Business Service Manager

Administrator's Guide

SC27-8782-03

Note

Before using this information and the product it supports, read the information in Appendix A, “Notices,” on page 371.

Edition notice

This edition applies to IBM® Tivoli Business Service Manager Version 6 Release 2.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. About this publication

This guide contains information how to operate, maintain, and configure the product.

Audience

This publication is for administrators and system programmers who need to use, install, maintain, or configure TBSM.

Publications

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

TBSM library

The following documents are available in the TBSM library:

- *Installation Guide*, GI11-8054-10

  Provides information about installing the product.

- *Quick Start*, GI11-8055-04

  Provides overview information about TBSM.

- *Exploring IBM Tivoli Business Service Manager*, GI11-8056-10

  Provides an overview of the product features.

- *Administrator's Guide*, SC23-6040-10

  Provides information about managing and configuring TBSM.

- *Service Configuration Guide*, SC23-6041-10

  Provides information on how to use the features of the product console.

- *Customization Guide*, SC23-6042-10

  Provides information on how to customize select features of the product.

- *Troubleshooting Guide*, GI11-8057-10

  Provides information about resolving common problems with the product.

- *Release Notes*,

  Provides latest information about the product discovered late in the test cycle that cannot be incorporated into the other publications.

Prerequisite publications

To use the information in this publication effectively, you must have some prerequisite knowledge, which you can obtain from the publications listed here.

These publications are available on the Tivoli Netcool/OMNIbus Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSSHTQ_8.1.0/com.ibm.netcool_OMNIbus.doc_8.1.0/omnibus/wip/common/reference/omn_ref_PDFbooks.html

- IBM Tivoli Netcool/OMNIbus *User Guide*

Provides an overview of Netcool/OMNIbus components, as well as a description of the operator tasks related to event management using the desktop tools. TBSM uses Netcool/OMNIbus as its event manager.

- IBM Tivoli Netcool/OMNIBUS *Administration Guide*

  Provides information about how to perform administrative tasks using the Netcool/OMNIbus Administrator GUI, command line tools, and process control. It also contains descriptions and examples of ObjectServer SQL syntax and automations.

- IBM Tivoli Netcool/OMNIBUS *Probe and Gateway Guide*

  Provides information contains introductory and reference information about probes and gateways, including probe rules file syntax and gateway commands. For more information about specific probes and gateways, refer to the documentation available for each probe and gateway.

- IBM Tivoli Netcool/OMNIBUS *Probe for Tivoli EIF*

  Provides reference information about the optional Probe for Tivoli EIF that is included with TBSM.

Related publications

The following documents also provide useful information and are included in the TBSM Information Center.

These publications are available on the IBM Tivoli Business Service Manager Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSSPFK

- IBM Tivoli Netcool/Impact *Administration Guide*

  Provides information about installing, configuring and running Netcool/Impact and its related software components. TBSM uses Netcool/Impact policies to parse events and other data.

- IBM Tivoli Netcool/Impact *User Interface Guide*

  Provides information about using the Netcool/Impact user interface.

- IBM Tivoli Netcool/Impact *Policy Reference Guide*

  Provides reference information about the Netcool/Impact Policy Language (IPL). It contains complete information about policy language syntax, data types, operators and functions.

- IBM Tivoli Netcool/Impact *Solutions Guide*

  Provides information about implementing Netcool/Impact in your environment.

- IBM Tivoli Netcool/Impact DSA *Reference Guide*

  Provides reference information about Netcool/Impact data source adaptors (DSA).

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

The format of the publications is PDF, HTML, or both.

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Knowledge Center at https://www.ibm.com/support/knowledgecenter/SSSPFK

Note: If you print PDF documents on 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.

Ordering publications

According to e-Business strategy, IBM Publications Center no longer supports ordering publications. The publications are made available in electronic format to be viewed or downloaded free of charge.

For documentation related to TBSM, go to https://www.ibm.com/support/knowledgecenter/en/SSSPFK.

Accessibility

This guide contains information how to operate, maintain, and configure the product.

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. In this release, the TBSM console does not meet all accessibility requirements.

Tivoli technical training

For Tivoli technical training information, refer to the IBM developerWorks Website at https://www.ibm.com/developerworks.

Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online
Access the IBM Software Support site at https://www.ibm.com/support/home/.

IBM Support Assistant
The IBM Support Assistant is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The Support Assistant provides quick access to support-related information and serviceability tools for problem determination. To install the Support Assistant software, go to https://www-01.ibm.com/software/support/isa/.

Troubleshooting Guide
For more information about resolving problems, see the problem determination information for this product.

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

Chapter 2. Introduction to IBM Tivoli Business Service Manager

This information can help you understand IBM Tivoli Business Service Manager (TBSM), including its business value and key technologies.

TBSM delivers the real-time information that you need in order to respond to alerts effectively and in line with business requirements, and optionally to meet service-level agreements (SLAs).

The TBSM tools enable you to build a service model that you integrate with IBM Tivoli Netcool/OMNIbus alerts or optionally with data from an SQL data source. TBSM includes optional components that let you access data from other IBM Tivoli applications such as IBM Tivoli Monitoring, and IBM Tivoli Application Dependency Discovery Manager. TBSM processes the external data based on the service model data you created in the TBSM database and returns a new or an updated TBSM service event to Netcool/OMNIbus.

The TBSM console provides a graphical user interface (GUI) that allows you to logically link services and business requirements within the service model. The service model provides an operator with a view of how, second by second, an enterprise is performing at any given moment in time or how the enterprise has performed over a given time period.

Chapter 3. What's new in TBSM Version 6.2.0

This documentation applies to TBSM version 6.2.0 and all fix packs, unless indicated otherwise. TBSM Version 6.2.0 contains support for Jazz for Service Management as the new Dashboard for TBSM and Netcool/Impact 7.1.0.

Jazz for Service Management

TBSM has been updated to use the IBM Dashboard Application Services Hub as its new UI. In order to use these new features, you also need to have Jazz® for Service Management and the IBM Dashboard Application Services Hub installed as part of your environment.

For more information see *Administration Guide > TBSM and IBM Dashboard Application Services Hub*.

IBM Dashboard Application Services Hub

The IBM Dashboard Application Services Hub provides user interface and dashboard services in Jazz for Service management. This new self-service dashboard capability enables you to combine a variety of visual widgets such as gauges, tables, charts, lists or topology views into custom dashboards using a guided work flow. These dashboards can also include management data from sources such as:

- Service status and metrics from TBSM
- Third-party data from Netcool/Impact
- Performance metrics from IBM Tivoli Monitoring

Mobile Support: The self-service dashboard enable you to view business dashboards on mobile devices including tablets and phones. This enables access to both information technology and business data anytime / anywhere and gives you the ability to support your customers more effectively.

New functionality in TBSM 6.2

The following functionality is new in TBSM 6.2:

- TBSM widgets and DASH sidget can communicate with each other without needing to create a separate connection.
- Since TBSM Portlet is installed on JazzSM, all the CURI TBSM datasets are available locally without the need for a remote connection.
- The dashboard provides the following types of widgets that can be used by custom pages to build responsive and interactive pages:
  - Status gauge
  - Area chart
  - Bar chart
  - Line chart
  - Topology

Features modified in, or removed, from TBSM

The following features have been either modified or removed in TBSM 6.2:

- Tivoli Integrated Portal (TIP) and its components (Graphs, Custom Canvas, Service viewer Applet code, Birt charts) have been removed.
- TBSM portlets and pages throughout DASH have been ported to DASH.
- TBSM has been upgraded to support the latest version of Java (JDK 1.8).
- Packaging method changed from Install Anywhere to Installation Manager.

- Obsolete features and functionality (Custom Canvas, JViews, and Birt charts) have been removed.
- Prerequisite stack (for example Netcool and Netcool Impact) moved from embedded to stand alone.
- Impact runs in the Liberty server and DASH server runs in WAS 8.5 Service Navigation.
- Custom canvases have been removed
- Birt charts have been removed but COGNOS reports are available as is.

Service Editor

The following changes have been made to the Service Editor:

- Tabs have been removed and the Edit view is now the default in the Server Editor portal. The View tab has been removed as it was based on Applet and now you can use DASH Topology widgets to create a similar topological view of services.
- Clicking on the service now opens the corresponding Service view in the Service Editor
- The Invalidate button and the button related to ESDA have been hidden by default and can be enabled if you wish by enabling the flag defined in property file. To enable it on the screen, go to JazzSM directory where TBSM Deployable Artifact has been installed: ./opt/IBM/JazzSM/profile/installedApps/ JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_ sla.props and change the key 'impact.esda.enable=false' value to true and restart the JazzSM Server.
- The Policy Editor communication protocol has been changed from SOAP to Rest API.
- SSO between JazzSM and Impact is now a prerequisite for the TBSM Policy editor
- Run functionality in the Policy Editor has been disabled.

Service Availability

The following changes have been made to Service Availabilty:

- Events/Node graphs have been implemented using Rave based on the policy.
- Service Viewer has been removed because it was Applet based.

