myForester, "Subaru");
   newAttribute(carYear, "year", 2003);
   addAttributeObject(myForester, carYear);
   ```

Generated XML:

```
<Subaru year="2003"/>
```

Adding XML attributes to element objects adding attributes from an OrgNode

Procedure

To add attributes from an OrgNode, use this function:

```
IPLtoXML.addOrgNodeAttributes(elementObject, OrgNode)
```

where

elementObject
The element object to add the attributes to.

OrgNode
An IPL object whose fields you want to add to element as attributes. The object could have come from a lookup or could have been built using `newobject();`.

Example

IPL:

```
newElement(myForester, "Subaru");
og=newobject();
og.color="blue";
og.doors=4;
og.turbo="no";
addOrgNodeAttributes(myForester, og);
```

Generated XML:

```
<Subaru color="blue" doors="4" turbo="no"/>
```

Adding the content to an XML element object

Content can be added to any element.

Procedure

To add the content to an XML element object, use this function:

```
IPLtoXML.setContent(myElementObject, content)
```

where

myElementObject
The XML element object to add the content to.

content
The text string to add to `myElementObject` as content.

Example

IPL:

```
newElement(myForester, "Subaru");
setContent(myForester, "Extra Car");
```

Generated XML:

```
<Subaru>Extra Car</Subaru>
```

Appending content to XML element objects

Additional content can be appended to any element.

Procedure

To append the content to an XML element object, use this function:

```
IPLtoXML.appendContent(myElementObject, content)
```

where

myElementObject
The XML element object to append the content to.

content
The text string to append to the existing content.

Example

IPL:

```
newElement(myForester, "Subaru");
setContent(myForester, "Extra Car");
appendContent(myForester, " driven by Dad");
```

Generated XML:

```
<Subaru>Extra Car driven by Dad</Subaru>
```

Adding XML comments to element objects

Procedure

To add XML comments to element objects, use this function:

```
IPLtoXML.addCommentToElement(myElementObject, comment)
```

where

myElementObject
The XML element object to add the XML comment to.

comment
The text string that becomes the XML comment. XML remarking code `<!-- -- >` is added by the function. Do not add it yourself.

Example

IPL:

```
newElement(myForester, "Subaru");
addCommentToElement(myForester, "mom drives the element");
```

Generated XML:

```
<Subaru><!-- mom drives the element --></Subaru>
```

Adding XML element objects to each other (nesting)

Procedure

To nest an XML element object use this function:

```
IPLtoXML.addElement(dore, myElementObject)
```

where

dore The XML document or element object to add `myElementObject` to.

myElementObject
The XML element object that is added to the `dore`.

Example

IPL:

```
newElement(myCars, "Cars");
newElement(myForester, "Subaru");
addElement(myCars, myForester);
```

Generated XML:

```
<Cars><Subaru/></Cars>
```

Generating XML strings from document objects

Procedure

To generate an XML string from the document object, use this function:

```
IPLtoXML.generateXML(xmlPiece, XML)
```

where

xmlPiece
Either an entire XML document object or just one XML element object.

XML The variable to hold the XML string that is generated from `xmlPiece`.

Example

IPL:

```
STACK=NewJavaObject("java.util.Stack", {}); //
this global variable is required for the IPLtoXML conversion
newDocument(carXML);
newElement(myForester, "Subaru");
addElement(carXML, myForester);
addCommentToElement(myForester, "mom drives the element");
generateXML(carXML, Output);
```

This XML is generated in the Output variable:

```
<?xml version="1.0" encoding="UTF- 8"?><Subaru><!-- mom drives
the element--></Subaru>
```

Replacement of default XML entities

The function `replaceEntities()` replaces the following default XML entities in XML content and attributes:

- `&` ampersand, replaced with `&`
- `<` less than, replaced with `<`
- `>` greater than, replaced with `>`
- `'` apostrophe, replaced with `&apos;`
- `"` quotation mark, replaces with `&quot;`

If you have additional entities that need to be replaced then edit the function `replaceEntities(x)` within the IPL to XML policy.

Element ordering in XML

The order in which elements at any particular nesting depth are added to the generated XML is based on the element object variable name.

Let us take the following policy for example:

```
IPLtoXML.newDocument(theWeather);
IPLtoXML.newSubElement(theWeather, system, "weatherSystem");
IPLtoXML.newSubElement(system, sub01, "tornado");
IPLtoXML.newSubElement(system, sub05, "thunderstorm");
IPLtoXML.newSubElement(system, SUB, "hail");
IPLtoXML.addComment(sub05, "bad thunderstorms and a tornado
where I live today");
```

The resulting XML looks like this example:

```
<weatherSystem><hail/><tornado/><thunderstorm><!-- bad
thunderstorms and a tornado where I live today
--></thunderstorm></weatherSystem>
```

The hail, tornado, and thunderstorm elements are all at the same depth in the XML nesting. The hail element was added first, element object name `SUB`. The tornado element was added second, element object name `sub01`. The thunderstorm element was added last `sub05`. If the element order in the XML is important then name the element objects carefully.

Examples of IPLtoXML functions usage

This section contains three examples of usage of IPLtoXML functions.

A simple example

This example shows how the different `IPLtoXML` functions can be used to generate a simple XML structure.

```
STACK=NewJavaObject("java.util.Stack", {});//required for
recursion in generateXML function
IPLtoXML.newDocument(docObj);
IPLtoXML.newElement(cars, "cars");
IPLtoXML.addElement(docObj, cars);
IPLtoXML.addAttribute(cars, "familyName", "Daniel");
IPLtoXML.setContent(cars, "the cars owned by a family");
IPLtoXML.newElement(subaru, "car");
IPLtoXML.newElement(honda, "car");
```

```
IPLtoXML.addElement(cars, subaru);
IPLtoXML.addElement(cars, honda);
IPLtoXML.newAttributeObject(carLocation, "location", "in the
garage");
IPLtoXML.addAttributeObject(subaru, carLocation);
IPLtoXML.addAttributeObject(honda, carLocation);
IPLtoXML.addAttribute(honda, "color", "black");
IPLtoXML.addAttribute(honda, "doors", 4);
IPLtoXML.addAttribute(honda, "driver", "susi");
IPLtoXML.addAttribute(honda, "model", "honda element");
foresterDetails=newobject();
foresterDetails.color="blue";
foresterDetails.doors="4";
foresterDetails.driver="tom";
foresterDetails.model="subaru forester";
IPLtoXML.addOrgNodeAttributes(subaru, foresterDetails);
IPLtoXML.newElement(hondaDriver, "Driver");
IPLtoXML.addAttribute(hondaDriver, "Name", "Susan Daniel");
IPLtoXML.addAttribute(hondaDriver, "Age", 33);
IPLtoXML.addCommentToElement(hondaDriver, "mother of
twins");
IPLtoXML.addElement(honda, hondaDriver);
IPLtoXML.generateXML(docObj, xml);
log(xml);
```

The results of logging the variable XML:

```
07 Oct 2007 17:11:06,888: SynchronousMessageProcessor:
Parser log: <?xml version="1.0" encoding="UTF-8"?><cars
familyName="Daniel">the cars owned by a family<car
color="blue" doors="4" driver="tom" location="in the garage"
model="subaru forester"/><car color="black" doors="4"
driver="susi" location="in the garage" model="honda
element"><Driver Age="33" Name="Susan Daniel"/><!-- mother
of twins -- ></car></cars>
```

ObjectServer event example

This example shows how `IPLtoXML` is run on an event from the Netcool/OMNIbus ObjectServer. Let us assume that we start with an Netcool/OMNIbusevent generated from the Netcool/OMNIbus Simnet probe.

1. Create an XML document object:

   ```
   IPLtoXML.newDocument(eventXML);
   ```

2. Create an XML element object and add it to the XML document object:

   ```
   IPLtoXML.newSubElement(eventXML, theEvent, "omniEvent");
   ```

3. Add the fields in the Event Container to the theEvent element object as attributes:

   ```
   IPLtoXML.addOrgNodeAttributes(theEvent, EventContainer);
   ```

4. Use the Java™ DSA to create a Java memory stack in a global variable called STACK:

   ```
   STACK=NewJavaObject("java.util.Stack", {});
   ```

   Note: `STACK` is required every time the `generateXML()` function is called and must be a global variable (defined outside of any function). `IPLtoXML` uses recursion and (v4.x) you cannot do recursive functions in the Impact parser without a separate memory stack.

5. Call the `GenerateXML()` function itself. Pass the function the XML document object, `eventXML`, and a results variable to put the output into:

   ```
   IPLtoXML.GenerateXML(eventXML, results);
   ```

The generated XML looks like this example:

```
<?xml version="1.0" encoding="UTF-8"?><omniEvent
Acknowledged="0" Agent="LinkMon" AlertGroup="Link"
AlertKey="" Class="3300" Customer="" EventId=""
EventReaderName="OMNIbusEventReader" ExpireTime="0"
FirstOccurrence="1188488450" Flash="0" Grade="0"
Identifier="link6LinkMon1Link" InternalLast="1188488841"
KeyField="8689" LastOccurrence="1188488841"
LocalNodeAlias="" LocalPriObj="" LocalRootObj=""
LocalSecObj="" Location="" Manager="Simnet Probe"
NmosCauseType="0" NmosObjInst="0" NmosSerial="" Node="link6"
NodeAlias="" OwnerGID="0" OwnerUID="65534" PhysicalCard=""
PhysicalPort="0" PhysicalSlot="0" Poll="0" ProcessReq="0"
ReceivedWhileImpactDown="1" RemoteNodeAlias=""
RemotePriObj="" RemoteRootObj="" RemoteSecObj=""
Serial="8689" ServerName="NCOMS" ServerSerial="8689"
Service="" Severity="0" StateChange="1188488841"
Summary="Link Up on port" SuppressEscl="0" Tally="8"
TaskList="0" Type="2" URL="" X733CorrNotif=""
X733EventType="0" X733ProbableCause="0" X733SpecificProb=""/
>
```

There is a single XML Element called `omniEvent`. Each of the fields in the original Event Container are XML attributes of `omniEvent`.

Example of generating WAAPI XML using IPL

This example shows how IPL to XML can be used to create an XML string that can be passed to the Netcool/Webtop API (WAAPI).

```
//create the XML Document object
IPLtoXML.newDocument(waapiXML);
//first element: the methodCall
IPLtoXML.newSubElement(waapiXML, myMethodCall,
"methodCall");
//a method element nested in the methodCall element
IPLtoXML.newSubElement(meMethodCall, method1, "method");
//attributes for method1
IPLtoXML.addAttribute(method1, "methodName",
"entity.createOrReplaceEntity");
//entitygroup element nested in method element
IPLtoXML.newSubElement(method1, entityGroup01,
"entitygroup");
//attributes for entitygroup
IPLtoXML.addAttribute(entityGroup01, "name",
"support");//this group was added to NGF
//entity element nested in entitygroup
IPLtoXML.newSubElement(entityGroup01, entity01, "entity");
//entity attributes
IPLtoXML.addAttribute(entity01, "name", "ncientity01");
IPLtoXML.addAttribute(entity01, "filter", "Severity > 4");
IPLtoXML.addAttribute(entity01, "metriclabel", "tom");
IPLtoXML.addAttribute(entity01, "metricshow", "Average");
IPLtoXML.addAttribute(entity01, "metricof", "Severity");
//entitylist nested in entitygroup
IPLtoXML.newSubElement(entityGroup01, entitylist01,
"entitylist");
//entitylist attributes
IPLtoXML.addAttribute(entitylist01, "name",
"ncientitylist01");
IPLtoXML.addAttribute(entitylist01, "list", "AllEvents");
IPLtoXML.addAttribute(entitylist01, "view", "basic");
IPLtoXML.addAttribute(entitylist01, "metriclabel", "tom");
```

The resulting XML string:

```
<?xml version="1.0" encoding="UTF-8"?><methodCall><method
methodName="entity.createOrReplaceEntity"><entitygroup
name="support "><entity filter="Severity &gt; 4"
metriclabel="tom" metricof="Severity" metricshow="Average"
name="ncientity01"/><entitylist list="AllEvents"
metriclabel="tom" name="ncientitylist01"
view="basic"/></entitygroup></method></methodCall>
```

This string can be passed to WAAPI via JRExec or Command/Response to create the Webtop entity.

Chapter 8. Connecting to WebSphere MQ and JMS DSA

Netcool/Impact 6.1.1 can communicate with WebSphere MQ 7 through the JMS data source.

For detailed information about JMS DSA, see the *Netcool/Impact DSA Reference Guide*.

There are two possible configuration options:

- Option 1: WebSphere MQ client and server and Netcool/Impact all on one machine.
- Option 2: WebSphere MQ client and Netcool/Impact on one machine and WebSphere MQ server on a separate machine.

For more information about WebSphere MQ, see http://www-01.ibm.com/software/integration/wmq/#.

Configuration option 1

How to configure the WebSphere MQ client and server and Netcool/Impact all on one machine.

Procedure

1. Configure WebSphere MQ server. For information see, http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp.
2. Copy the following four JAR files from `MQ_HOME/java/lib` to `IMPACT_HOME/dsalib`.
   - `com.ibm.mq.jmqi.jar`
   - `com.ibm.mqjms.jar`
   - `fscontext.jar`
   - `providerutil.jar`
3. Restart the Impact profile so that it picks up the new JAR files.
4. The Netcool/Impact JMS DSA will have the following parameters: For more information about configuring JMS DSA, see *JMS data source configuration properties* in the *DSA Reference Guide*.
   - **JNDI Factory initial:** `com.sun.jndi.fscontext.RefFSContextFactory`
   - **JNDI Provider URL:** `file:/<PathToBindingDirOnMQClient>` for example, `file:/C:/MQClientBindings`.
   - **JMS Connection Factory Name** as configured on the WebSphere MQ Server.
   - **JMS Destination Name** as configured on the WebSphere MQ Server.

Configuration option 2

How to connect the WebSphere MQ client and Netcool/Impact on one machine and WebSphere MQ server on a separate machine.

Procedure

1. Copy the following five JAR files from `MQ_HOME/java/lib` to `IMPACT_HOME/dsalib`.

- `com.ibm.mq.jmqi.jar`
- `com.ibm.mqjms.jar`
- `fscontext.jar`
- `providerutil.jar`
- `dhbcore.jar`

2. Restart the Impact profile so that it picks up the new JAR files.
3. You must configure the MQSeries® server to use a file system-based JNDI provider. WebSphere MQ then uses the local file system as a JNDI registry when it registers the JMS resources accessed by the DSA. For information about the JMS DSA, see the *DSA Reference Guide*.
4. You must use "client" mode in your connection factory on the WebSphere MQ server to ensure that the WebSphere MQ Client communicates through tcp with the WebSphere MQ Server.
5. Ensure that you have a listener that is configured on your preferred port on the WebSphere MQ server.
6. Create the queues and destinations that you need as specified by the WebSphere MQ documentation. For information see, http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp.
7. Copy the bindings directory, which is configured on the WebSphere MQ server, to the local file system of the WebSphere MQ client.
8. The Netcool/Impact JMS DSA will have the following parameters: For more information about configuring JMS DSA, see *JMS data source configuration properties* in the *DSA Reference Guide*.
   - **JNDI Factory initial:** `com.sun.jndi.fscontext.RefFSContextFactory`
   - **JNDI Provider URL:** `file:/<PathToBindingDirOnMQClient>` for example, `file:/C:/MQClientBindings`.
   - **JMS Connection Factory Name** as configured on the WebSphere MQ Server.
   - **JMS Destination Name** as configured on the WebSphere MQ Server.
9. If there are any configuration changes on the WebSphere MQ Server, you must repeat 7 to ensure that the WebSphere MQ Client picks up the configuration changes.

Appendix A. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. These are the major accessibility features you can use with *Netcool/Impact* when accessing it on the *IBM Personal Communications* terminal emulator:

- You can operate all features using the keyboard instead of the mouse.
- You can read text through interaction with assistive technology.
- You can use system settings for font, size, and color for all user interface controls.
- You can magnify what is displayed on your screen.

For more information about viewing PDFs from Adobe, go to the following web site: http://www.adobe.com/enterprise/accessibility/main.html

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. 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 give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

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 might 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

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

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

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

Linux is a 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.

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

Glossary

This glossary includes terms and definitions for Netcool/Impact.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology (opens in new window).

A

assignment operator
An operator that sets or resets a value to a variable. See also operator.