Change to TBSM Admin pages, to be within JazzSM

The following changes have been made to the TBSM admin pages to be within JazzSM:

- TBSM admin pages have been put under the catalog listing of Business Service Management.
- You can create custom pages using TBSM widgets by choosing from the Business Service Management catalog.

Changes to the use of TBSM widgets

The following changes have been made to the use of TBSM widgets:

- The pre-existing TBSM widget remains functional as it was in the earlier version.
- TBSM widgets can communicate with DASH widgets. Clicking on any row on the Service Tree can render respective topology.

What's new in Netcool/Impact

TBSM Data Server integrates with Netcool/Impact version 7 release 1.0.13. The new version includes these enhancements:

New visualization: The new visualization include Operator View customization enhancements and UI Services provided by Jazz for Service Management. These will enable clients to link their own data accessed through Impact's proven data access methods with visual widgets such as gauges, tables, or lists to create dashboards.

Linked data integration: Netcool/Impact can also use the Jazz for Service Management registry services that follow the Open Services for Lifecycle Collaboration (OSLC) standards.

Service Level Objective (SLO) Reporting: Enables you to establish and report on service level objectives based on their own measures (for example, incidents, tickets, and availability).

Consumability: Continued improvements to enhance the user experience, including MWM cluster replication and e-mail reader enhancements.

Enhanced Web Services Integrations and Wizards: Enhances and simplifies access to web services data sources.

Chapter 4. Technical overview of TBSM

This section contains topics about the product architecture and the main software components.

TBSM architecture

This section describes the basic architecture of the IBM Tivoli Business Service Manager (TBSM).

TBSM archictecture shows the basic architecture for TBSM. The TBSM Data server analyzes IBM Netcool/OMNIbus ObjectServer events or SQL data for matches against the incoming-status rules you configured for your service models. If the matching data changes the service status, the status of the TBSM service model changes accordingly. When a services status changes, TBSM sends corresponding service events back to the ObjectServer.

You can also use data from an external database or an ObjectServer to drive custom views and charts. The Discovery Library Toolkit lets you create TBSM service objects using data from Discovery Library Adaptor (DLA) books or from the IBM Tivoli Application Dependency Discovery Manager.

The TBSM users and group permissions are managed by the Dashboard Application Service Hub, which can authenticate users internally, or use data from an external source such as an ObjectServer or LDAP server.

TBSM components

This topic describes the components included on the product DVD.

The applications IBM Tivoli Netcool/Omnibus, Netcool/Omnibus WebGUI, IBM Tivoli Netcool/Impact and JazzSM/Dashboard Application Service Hub are not included on the TBSM product DVD. These applications must be installed as pre-requisite products on a host that is accessible by the TBSM server.

TBSM has the following components included in the Installer package or DVD:

- TBSM Dashboard server
- TBSM Data server
- TBSM DBConfig
- Discovery Library Toolkit

Note: In TBSM 6.2, Netcool/OMNIbus, Netcool/OMNIbus WebGUI and Netcool/Impact products are not included in the installer. These products are prerequisites and must be installed separately before installing TBSM.

Tivoli Netcool/OMNIbus

TBSM monitors the Tivoli Netcool/OMNIbus ObjectServer for incoming events. The ObjectServer collects events from probes, monitors, and other applications such as IBM Tivoli Monitoring. You use TBSM to create service models that respond to the data received in the incoming events. For example, the incoming event data can change the status of a service or start the tracking of a potential SLA violation. In short, if you can set up a probe or other application to forward data to the TBSM ObjectServer, you can use that data to build and monitor your service models. The TBSM installation package includes Netcool/OMNIbus. If you want to use the Discovery Library toolkit, or the IBM Tivoli Event Integration Facility (EIF) probe you need version 7.1 or higher.

Note that Tivoli Netcool/OMNIbus V8.1.0.x should be installed as a pre-requisite before installing TBSM.

For more information see: IBM Tivoli Netcool/OMNIbus documentation

Netcool/OMNIBus Web GUI

The Web GUI is the browser console for Netcool/OMNIbus and TBSM uses Web GUI components to display events related to service models. The Active Event List (AEL) and Service Details portlet in TBSM are Web GUI components, and are required to be installed as part of the pre-requisite products before installing TBSM on the server where JazzSM is installed.

For more information see: **IBM Tivoli Netcool/OMNIbus Considerations** in the *IBM Tivoli Business Service Manager Installation Guide* .

TBSM Dashboard server

The TBSM Dashboard server manages the TBSM console display. You can have multiple dashboard servers for a single data server. The dashboard server enhances the scalability, performance, and availability of TBSM.

The TBSM Dashboard server communicates with the TBSM Data server to support the creation and visualization of service models through connected TBSM consoles. As console users view portions of the service model, the dashboard server will acquire and maintain status of services from the data server.

TBSM Data server

The TBSM Data server monitors the ObjectServer and external databases for data that affect the status of the services you configured in the TBSM console or with the RAD shell command line tool. The server calculates the status of these services by applying rules to the external data. Your service models and the rules are stored in the TBSM database.

TBSM DB2 database

The TBSM DB2® database stores all the information on the service models you created in the TBSM console. This data includes rules that determine how your service model changes in relation to data in external data sources. This database also includes tables for the metrics and markers used in the Time Window Analyzer and demo data. A Metric History database, which has a default name of TBSMHIST, is also included to store the historical metric data,

Discovery Library Toolkit

The Discovery Library Toolkit enables TBSM to discovery resources and to automatically build service models from Discovery Library data sources. These sources include:IBM Tivoli Application Dependency Discovery Manager , Discovery Library books conforming to the common data model, Discovery Library books containing objects for an alternate namespace, the Discovery Library toolkit API, or auto-pop objects.

Data discovered through the toolkit can be enriched through notifications sent to Impact. This enriched data can then be used in the automatic building of the service model.

Netcool/Impact

Netcool/Impact is the automation, correlation, and integration engine for the IBM Tivoli Netcool suite of software products. You can use Netcool/Impact to automate event management tasks, to correlate event information with other information in your environment, and to integrate Netcool products with a wide variety of third party systems and applications.

TBSM 6.2 does not include the Netcool/Impact as part of the Data Server. You need to install Impact 7.1.0.13 prior to installing TBSM.

As a consequence of this integration, you can now take advantage of these Netcool/Impact capabilities:

- You can use Netcool/Impact services and policies to acquire, enrich, and pass data to TBSM to use for service status determination or visualization.

- TBSM uses the same policy functions and policy language as Netcool/Impact. Javascript is supported as a policy language in addition to IPL (Impact Policy Language).
- Event enrichment is supported as an out-of-box function. Impact policies enrich events before TBSM reads these same events for status determination and propagation.
- The Impact User Interface is installed separately as prerequisite along with Impact Server and SSO configuration between Dashboard Application Service Hub and Impact. This provides a common user interface for administration of both TBSM and Impact policies and services.
- The Data server package includes a name server that enables you to access Netcool/Impact server clusters.

For more information about Netcool/Impact, see the Tivoli Netcool/Impact publications at: Netcool/Impact Documentation.

Integrated applications

This section is an overview of the optional external applications you can integrate with TBSM.

The following applications either forward data to TBSM, or receive data from TBSM:

- Using the IBM Tivoli EIF probe, you can forward data from IBM Tivoli Monitoring version 6 release 1 and above, Tivoli Enterprise Console® version 3 release 9 or later, IBM Tivoli Netview version 3 release 7 or later.
- IBM Tivoli Application Dependency Discovery Manager version 7 release 1.2 or later
- IBM Tivoli Change and Configuration Management Database (CCMDB) version 7 releases 1 and 1.1
- Discovery Library Adapters including those from the following products:
  - IBM Tivoli Monitoring (6.2.3 or higher is recommended)
  - IBM Tivoli Composite Application Manager for SOA
  - IBM Tivoli Composite Application Manager for WebSphere®
  - IBM Tivoli Composite Application Manager for Transaction Tracking
  - IBM Tivoli Network Manager
  - IBM Tivoli NetView® for z/OS®
  - IBM Tivoli Storage Productivity Center version 4, release 1.1

You can launch to or from the following applications from TBSM:

- Tivoli Monitoring 6.2 with fix pack 1 or later
- Tivoli Application Dependency Discovery Manager 7.1 or later
- CCMDB version 7.1 or later
- Netcool/OMNIbus Web GUI component.
- IBM Tivoli Network Manager IP Edition version 3 release 8
- IBM Tivoli Composite Application Manager for Transactions version 7 release 1.0.2
- IBM Tivoli TotalStorage Productivity Center (TPC)

Note: For launch support, the supported product versions may be more restrictive than those specified for data exchange above.

Jazz for Service Management

TBSM uses Dashboard Application Service Hub as its UI component in 6.2 instead of Tivoli Integrated Portal. In order to use these new features, you also need to have Jazz for Service Management and the IBM Dashboard Application Services Hub installed as part of your environment.

Jazz for Service Management employs a new deployment pattern and mechanism that helps you integrate shared components such as your User Interface, Linked Data Registry, Reporting, Security, and Administrative Services. This new mechanism helps you speed up delivery cycles for clients and simplify deployments.

For more information see *Administration Guide > TBSM and IBM Dashboard Application Services Hub*.

IBM Dashboard Application Services Hub

The IBM Dashboard Application Services Hub provides user interface and dashboard services in Jazz for Service management. This new self-service dashboard capability enables you to combine a variety of visual widgets such as gauges, tables, charts, lists or topology views into custom dashboards using a guided work flow. These dashboards can also include management data from sources such as:

- Service status and metrics from TBSM
- Third-party data from Netcool/Impact
- Performance metrics from IBM Tivoli Monitoring

Mobile Support: The self-service dashboard enable you to view business dashboards on mobile devices including tablets and phones. This enables access to both information technology and business data anytime / anywhere and gives you the ability to support your customers more effectively.

Tivoli Event Integration Facility (EIF) probe

You can set up the optional IBM Tivoli Event Integration Facility (EIF) probe to access the event data from applications such as IBM Tivoli Monitoring, Tivoli Enterprise Console, and Tivoli Netview. The probe forwards the event data to the TBSM Netcool/OMNIbus ObjectServer. You can use TBSM to create service models based on the event data from the Event Pump for z/OS, Tivoli Monitoring (and Tivoli Monitoring agents), Tivoli Enterprise Console, and Tivoli NetView.

IBM Tivoli Netcool/Impact

Netcool/Impact is the automation, correlation, and integration engine for the IBM Tivoli Netcool® suite of software products. You can use Netcool/Impact to automate event management tasks, to correlate event information with other information in your environment, and to integrate Netcool products with a wide variety of third party systems and applications.

You can configure Netcool/Impact to forward events to the Netcool/OMNIbus ObjectServer monitored by TBSM and use those events to update your service model. Netcool/Impact is designed for Netcool administrators who want to enhance, customize, and extend the capabilities of the Netcool suite. For more information, see the Netcool/Impact publications.

Change and Configuration Management Database

TBSM can launch into a Change and Configuration Management Database (CCMDB) associated with a Tivoli Application Dependency Discovery Manager. If you create service models with the TBSM Discovery Library integration, these services can contain data about a Tivoli Application Dependency Discovery Manager server. If a service contains data about a Tivoli Application Dependency Discovery Manager server, you can launch the Tivoli Application Dependency Discovery Manager and CCMDB consoles from the TBSM console. Likewise, you can also launch TBSM from the Tivoli Application Dependency Discovery Manager console.

Operating system variables and paths

On both the Data server and the Dashboard server a script is provided that allows you to set environment variables for quick access to the TBSM directory structure. If you do not set the variables, you can substitute directories with full path names when you run commands.

You must run the script that applies to the servers that you installed. If you installed both servers on the same system, you must run both scripts.

The locations of these setup scripts on UNIX systems are as follows:

- *installdirectory*/tbsm/bin/setupTBSMData.sh for the Data server
- *installdirectory*/tbsmdash/bin/setupTBSMDash.sh for the Dashboard server

where *installdirectory* is the directory in which you installed the server. The default directory is /opt/IBM/tivoli.

The syntax used to run the UNIX scripts is:

```
. installdirectory/tbsm/bin/setupTBSMData.sh
```

The locations of these setup scripts on Windows systems are as follows:

- *installdirectory*\tbsm\bin\setupTBSMData.bat for the Data server
- *installdirectory*\tbsmdash\bin\setupTBSMDash.bat for the Dashboard server

where *installdirectory* is the directory in which you installed the server. The default directory is C:\Program Files\IBM\tivoli.

The setupTBSMDash script sets the following variables:

```
TBSM_HOME=/opt/IBM/tivoli/tbsmdash
JAZZ_HOME=/opt/IBM/JazzSM
TBSM_DASHBOARD_SERVER_HOME=
   /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war
DASHBOARD_PROFILE=JazzSMProfile
JAVA_HOME=/opt/IBM/tivoli/tbsmdash/_jvm/jre
```

The setupTBSMData script sets the following variables:

```
TBSM_DATA_SERVER_HOME=/opt/IBM/tivoli/impact/wlp/usr/servers/TBSM/apps/TBSM.ear/
TBSM_HOME=/opt/IBM/tivoli/tbsm
TBSM_LIBS=/opt/IBM/tivoli/impact/lib3p
HOSTNAME=<hostname of the installed server>
HTTPSPORT=<https port of the Impact GUI server>
HTTPPORT=<http port of the Impact GUI server>
```

Variables used in TBSM Publications

For many of the commands and paths specified in this publication, both the UNIX and Windows equivalents are provided. However, in instances where only the UNIX convention has been specified, follow these directions for Windows systems.

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.

Java support

This topic describes the Java™ runtime Environment (JRE) plug-in versions that are required for the IBM Tivoli Business Service Manager user interface in a web browser.

Supported Java runtime versions: The most up-to-date information about supported hardware, software, browsers and operating systems is provided by the IBM Software Product Compatibility Reports at:

https://www.ibm.com/software/reports/compatibility/clarity/prereqsForProduct.html

1. In the **Full or partial product name:** field, type `Business Service` and click the search button.
2. From the **Search Results**, select **Tivoli Business Service Manager.**
3. From the **Version** field, select **6.2.0**.
4. From **Mandatory capabilities:**, select **Java**.
5. Click **Submit**.

Note: The Java Runtime Environment that is being used should be updated to the most recent fix level.

Important: These web browser settings are required:

- JavaScript is enabled in the browser.
- Set your browser to allow pop-up windows. If you block pop-up windows, you will disable features of TBSM that require pop-up windows.
- Set your browser to accept third-party cookies.

Chapter 5. Configuring TBSM with external LDAP user registry considerations

TBSM Data Server

If you selected the file-based repository option during installation, you can manually configure TBSM to use an LDAP repository.

If you want to use an LDAP user repository, you must configure it after the installation has completed

To configure TBSM to authenticate to an LDAP repository, select **File Registry** as your user repository during an advanced installation. You can then configure TBSM to use an LDAP authentication source after the installation has completed.

When you install TBSM, the users `tbsmadmin` and `tbsmuser` and the groups `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsers`, are created in the target user repository. By default this is a IBM Tivoli Netcool/OMNIbus repository. After installation, you can configure TBSM to use an external user repository.

If you want to use LDAP and have not already installed TBSM, complete an advanced installation of TBSM and select the file-registry option as the repository. After the installation completes, follow the steps to configure the Data Server for LDAP, see Configuring the TBSM Data server for LDAP.

If TBSM has already been installed with a Netcool/OMNIbus user repository, you can complete this step to configure the Data Server for LDAP as an additional repository. However, you must ensure that the same users and groups do not exist in both repositories.

TBSM DASH Server

You can configure the Dashboard Application Service Hub Server to communicate with an external LDAP repository.

If you want to use LDAP and have not already installed TBSM, complete an installation of TBSM and select the file-registry option as the repository. After the installation completes, follow the steps to configure the Dashboard server for LDAP, see Configuring the TBSM Dashboard server for LDAP.

You can complete this step to configure the Dashboard server for LDAP as an additional repository. However, you must ensure that the same users and groups do not exist in both repositories.

To create or manage users in the portal that are defined in your LDAP repository, in the WebSphere® Application Server administrative console specify the supported entity types.

For details about creating or managing LDAP users in the portal, see Managing LDAP users in the console.

When you install TBSM, the users `tbsmadmin` and `tbsmuser` and the groups `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsers` are created in the target user repository. By default this is a IBM Tivoli Netcool/OMNIbus repository. After installation, you can configure TBSM to use an external user repository. However, if you select a repository after installation that contains the same user ID or groups as those already in the default repository, you must remove any duplicate groups from the default repository before you configure the external repository. If you do not, you might not be able to log into TBSM using the default administrative user ID.

To redefine the TBSM user and groups in the default repository and add them to the external user repository, see “Configuring TBSM to use an external user repository after installation” on page 46

Configuring the Data server

This section contains information about several configuration tasks: configuring the consistency checker, configuring the discriminator field, configuring the timeout property for large service models, and configuring the urgent services threshold.

Configuring the consistency checker

The consistency checker periodically checks Overall status events that are in Netcool/OMNIbus against the TBSM model to make sure that they are correct. It assumes that the TBSM memory model is the definitive source from which to determine status. It then adds or deletes overall status events from ObjectServer if it finds inconsistencies with the model. The consistency checker performs a similar function for the `service_deps` table in Netcool/OMNIbus which is a mapping of all events in Netcool/OMNIbus to the services they affect. This table is used by WebGUI to display the matching events in the **Service Details** panel. It adds and deletes entries in the `service_deps` table if it finds inconsistencies with the memory model. You can configure settings for the consistency checker by editing the `TBSM_consistency.props` file. The editable properties include logging options, the frequency with which the consistency checker runs, and the event threshold.

Before you begin

For systems that have a thousands of rows being created or updated in the service_deps table in a short period of time, this can lead to memory cache issues.

Caching works efficiently if there is a small number of updates to the service deps table, but if there will be 100s of updates per minute, caching does not provide any usefulness. You should disable caching, for both data and query. The exact threshold of when this should occur is not known, the administrator will have to use their judgement to assess if caching will be needed or not.

The cache settings for service_deps can be modified in **System Configuration** > **Event Automation** > **Data Model**.