B

Boolean operator
A built-in function that specifies a logical operation of AND, OR or NOT when sets of operations are evaluated. The Boolean operators are &&, || and !. See also operator.

C

command execution manager
The service that manages remote command execution through a function in the policies.

command line manager
The service that manages the command-line interface.

Common Object Request Broker Architecture (CORBA)
An architecture and a specification for distributed object-oriented computing that separates client and server programs with a formal interface definition.

comparison operator
A built-in function that is used to compare two values. The comparison operators are ==, !=, <, >, <= and >=. See also operator.

control structure
A statement block in the policy that is executed when the terms of the control condition are satisfied.

CORBA
See Common Object Request Broker Architecture.

D

database (DB)
A collection of interrelated or independent data items that are stored together to serve one or more applications. See also database server.

database event listener
A service that listens for incoming messages from an SQL database data source and then triggers policies based on the incoming message data.

database event reader
An event reader that monitors an SQL database event source for new and modified events and triggers policies based on the event information. See also event reader.

database server
A software program that uses a database manager to provide database services to other software programs or computers. See also database.

data item
A unit of information to be processed.

data model
An abstract representation of the business data and metadata used in an installation. A data model contains data sources, data types, links, and event sources.

data source
A repository of data to which a federated server can connect and then retrieve data by using wrappers. A data source can contain relational databases, XML files, Excel spreadsheets, table-structured files, or other objects. In a federated system, data sources seem to be a single collective database.

data source adapter (DSA)
A component that allows the application to access data stored in an external source.

data type
An element of a data model that represents a set of data stored in a data source, for example, a table or view in a relational database.

DB See database.

DSA See data source adapter.

dynamic link
An element of a data model that represents a dynamic relationship between data items in data types. See also link.

E

email reader
A service that polls a Post Office Protocol (POP) mail server at intervals for incoming email and then triggers policies based on the incoming email data.

email sender
A service that sends email through an Simple Mail Transfer Protocol (SMTP) mail server.

event An occurrence of significance to a task or system. Events can include completion or failure of an operation, a user action, or the change in state of a process.

event processor
The service responsible for managing events through event reader, event

listener and email reader services. The event processor manages the incoming event queue and is responsible for sending queued events to the policy engine for processing.

event reader
A service that monitors an event source for new, updated, and deleted events, and triggers policies based on the event data. See also database event reader, standard event reader.

event source
A data source that stores and manages events.

exception
A condition or event that cannot be handled by a normal process.

F

field A set of one or more adjacent characters comprising a unit of data in an event or data item.

filter A device or program that separates data, signals, or material in accordance with specified criteria. See also LDAP filter, SQL filter.

function
Any instruction or set of related instructions that performs a specific operation. See also user-defined function.

G

generic event listener
A service that listens to an external data source for incoming events and triggers policies based on the event data.

graphical user interface (GUI)
A computer interface that presents a visual metaphor of a real-world scene, often of a desktop, by combining high-resolution graphics, pointing devices, menu bars and other menus, overlapping windows, icons and the object-action relationship. See also graphical user interface server.

graphical user interface server (GUI server)
A component that serves the web-based graphical user interface to web browsers through HTTP. See also graphical user interface.

GUI See graphical user interface.

GUI server
See graphical user interface server.

H

hibernating policy activator
A service that is responsible for waking hibernating policies.

I

instant messaging reader
A service that listens to external instant messaging servers for messages and triggers policies based on the incoming message data.

instant messaging service
A service that sends instant messages to instant messaging clients through a Jabber server.

IPL See Netcool/Impact policy language.

J

Java Database Connectivity (JDBC)
An industry standard for database-independent connectivity between the Java platform and a wide range of databases. The JDBC interface provides a call level interface for SQL-based and XQuery-based database access.

Java Message Service (JMS)
An application programming interface that provides Java language functions for handling messages.