You will notice caching errors in the netcool.log and consistency checker log (if enabled) if the service_deps cache is having problems with the volume of updates it has to handle.

Procedure

1. Using a text editor, open the `TBSM_consistency.props` file. This file is in the `/opt/IBM/tivoli/impact/etc` directory.
2. Modify the property values.
3. Save the modified file.
4. Restart the Data server.

   Note: In version 6.1.1, the default interval has been changed from 5 to 11 minutes. If you upgraded from version 6.1 and have a different value already defined in the `TBSM_consistency.props` file, your value is retained.

Changing the existing host name for TBSM 6.2 Data Server and Dashboard Server

You can re-configure an existing TBSM 6.2 installation to a different host name. This topic describes the changes required to the TBSM 6.2 Dashboard Server and Data Server.

On the Dashboard Server

Replace the old Dashboard Server host name in the files listed below with the new Dashboard host name.

Replace the old TBSM Data Server host name in the files listed below with the new TBSM Data Server host name if the TBSM Data Server host name gets changed.

```
<JAZZ_INSTALL_DIR>/var/admin_task_bundles/dash.response
<JAZZ_INSTALL_DIR>/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_server.props
<JAZZ_INSTALL_DIR>/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_sla.props
<JAZZ_INSTALL_DIR>/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/nameserver.props
<JAZZ_INSTALL_DIR>/profile/config/cells/JazzSMNode01Cell/nodes/JazzSMNode01/serverindex.xml
<JAZZ_INSTALL_DIR>/profile/config/cells/JazzSMNode01Cell/security.xml
<JAZZ_INSTALL_DIR>/profile/installedFilters/wlm/server1/target.xml
```

Key stores and certificates

The old dashboard host name may also exist in the key store (`key.p12`) and trust store (`trust.p12`) files. For details about creating certificates to replace existing ones: see https://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/tsec_sslreplacecell.html

To replace default certificates in the Dashboard Server, you must create a new self-signed root certificate in the root keystore (`NodeDefaultRootStore`) and replace the old root certificate with the new one.

On the Data Server

The TBSM Data Server includes the Impact component. For details of changing the TBSM Data Server host name, see https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.17/com.ibm.netcoolimpact.doc/admin/changing_hostname_of_installation.html

Note: The **`nci_changehost`** command generates certificates in the key and trust stores (with the new host name).

Then, change the old TBSM Data Server host name to the new TBSM Data Server host name in the following files (and also any specific files that you may have on your server):

```
<INSTALL_DIR>/tivoli/impact/etc/server.props
<INSTALL_DIR>/tivoli/impact/etc/TBSM_RelatedEventsDatasource.ds
<INSTALL_DIR>/tivoli/impact/etc/TBSM_seasonalReportDatasource.ds
<INSTALL_DIR>/tivoli/impact/etc/TBSM_emailsender.props
<INSTALL_DIR>/tivoli/impact/etc/TBSM_NOIReportDatasource.ds
<INSTALL_DIR>/tivoli/impact/etc/TBSM_sla.props
<INSTALL_DIR>/tivoli/impact/wlp/usr/servers/ImpactUI/jvm.options
<INSTALL_DIR>/tivoli/impact/wlp/usr/servers/TBSM/jvm.options
```

For the files in the `<INSTALL_DIR>/tivoli/impact/etc/` directory, the changes will need to be put under `svn` control.

For example:

1. Check out the `server.props` file:

   ```
   $IMPACT_HOME/bin/nci_version_control TBSM co etc/server.props impactadmin
   ```

2. Update the `server.props` file with the new host name.
3. Check in the `server.props` file:

   ```
   $IMPACT_HOME/bin/nci_version_control TBSM ci etc/server.props "change of
   host name" impactadmin
   ```

Note: The Impact version control script, `$IMPACT_HOME/bin/nci_version_control`, stores the copy of changes to the server properties files in version control. For details about `$IMPACT_HOME/bin/nci_version_control`, see https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.17/com.ibm.netcoolimpact.doc/admin/imag_version_version_control_script_c.html.

Also, under the `<INSTALL_DIR>/tivoli/tbsm` directory, replace the old TBSM Data Server host name in the following files with the new TBSM Data Server host name:

```
tbsm/XMLtoolkit/bin/install/xmltoolkitsvc.properties
tbsm/XMLtoolkit/bin/xmltoolkitsvc.properties
tbsm/bin/setupTBSMData.sh
tbsm/etc/rad_dbconf
```

After making these changes, restart all servers: JazzSM Server, TBSM Data Server, and TBSM Impact UI Server.

Configuring the discriminator field

You can configure the discriminator field by modifying the TBSM_sla.props file.

About this task

The discriminator field is an ObjectServer event field that IBM Tivoli Business Service Manager checks before other event identification filters. The discriminator field speeds up the system, because TBSM does not have to test every filter in the rule against every event. By default, the Class field is used.

To configure the discriminator field, complete the following steps:

Procedure

1. Using a text editor, open the TBSM_sla.props file. This file is located in the /opt/IBM/tivoli/impact/etc directory.
2. Add the following property and value:

   ```
   impact.sla.eventdiscriminatorfield=fieldname
   ```

   where *fieldname* is the name of the ObjectServer event field.

   For example, if you want to set the discriminator field to the Poll field, add the following property and value pair:

   ```
   impact.sla.eventdiscriminatorfield=Poll
   ```

3. Save the modified file.

   Note: If the new discriminator field you choose is not an integer, the conversions performed in the ObjectServer must be selected in the Netcool/OMNIbus SQL interface. This is because the Netcool/OMNIbus console only allows the creation of conversions for integer fields. For more information, see “Adding values to the Conversions table” on page 21.
4. Edit the EventDiscriminatorField.txt file:
   a) Open the /opt/IBM/tivoli/impact/etc/EventDiscriminatorField.txt file in a text editor.
   b) To change the field name to which the discriminator field points, edit the Class value to match the required field. For example, if you wanted to change the discriminator field to the Poll field, edit the files as shown:

      ```
      FieldName,EventDiscriminatorID
      "Class","TheEventDiscriminator"

      to

      FieldName,EventDiscriminatorID
      "Poll","TheEventDiscriminator"
      ```

   c) Restart the TBSM Data and Dashboard servers.

Configuring the timeout property for large service models

If you have ESDA rules that create large service models with more than 500 service instances, increase the timeout period for ESDA policies. By default, the timeout period is set to 120 seconds.

About this task

To change the timeout property for the ESDA policy, complete the following steps:

Procedure

1. Using a text editor, open the TBSM_sla.props file. This file is located in the $IMPACT_HOME/etc directory.

2. Add the following property and value to set the ESDA policy timeout value to 600 seconds (10 minutes):

```
impact.sla.maxesdapolicytimesecs=600
```

3. Save the modified file.
4. Stop and restart the Data server.

Deleting a maintenance schedule

You can delete a maintenance schedule by editing the `scheduleTime.xml` file. This file is stored in the TBSM database. This removes the maintenance schedule from the selection field in the IBM Tivoli Business Service Manager console.

Procedure

1. Copy the `scheduleTime.xml` file from the TBSM database to your temporary file directory by typing the following at a command prompt:

```
%TBSM_TOOLKIT_BASE%\bin\getArtifact.bat -name scheduleTime.xml
-category maintenance -directory <temp>
```

UNIX

```
$TBSM_TOOLKIT_BASE/bin/getArtifact.sh -name scheduleTime.xml
-category maintenance -directory <temp>
```

   where `scheduleTime.xml` is the file that you want to edit and `<temp>` is the location on your system that you use as a temporary file system directory.
2. Using a text editor, open the `scheduleTime.xml` file.
3. Locate the applicable `<timeWindowDefinition>` element. The name attribute is set to the name of the maintenance schedule.
   For example, if you want to delete the `PE_Maintenance-1` schedule, locate the following element:

```
<timeWindowDefinition name = "PE_Maintenance-1">
    <timeWindowCombo>
       <timeWindow>Thursday 7:00 PM - 8:50 PM</timeWindow>
       <timeWindow>23 Oct 2003 7:30 PM - 24 Oct 2003 10:01 PM</timeWindow>
    </timeWindowCombo>
 </timeWindowDefinition>
```

4. Delete the `<timeWindowDefinition>` element.
5. Save the modified file.
6. To copy the updated view definition XML file into the TBSM database, at a command prompt type:

```
%TBSM_HOME%\XMLtoolkit\bin\putArtifact.bat -name <temp>/scheduleTime.xml
-category maintenance
```

UNIX

```
$TBSM_HOME/XMLtoolkit/bin/putArtifact.sh -name <temp>/scheduleTime.xml
-category maintenance
```

7. Stop and restart the TBSM Data server.

Adding values to the Conversions table

To add field values to the ObjectServer Conversions table, use the SQL interactive interface utility. You might want to add conversions for the **Event Discriminator** field and the **Output Severity Threshold**

fields for incoming status rules. This example shows how to add the following **Manager** field values to the Conversions table:

- `Pinger`
- `TrapReceiver`

Change values using the SQL interactive interface on UNIX systems

To run the SQL interactive interface on a UNIX system, issue the following commands:

```
$OMNIHOME/bin/nco_sql -server NCOMS -user root -password ''
INSERT INTO conversions (KeyField,Colname,Value,Conversion) values
('Manager0','Manager',0,'Pinger');
INSERT INTO conversions (KeyField,Colname,Value,Conversion) values
('Manager1','Manager',1,'TrapReceiver');
```

Change values using the SQL interactive interface on Windows systems

To run the SQL interactive interface on a Windows system, issue the following commands:

```
%OMNIHOME%\bin\redist\isql.exe -S NCOMS -U root -password ''
INSERT INTO conversions (KeyField,Colname,Value,Conversion) values
('Manager0','Manager',0,'Pinger');
INSERT INTO conversions (KeyField,Colname,Value,Conversion) values
('Manager1','Manager',1,'TrapReceiver');
```

Delete checker

If an event is deleted from the TBSM default ObjectServer, TBSM does not immediately update the status of services affected by the deleted event. However, there is a maximum period the deleted event checker waits until running again (5 minutes by default). TBSM periodically checks the ObjectServer for the raw events of each service with the delete checker utility. If the events have been deleted, TBSM clears the affected service.

Default delete checker time period

By default, the delete checker runs periodically according to the following formula:
20 times (how long it took last time to check all events for deletes) plus 10 seconds.

Note: Deleting uncleared events is still *strongly* discouraged as the delete checker does not run in real time. This time difference between the delete checker time period and the actual time of the status change makes SLA time tracking inaccurate.

Example

If the delete checker took 1 minute to check all events, it waits 5 minutes before checking again. This is why it is still unwise to rely on the delete checker for clearing services. If an SLA was being tracked, you could have as much as five extra minutes of downtime.

Changing the delete checker period

You can override the period of the delete checker using the following property in the `$IMPACT_HOME/etc/TBSM_sla.props` file:

```
impact.timetowaitbetweendeletechecks=seconds
```

where *seconds* is the number of seconds to wait between delete checks.

The property **impact.timetowaitbetweenedeletechecks** sets an absolute number of seconds to check for deleted events after the previous process has finished. Alternatively you can set the property **impact.maximumtimetowaitbetweendeletechecks**, which sets the maximum amount of time to wait before checking for deleted events.

Setting SLA timing for multiple hosts

Synchronizing the host system clocks when you run TBSM and the ObjectServer on different hosts.

About this task

SLA tracking is time-sensitive. You must synchronize the host system clocks when you run TBSM and the ObjectServer on different hosts. In failover configurations, you must also synchronize the clock for your backup TBSM server. If you do not synchronize your servers, incorrect results, ranging from negative downtime to over 100% uptime, might appear.

Synchronize with Network Time Protocol

Using the Network Time Protocol to synchronize your server.

About this task

You can synchronize a host computer with a second computer on UNIX by issuing the command:

```
rdate -s hostname
```

where *hostname* is the name of the host system that supplies the synchronized time to your host system.

Configuring TBSM for Event Enrichment

When you want to enrich ObjectServer events with an Impact policy, before they are processed by TBSM, you must make changes to the configuration of the Impact Event Reader and the TBSM ObjectServer Event Reader.

If you do not apply these changes, the TBSM ObjectServer Event Reader might read the event before it has been enriched by Impact. TBSM does not reread that event because the severity does not change as a result of the enrichment by Impact, and TBSM is configured to read updates only when the severity changes. Make the changes to the configuration outlined in this topic to ensure that ObjectServer events are enriched by Impact before being processed by TBSM.

Note: The properties in this topic can also be configured using the **Event Enrichment** screen of the **Service Navigator**. It is preferable that these properties are changed in the user interface. The manual configuration procedure is provided for reference purposes. For more information, see the Impact Event Enrichment section of the *TBSM Service Configuration Guide*.

Purpose

The configuration changes outlined in this topic ensure that TBSM does not read events until Impact has enriched them. To avoid this issue, configuration changes are necessary to ensure that Impact updates a field in OMNIbus called `RAD_SeenByImpact` after Impact is finished with the event. An additional TBSM restriction added to the TBSMOMNIbusEventReader ensures that events are not read unless `RAD_SeenByImpact = 1`.

Parameters

Update the following parameters for the Impact Event Reader that you want to configure located:

Windows:

```
C:\Program Files\IBM\tivoli\impact\etc
```

UNIX:

```
/opt/IBM/tivoli/impact/etc
```

Note: If you choose to configure the Impact Event Reader called `OMNIbusEventReader`, which is the name of an unconfigured ObjectServer Event Reader provided with the Impact installation for TBSM, the props file is located in the directory outlined and named `TBSM_omnibuseventreader.props`.

impact.<servicename>.objectserver.seenbyimpactupdate= RAD_SeenByImpact=1
This property causes the `ReturnEvent` function to automatically update the`RAD_SeenByImpact` field to 1 in an updated event. This enables TBSM to process the event immediately after the enrichment is completed. The value of this property must be set to `RAD_SeenByImpact=1`.

impact.<servicename>.objectserver.forceupdateforread
This property causes Impact to update the `RAD_SeenByImpact` field to 1 for all events that do not match the `eventbroker` filter. This is necessary to enable TBSM to read events for which Impact does not otherwise set `RAD_SeenByImpact =1`. The value of this property must be set to `true`.

impact.<servicename>.objectserver.forceupdateforpersist
This property causes Impact to update the `RAD_SeenByImpact=1` field to 1 for every event that it reads. The value of this property must be set to `true`.

In addition to the Impact Event Reader properties, make the following change to the `TBSM_tbsmomnibuseventreader.props` file located:

Windows:

```
C:\Program Files\impact\tivoli\tbsm\etc
```

UNIX:

```
/opt/IBM/tivoli/impact/etc
```

impact.tbsmomnibuseventreader.objectserver.seenbyimpactselect= RAD_SeenByImpact=1
This property restricts the TBSM Event Reader so that it cannot process any events until `RAD_SeenByImpact=1`. The value of this property must be set to `RAD_SeenByImpact=1`.

Note: The OMNIbus schema for TBSM has an automation that changes the value of the `RAD_SeenByImpact` field to 0 for any update to the event that does not come from Impact. This configuration forces TBSM to wait until Impact has read an event and decided to either process or not process it before TBSM reads the event.

Additional properties

In addition to the configuration properties outlined above, there are two additional properties that might require changes:

impact.<servicename>.objectserver.seenbyimpactselect= RAD_SeenByImpact=0
This property is set to `RAD_SeenByImpact=0` by default. This prevents Impact from rereading the events after TBSM has updated them. It does this by restricting the Event Reader to read only events where `RAD_SeenByImpact = 0`

Note: This is only effective if the other properties are configured as described in this topic.

RAD_SeenByTBSM
This field was introduced to prevent TBSM from reading its own updates if the restriction clause for the OMNIbus Event Reader is set to allow TBSM to read events that did not change the severity. This field is set to 1 by TBSM after it has processed the event. The `rad_updateseenbyfields` automation in omnibus resets the field to 0 if any other client updates the event. The filter of the TBSM OMNIbus Event Reader reads events only where `RAD_SeenByTBSM = 0`. This prevents TBSM from rereading its own updates of the raw events.

Authorizing additional users for TBSM Data server services

To authorize additional users for the protected Time Window Analyzer web services running on the TBSM Data server, run the `wsadmin` command outlined in this topic. This command replaces all users, therefore, you must include all users in the command when completing the configuration update.

```
wsadmin    -profileName TBSMProfile  -username <tbsmadmin>
-password  <tbsmadmin user's password>

$AdminApp  edit twamarker   {-MapRolesToUsers  { { "bsmAdministrator"
No  No "<complete list of userids>"  ""} } }

$AdminConfig save
exit
```

Example

```
$AdminApp  edit twamarker   {-MapRolesToUsers  { { "bsmAdministrator"
No  No "tbsmadmin|jack|jill|bob"  ""} } }
```

Keep Instance Invalid after Exception

Keep Instance Invalid after Exception

If this property 'Keep Instance Invalid after Exception' is set to true, then if exception occurs getting children, it will stay invalid and recalculate the next time.

Default Keep Instance Invalid after Exception

By default the 'Keep Instance Invalid after Exception' property is set to true.

Changing the Keep Instance Invalid after Exception

You can override the 'Keep Instance Invalid after Exception' using the following property in the `IMPACT_HOME/etc/TBSM_sla.props file:`

`impact.sla.keepinstanceinvalidafterexception=false`

The property **`impact.sla.keepinstanceinvalidafterexception`** sets to false, and if exception occurs getting children, it will not stay invalidate.

Configuring TBSM with external LDAP user registry considerations

TBSM Data Server

If you selected the file-based repository option during installation, you can manually configure TBSM to use an LDAP repository.

If you want to use an LDAP user repository, you must configure it after the installation has completed

To configure TBSM to authenticate to an LDAP repository, select **File Registry** as your user repository during an advanced installation. You can then configure TBSM to use an LDAP authentication source after the installation has completed.

When you install TBSM, the users `tbsmadmin` and `tbsmuser` and the groups `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsers`, are created in the target user repository. By default this is a IBM Tivoli Netcool/OMNIbus repository. After installation, you can configure TBSM to use an external user repository.