JDBC See Java Database Connectivity.

JMS See Java Message Service.

JMS data source adapter (JMS DSA)
A data source adapter that sends and receives Java Message Service (JMS) messages.

JMS DSA
See JMS data source adapter.

K

key expression
An expression that specifies the value that one or more key fields in a data item must have in order to be retrieved in the IPL.

key field
A field that uniquely identifies a data item in a data type.

L

LDAP See Lightweight Directory Access Protocol.

LDAP data source adapter (LDAP DSA)
A data source adapter that reads directory data managed by an LDAP server. See also Lightweight Directory Access Protocol.

LDAP DSA
See LDAP data source adapter.

LDAP filter
An expression that is used to select data elements located at a point in an LDAP directory tree. See also filter.

Lightweight Directory Access Protocol (LDAP)
An open protocol that uses TCP/IP to provide access to directories that support an X.500 model and that does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). For example, LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory. See also LDAP data source adapter.

link An element of a data model that defines a relationship between data types and data items. See also dynamic link, static link.

M

mathematic operator
A built-in function that performs a mathematic operation on two values. The mathematic operators are +, -, *, / and %. See also operator.

mediator DSA
A type of data source adaptor that allows data provided by third-party systems, devices, and applications to be accessed.

N

Netcool/Impact policy language (IPL)
A programming language used to write policies.

O

operator
A built-in function that assigns a value to a variable, performs an operation on a value, or specifies how two values are to be compared in a policy. See also assignment operator, Boolean operator, comparison operator, mathematic operator, string operator.

P

policy A set of rules and actions that are required to be performed when certain events or status conditions occur in an environment.

policy activator
A service that runs a specified policy at intervals that the user defines.

policy engine
A feature that automates the tasks that the user specifies in the policy scripting language.

policy logger
The service that writes messages to the policy log.

POP See Post Office Protocol.

Post Office Protocol (POP)
A protocol that is used for exchanging network mail and accessing mailboxes.

precision event listener
A service that listens to the application for incoming messages and triggers policies based on the message data.

S

security manager
A component that is responsible for authenticating user logins.

self-monitoring service
A service that monitors memory and other status conditions and reports them as events.

server A component that is responsible for maintaining the data model, managing services, and running policies.

service
A runnable sub-component that the user controls from within the graphical user interface (GUI).

Simple Mail Transfer Protocol (SMTP)
An Internet application protocol for transferring mail among users of the Internet.

Simple Network Management Protocol (SNMP)
A set of protocols for monitoring systems and devices in complex networks. Information about managed devices is defined and stored in a Management Information Base (MIB). See also SNMP data source adapter.

SMTP See Simple Mail Transfer Protocol.

SNMP
See Simple Network Management Protocol.

SNMP data source adapter (SNMP DSA)
A data source adapter that allows management information stored by SNMP agents to be set and retrieved. It also allows SNMP traps and notifications to be sent to SNMP managers. See also Simple Network Management Protocol.

SNMP DSA
See SNMP data source adapter.

socket DSA
A data source adaptor that allows information to be exchanged with external applications using a socket server as the brokering agent.

SQL database DSA
A data source adaptor that retrieves information from relational databases and other data sources that provide a public interface through Java Database Connectivity (JDBC). SQL database DSAs also add, modify and delete information stored in these data sources.

SQL filter
An expression that is used to select rows in a database table. The syntax for the filter is similar to the contents of an SQL WHERE clause. See also filter.

standard event reader
A service that monitors a database for new, updated, and deleted events and triggers policies based on the event data. See also event reader.

static link
An element of a data model that defines a static relationship between data items in internal data types. See also link.

string concatenation
In REXX, an operation that joins two characters or strings in the order specified, forming one string whose length is equal to the sum of the lengths of the two characters or strings.

string operator
A built-in function that performs an operation on two strings. See also operator.

U

user-defined function
A custom function that can be used to organize code in a policy. See also function.

V

variable
A representation of a changeable value.

W

web services DSA
A data source adapter that exchanges information with external applications that provide a web services application programming interface (API).

X

XML data source adapter
A data source adapter that reads XML data from strings and files, and reads XML data from web servers over HTTP.

Index

Printed in USA

SC27-4854-00