If you want to use LDAP and have not already installed TBSM, complete an advanced installation of TBSM and select the file-registry option as the repository. After the installation completes, follow the steps to configure the Data Server for LDAP, see Configuring the TBSM Data server for LDAP.

If TBSM has already been installed with a Netcool/OMNIbus user repository, you can complete this step to configure the Data Server for LDAP as an additional repository. However, you must ensure that the same users and groups do not exist in both repositories.

TBSM DASH Server

You can configure the Dashboard Application Service Hub Server to communicate with an external LDAP repository.

If you want to use LDAP and have not already installed TBSM, complete an installation of TBSM and select the file-registry option as the repository. After the installation completes, follow the steps to configure the Dashboard server for LDAP, see Configuring the TBSM Dashboard server for LDAP.

You can complete this step to configure the Dashboard server for LDAP as an additional repository. However, you must ensure that the same users and groups do not exist in both repositories.

To create or manage users in the portal that are defined in your LDAP repository, in the WebSphere® Application Server administrative console specify the supported entity types.

For details about creating or managing LDAP users in the portal, see Managing LDAP users in the console.

When you install TBSM, the users `tbsmadmin` and `tbsmuser` and the groups `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsers` are created in the target user repository. By default this is a IBM Tivoli Netcool/OMNIbus repository. After installation, you can configure TBSM to use an external user repository. However, if you select a repository after installation that contains the same user ID or groups as those already in the default repository, you must remove any duplicate groups from the default repository before you configure the external repository. If you do not, you might not be able to log into TBSM using the default administrative user ID.

To redefine the TBSM user and groups in the default repository and add them to the external user repository, see “Configuring TBSM to use an external user repository after installation” on page 46

Configuring TBSM with external LDAP user registry considerations

TBSM Data Server

If you selected the file-based repository option during installation, you can manually configure TBSM to use an LDAP repository.

If you want to use an LDAP user repository, you must configure it after the installation has completed

To configure TBSM to authenticate to an LDAP repository, select **File Registry** as your user repository during an advanced installation. You can then configure TBSM to use an LDAP authentication source after the installation has completed.

When you install TBSM, the users `tbsmadmin` and `tbsmuser` and the groups `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsers`, are created in the target user repository. By default this is a IBM Tivoli Netcool/OMNIbus repository. After installation, you can configure TBSM to use an external user repository.

If you want to use LDAP and have not already installed TBSM, complete an advanced installation of TBSM and select the file-registry option as the repository. After the installation completes, follow the steps to configure the Data Server for LDAP, see Configuring the TBSM Data server for LDAP.

If TBSM has already been installed with a Netcool/OMNIbus user repository, you can complete this step to configure the Data Server for LDAP as an additional repository. However, you must ensure that the same users and groups do not exist in both repositories.

TBSM DASH Server

You can configure the Dashboard Application Service Hub Server to communicate with an external LDAP repository.

If you want to use LDAP and have not already installed TBSM, complete an installation of TBSM and select the file-registry option as the repository. After the installation completes, follow the steps to configure the Dashboard server for LDAP, see Configuring the TBSM Dashboard server for LDAP.

You can complete this step to configure the Dashboard server for LDAP as an additional repository. However, you must ensure that the same users and groups do not exist in both repositories.

To create or manage users in the portal that are defined in your LDAP repository, in the WebSphere® Application Server administrative console specify the supported entity types.

For details about creating or managing LDAP users in the portal, see Managing LDAP users in the console.

When you install TBSM, the users `tbsmadmin` and `tbsmuser` and the groups `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsers` are created in the target user repository. By default this is a IBM Tivoli Netcool/OMNIbus repository. After installation, you can configure TBSM to use an external user repository. However, if you select a repository after installation that contains the same user ID or groups as those already in the default repository, you must remove any duplicate groups from the default repository before you configure the external repository. If you do not, you might not be able to log into TBSM using the default administrative user ID.

To redefine the TBSM user and groups in the default repository and add them to the external user repository, see "Configuring TBSM to use an external user repository after installation" on page 46

Modifying the session timeout value for Dashboard Application Service Hub applications

By default, the session inactivity timeout value for Dashboard Application Service Hub applications is 30 minutes. You can modify this value for all Dashboard Application Service Hub applications, including TBSM.

About this task

To modify the session inactivity timeout value for Dashboard Application Service Hub applications, perform the following steps:

Procedure

1. Stop the TBSM Dashboard server.
2. Open the following file:

   ```
   /opt/IBM/JazzSM/profile/config/cells/JazzSMNode01Cell/applications/isclite.ear/
   deployments/isclite/deployment.xml
   ```

3. Locate the line that starts with `<tuningParams`.

   Near the end of this line, there is a string that is like the following example:

   ```
   invalidationTimeout="30"
   ```

4. Change the value `30` to the number of minutes that you want to use for the session inactivity timeout.
5. Save the change that you made and start the Dashboard server.

Related information

Configuring the Discovery Library Toolkit

You must configure some TBSM components and settings to ensure that the Discovery Library Toolkit can integrate with other components and products.

About this task

To perform basic configuration tasks associated with the Discovery Library Toolkit:

Procedure

1. If you need to reset the user IDs or passwords for TBSM, the TBSM database, Tivoli Application Dependence Discovery Manager, or Tivoli Application Dependency Discovery Manager database the for use with the Discovery Library Toolkit, run the **setxmlaccess** command.

   The toolkit installation program sets these values based on input that was provided when the toolkit was being installed. You can issue the **setxmlaccess** command to change these values or to add values that were not required during the installation. Access online help for the command by issuing **setxmlaccess -?**.
2. If you are using Tivoli Application Dependency Discovery Manager as the data source, You must copy the Tivoli Application Dependency Discovery Manager client jar files from the Tivoli Application Dependency Discovery Manager to the TBSM server by doing one of the following:
   - If you are running Tivoli Application Dependency Discovery Manager 7.1.2, copy the file `/dist/clientlib/taddm-api-client.jar` file to the TBSM `$TBSM_HOME/XMLtoolkit/sdk/clientlib` directory.
   - If you are running Tivoli Application Dependency Discovery Manager 7.2, copy the file `taddm-api-client.jar, platform-model.jar` and `oal-topomgr.jar` files from the `/dist/clientlib/taddm-api-client.jar` directory to the TBSM `$TBSM_HOME/XMLtoolkit/sdk/clientlib` directory.
   - If using TADDM 7.2.1 and above, the jar files are copied automatically on the first occasion that the Discovery Library Toolkit is started. The list of jar files that are copied is:

     ```
     taddm-api-client.jar,
     platform-model.jar
     oal-topomgr.jar.
     ```

     After the initial copy, the Discovery Library Toolkit is shut down. Restart the Discover Library Toolkit to complete the installation of the jar files and to start the toolkit process.

registryupdate

The `registryupdate` command allows you to manage the Discovery Library Toolkit registry table.

Purpose

The Discovery Library Toolkit registry table is setup during installation. The registry table generally contains two rows, one for the primary Discovery Library Toolkit and one for the alternate Discovery Library Toolkit. The table entries are not order dependent, row 1 is not necessarily the primary. The table only contains one row if an alternate Discovery Library Toolkit is not defined. The `registryupdate` command sets the `Name` field in the selected database row based on the `DL_Toolkit_Instance_ID` property from the `xmltoolkitsvc.properties` file and the `Primary` value based on the `DL_Preferred_Primary` property. The `xmltoolkitsvc.properties` file is located in the `%TBSM_HOME%\XMLToolkit\bin` directory on Windows systems and `$TBSM_HOME /XMLToolkit/bin` on UNIX systems. If you change the `DL_Preferred_Primary` or `DL_Alternate_TBSM_Hostname` properties in the `xmltoolkitsvc.properties` file, run the `registryupdate` command to update the registry entries in the database.

Syntax

```
registryupdate -U dbUser -P dbPassword [-s ID] [-v]
```

Parameters

The parameters for the `registryupdate` command are:

where:

-U
The database user ID. If you do not provided a value for this parameter, you are prompted to provide a value.

-P
The database user password. If you do not provided a value for this parameter, you are prompted to provide a value.

-s
Updates the Discovery Library Toolkit registry table based on the integer value that you enter. This integer indicates the registry entry that you want to update. Valid values are 1 and 2.

-v
Display the contents of the Discovery Library Toolkit registry table.

Configuring TBSM console

This section contains information about modifying the appearance of the graphical user interface (GUI).

Adding custom icons

You can add custom icons for service templates. The icons are displayed in the IBM Tivoli Business Service Manager console.

Before you begin

You need the following two versions of the icon:

- a Graphics Interchange Format (GIF) file, and
- a Scalable Vector Graphic (SVG) file

Note: If you add any custom icons, you must add them to each Dashboard server separately.

Procedure

1. Ensure that the graphics files use the following naming format:

File type	Name
GIF	*filename*_svg.gif
SVG	*filename*.svg

   where *filename* is a text string.
2. Copy the GIF file to the `/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/icons/` directory.
3. Copy the SVG file to the `/opt/IBM/netcool/gui/omnibus_webgui/integration/tipInstallableApps/content/scripts/dt/idx/themes/dlBlue/idx/widget/images/` directory.

Configuring the EIF Probe

About this task

The basic configuration of the probe should be appropriate for most installations. One configuration property that might need to be adjusted is the ***Inactivity*** value. This value (in seconds) controls how long after starting, the probe will continue to run without receiving any events. The default value of 600 seconds may be too low for some implementations. A value of 0 allows the probe to run until it is manually stopped. This value is adjusted by modifying the tivoli_eif.props file and adding to the bottom (in the **Add your settings here** section) **Inactivity : 0**

Modifying the tivoli_eif.props file

About this task

The `tivoli_eif.props` file is self-documented in that it contains each supported property name along with its default value. All of these properties are contained in the section that begins with the following lines:

```
##############################################################################
#
# Property Name                     Default
#
# Generic Properties
```

If not changed, the default value for any property is the value that appears in the ***Default*** column.

It is strongly recommended that any property value that is modified from its default value be added to this section at the bottom of the file:

```
##############################################################################
#
# Add your settings here
#
##############################################################################
```

Doing this ensures that the original default value and the changed value are available for debugging and problem support.

Configuring users, groups, and roles

In the left navigation pane, the pages listed under **Users and Groups** allow you to create users, groups, and roles and to assign the authority to view, modify, and administer specific TBSM settings. This information describes the roles that are specific to TBSM.

Before you begin

Important: Do not use multibyte characters or special characters for any field value or parameter in the Dashboard Application Service Hub user interface, including ?, ", !, $, %, /, \, &, and #.

To access the **Roles** page, you must be assigned administrator roles in TBSM. To access the **User Roles** and the **Group Roles** pages, you must be assigned iscadmins roles in TBSM.

You can perform all of the tasks discussed in this section using Dashboard Application Service Hub command line. For more information, see the *Command Reference* section of the Dashboard Application Service Hub documentation contained in the *Reference* chapter of this guide.

TBSM users, user groups, and roles

IBM Tivoli Business Service Manager includes predefined users and user groups. Each of these groups assigns certain roles to members of the group.

Users and user groups

Two users, `tbsmadmin` and `tbsmuser`, are created by TBSM Dashboard Server when it is installed. The default password for these users is the same as the password of the `impactadmin` user which was created as part of Netcool/Impact installation. Use these users or `tbsmadmin` to log in to TBSM for the first time. If you are installing the TBSM Data Server before installing the TBSM Dashboard Server, you need to create the `tbsmadmin` user with the same password as the `impactadmin` user password in the external User repository or the ObjectServer, whichever is configured as the TBSM user repository.

Important: The default login `tbsmadmin` is not defined in external repositories, such as LDAP or Netcool/OMNIbus, even though this default login is created during installation when an external repository is specified.

Users with a blank password cannot log in to the TBSM Dashboard server. The default password for the OMNIbus ObjectServer root user is `null`; therefore, if you want to log in as `root` and you have ObjectServer authority, you must specify a non-null password for the `root` user in the ObjectServer. If you need to change the ObjectServer password, use the procedure *Configuring TBSM > Changing the TBSM configuration > Changing the Netcool/OMNIbus ObjectServer password or user ID*.

The TBSM users and groups are created in the repository that you select during the installation for an advanced installation or in the Netcool/OMNIbus repository during a simple installation. For more information, see the TBSM *Installation Guide*.

You can assign users to the following predefined groups to define their level of access and authority in TBSM:

tbsmAdmins
Use this group for administrators. The roles assigned to this user group enable the group members to view and modify all TBSM objects in the graphical user interface (GUI).

tbsmUsers
Use this group for users who need to view all templates that are defined in the model.

tbsmViewAllServicesUsers
Use this group for users who only need to view all services that are defined in the model.

tbsmReadOnly
Use this group for users who you want to have only read-only access. By default, roles are assigned to this group that provide view-only capabilities. Users assigned to this group are restricted to the **Service Availability** page. These users cannot access administrative tasks.

By default, the `tbsmadmin` is assigned to the `tbsmAdmins` group and also to the WebGUI `Netcool_OMNIbus_Admin` group. The `tbsmuser` is assigned to the `tbsmUsers` group and also to the WebGUI `Netcool_OMNIbus_Users` group. If you perform an advanced installation and select the file registry as the file repository, the `tbsmadmin` user is also added to the `tbsmAdmins`, WebGUI `Netcool_OMNIbus_Admin`, and `Netcool_OMNIbus_Users` groups.

You can also manage user and group permissions for each service or service template in the TBSM GUI. For more information, see the TBSM *Service Configuration Guide*.

For information about changing the default service and template privileges for users and groups, see *Modifying the default service and template privileges* in the *Administrator's Guide*.

User roles

You can assign any of the following roles to users or groups. These roles specify the authority that users or groups have to view, modify, or administer TBSM settings.

Table 1. TBSM user roles

Role	Authority assigned to user or group
tbsmAdminUser	Access to both the **Service Availability** and **Service Administration** pages in TBSM
tbsmSLAChartViewVisible	Assigned automatically by TBSM to the necessary users and groups. This role does not display in the list roles for users and groups. Do not assign this role manually.
tbsmViewRawEvents	View ObjectServer event lists. **Note:** This role is no longer used.
tbsmTemplateAdmin	Add, edit, delete, or view templates.

Table 1. TBSM user roles (continued)	
Role	**Authority assigned to user or group**
tbsmServiceAdmin	Add, edit, delete, or view services.
tbsmCreateTemplate	Add, edit, or view templates.
tbsmEditTemplate	Edit or view templates.
tbsmViewTemplate	View templates.
tbsmCreateService	Add or view services.
tbsmEditService	Edit or view services.
tbsmViewService	View services.
tbsmDataSourceAdmin	Add, edit, delete, or view data sources.
tbsmCreateDataSource	Add, edit, or view data sources.
tbsmEditDataSource	Edit or view data sources.
tbsmViewDataSource	View data sources.
tbsmDataFetcherAdmin	Add, edit, delete, or view data fetchers.
tbsmCreateDataFetcher	Add, edit, or view data fetchers.
tbsmEditDataFetcher	Edit or view data fetchers.
tbsmViewDataFetcher	View data fetchers.
tbsmChartAdmin	Add, edit, delete, or view charts.
tbsmCreateChart	Add, edit, or view charts.
tbsmEditChart	Edit or view charts.
tbsmViewChart	View charts.
tbsmViewDefinitionAdmin	Edit or delete view definitions. **Note:** The default view definitions are read-only and cannot be edited or deleted.
tbsmReadOnlyUser	Access to the **Service Availability** page only. This role is assigned by default to the `tbsmReadOnly` group; users assigned to that group automatically have this role.

Setting up users, user roles, and groups

In the Dashboard Application Service Hub console, click **Users and Groups** in the left navigation panel to add users, assign users to groups, and give users and groups authority to view, modify, and administer TBSM settings. For general information about setting up users, groups, and roles, see the console help.

Procedure

1. Log in to the Dashboard Application Service Hub as a user with administration privileges.
2. Create the user by completing the following steps:
   a) From the left navigation pane, click **Settings > WebSphere Application Server Console > Users and Groups** and then click **Manage Users**.
   b) Click **Create**.
   c) Complete the required fields, which are indicated with an asterisk (*) and then click **Create**.

The user ID is added to the user registry and is used as the login account name.

3. (Optional) Create a group:
   a) Click **Manage Groups**.
   b) Click **Create**.
   c) Type the group name in the **Group name** field and click **Create**.
4. Assign the user to a group:
   a) Click **Manage Groups**.
      The **Search for Groups** pane is displayed.
   b) Specify the name of the group to which you want to add this user and click **Search**.
      To see all existing groups, accept the default value (*).
   c) Click the name of the group to which you want to add this user.
   d) Select the **Members** tab.
   e) Click **Add Users**.
   f) Complete the information about the user that you want to add to the group and click **Search**.
   g) Select the user that you want to add to the group and click **Add**.
5. Assign the roles required by the new user to the user group. You can assign roles to an individual user, but if you have multiple users with the same roles, it is better to assign the roles to a group.
   See "Managing roles for groups" on page 36.
6. To change the user privileges for a service, edit the service.

   Tip:

   If the user has a service or template role that is assigned either directly or indirectly through membership in a group, you do not need to assign that role to the user by editing the service or template.

   a) In the left navigation pane of the Dashboard Application Service Hub console, select **BSM** and then select **Service Configuration**.
      The Service Configuration information displays.
   b) In the **Service Navigation** portlet, select **Services** from the drop-down menu.
   c) Click the service that you want to edit. The service is opened in the **Service Editor**.
   d) In the **Service Editor** pane, click **Edit Service** and then select the **Security** tab.
   e) Assign group or user roles for a particular service instance.

   You can also assign group or user roles when you edit service templates. See the TBSM *Service Configuration Guide* for more information about assigning roles for a given service or service template.

Creating roles

Portal users are granted access to resources based on the role to which they have been assigned. All roles that are created in the portal have a resource type of `Custom`. This procedure describes creating a role for testing purposes. After completing these steps, you can remove or edit this role for production use.

Procedure

1. Click **Settings > Roles** in the navigation.
   A list of all roles in the portal is displayed.
2. Click **New**.
   The properties panel for the new role is displayed.
3. Enter a descriptive name for the role.
4. Optional: Expand the **Users and Groups** section.

Use this section to associate a role with one or more users and groups. The method to add users and groups is similar, so this topic describes adding users only. To associate a user with a role, follow these steps:

a) In the **Users** panel, click **Add**.

A new page is displayed that allows you to search for and select users to be added to the role.

b) Provide search filters in the relevant fields, select the maximum number of results that you want returned and click **Search** to return a list of users that match your criteria.

Tip: If you leave the search filter fields blank, the system returns all users (up to a limit of 1000).

c) From the returned results, select the users that you want to associate with the role and click **Add**.

The previous page is displayed listing the selected users in the **Users** panel.

5. Expand the **Access to Views** section.

Use this section to grant access to one or more custom views for users who are assigned to the new role. If you have already created a custom view, follow these steps.

a) Click **Add**.

A list of available views is displayed.

b) Select one or more views and click **OK**.

c) To make sure the role has access to all of the pages within the view, click **Grant to All**.

6. Expand the **Access to Pages** section.

A list of pages that the role can access is displayed. However, this list is empty if you did not add a view and grant access to all of the pages within the view.

7. Optional: Click **Add** to grant access to additional pages.
8. For each page that is listed, verify that the **Access Level** is set correctly.
9. Click **Save** to save your changes and return to **Roles**.

Results

The new role is created with access to the views, users and groups, and pages that you indicated. To grant access to the portlets on those pages you must edit the portlets.

Editing roles

Portal users are granted access to resources based on the role to which they have been assigned. If you have sufficient authorization in the portal, you can change the name of custom roles. For all roles, you can change access to views and pages and set the access level to pages.

About this task

Procedure

1. In the navigation pane, click **Settings > Roles**.

A list of all roles in the portal is displayed.

2. Click the name of the role that you want to edit.

The properties panel for the role is displayed. If this is a custom role, the only field you can edit is **Role Name**. For all other resource types, you cannot edit any of the role properties.

3. Optional: Expand the **Users and Groups** section.

Use this section to associate a role with one or more users and groups. The method to add users and groups is similar, so this topic describes adding users only. To associate a user with a role, follow these steps:

a) In the **Users** panel, click **Add**.

A new page is displayed that allows you to search for and select users to be added to the role.

b) Provide search filters in the relevant fields, select the maximum number of results that you want returned and click **Search** to return a list of users that match your criteria.

Tip: If you leave the search filter fields blank, the system returns all users (up to a limit of 1000).

c) From the returned results, select the users that you want to associate with the role and click **Add**.

The previous page is displayed listing the selected users in the **Users** panel.

4. Expand the **Access to Views** section.

Use this section to grant access to one or more custom views for users who are assigned to the new role. If you have already created a custom view, follow these steps.

a) Click **Add**.

A list of available views is displayed.

b) Select one or more views and click **OK**.

c) To make sure the role has access to all of the pages within the view, click **Grant to All**.

5. Expand the **Access to Pages** section.

A list of pages that the role can access is displayed. However, this list is empty if you did not add a view and grant access to all of the pages within the view.

6. Optional: Click **Add** to grant access to additional pages.
7. For each page that is listed, verify that the **Access Level** is set correctly.
8. Click **OK**.

Results

Your changes are saved and you are returned to the **Roles** page.

What to do next

For any pages that you added for the role, you should ensure that the role also has access to the portlets on the page..

Deleting custom roles

You can delete only roles with the resource type of `Custom`. These are roles created using the portal.

About this task

Attention: Before deleting a role, consider whether any users are actively using the role and any impacts this might have on services. If necessary, notify users in advance of any plans for changes that could affect their work.

Follow these steps to delete a custom role.

Procedure

1. Click **Settings > Roles** in the navigation pane.

The **Roles** page is displayed with the list of roles in the portal.

2. Select the custom role that you want to delete. You can select more than one custom role.
3. Click **Delete**.

A message is displayed at the top prompting you to confirm the deletion.

4. Click **OK**.

Results

The custom role is removed from the list.

Managing roles for users

Administrators can search for users and manage their roles in the User Roles page.

About this task

To search for users and manage their roles:

Procedure

1. In the navigation pane, click **Settings > User Roles**.

   The **User Roles** page is displayed.
2. In the search fields provided, you can enter search criteria by given name, surname, user ID, and e-mail address.

   If you do not have exact details for a particular item, all of the search fields support using an asterisk (*) as a wildcard character. For example, to return all user records with a given name that starts with "Mich", enter `mich*` in the **First name** field.

   Tip: You can leave the search fields blank to return all user records.
3. From the **Number of results to display** list, select the number of records that you want returned and click **Search**.

   Restriction: Returned records are displayed one page only. If more records are available than the setting you chose from the list, only a partial list is returned. To display all records you need to search again after selecting a larger number from the **Number of results to display** list.

   A list of records that match your search criteria are listed in the grid.
4. Select a user from the **User ID** column.

   A list of available roles for the selected user is displayed on a new page. Those roles that are currently associated with the selected user are checked.
5. Modify the roles associated with the user as required, that is, check the roles that you want associated with the user and clear those that you do not.
6. Click **Save** to commit your changes, or **Reset** to reset the form to its initial state.

   Once you click **Save**, the **User Roles** page is displayed. The entry for the user in the **Roles** column is updated to reflect your changes.

What to do next

You can select another user from the search results and update their role settings, enter new search criteria to manage other user records, or close the **User Roles** page.

Managing roles for groups

Administrators can search for groups and manage their roles in the Group Roles page.

About this task

To search for user groups and manage their roles:

Procedure

1. In the navigation pane, click **Settings > Group Roles**.

   The **Group Roles** page is displayed.
2. In the search fields provided, you can enter search criteria by group ID and description.

   If you do not have exact details for a particular item, both search fields support using an asterisk (*) as a wildcard character. For example, to return all group records with a group ID that starts with "tes", enter `tes*` in the **Group ID** field.

   Tip: You can leave the search fields blank to return all records.

3. From the **Number of results to display** list, select the number of records that you want returned and click **Search**.

   Restriction: Returned records are displayed one page only. If more records are available than the setting you chose from the list, only a partial list is returned. To display all records you need to search again after selecting a larger number from the **Number of results to display** list.

   A list of records that match your search criteria are listed in the grid.
4. Select a group from the **Group Name** column.

   A list of available roles for the selected group is displayed on a new page. Those roles that are currently associated with the selected group are checked.
5. Modify the roles associated with the group as required, that is, check the roles that you want associated with the group and clear those that you do not.
6. Click **Save** to commit your changes, or **Reset** to reset the form to its initial state.

   Once you click **Save**, the **Group Roles** page is displayed. The entry for the group in the **Roles** column is updated to reflect your changes.

What to do next

You can select another group from the search results and update its role settings, enter new search criteria to manage other group records, or close the **Group Roles** page.

Role properties

This panel is used to edit the general properties of a role. The properties panel is displayed when you click one of the roles to edit it. This panel is also displayed when you create a new role.

To access this panel, click **Settings > Roles** in the navigation pane. Then click the name of one of the roles that are listed, or click **New** to create a new role.

Role name
Enter a descriptive name for the role. This name should be informative enough to indicate the actions and resources that are available to the users in this role. The role name must be unique within the portal. The characters in a role name are restricted to letters, digits, blank spaces, and underscores. The role name cannot start with a digit.

Type of role
Displays either `Custom`, `System`, or `Core`. This field cannot be changed.

Users and Groups
This section shows two table displaying the users and groups that have been assigned to this role. The following options are unique for these tables.

Users
This table lists the users with access to the role. The following column headings are unique to this table:

User ID
Displays the unique system ID associated with the user.

First Name
Displays the first name associated with the user ID.

Last Name
Displays the last name associated with the user ID.

Groups
This table lists the user groups with access to the role. The following column headings are unique to this table:

Group Name
Displays the name associated with this group.

Unique Name
Displays the unique system ID associated with the group.

Access to Views
This section provides a table displaying views that have been assigned to this role. The following options are unique for the views table.

View Name
Indicates the name of the view. You can sort the list of names by clicking the column heading.

Access Level
Indicates the level of access for role members in relation to the view.

Access to Child Resources
Grants access to all pages that are part of this view. After clicking this option, the **Access to Pages** table is updated with the new pages. If necessary, you can individually select pages from that table to remove access.

Access to Pages
This section provides a table displaying pages that have been assigned to this role. The following options are unique for the pages table.

Name
Displays the name of the page or node as it appears in the navigation.

Unique Name
Displays the identifier that uniquely identified the page in the portal.

The following fields and controls are available in all sections.

Select all icon
Selects all items displayed in the table for deletion. If you are displaying only a filtered set of items, only those items are selected. You can deselect specific items before actually deleting.

Deselect all icon
Deselects all items displayed in the table.

Add
Adds an item to the table.

Remove
Removes all selected items from the table. There is no warning prompt when you click **Remove**.

Filter
Type in this field to quickly find an item in the table. This field is useful when there are a large number of items to look through.

Select
Selects or deselects a single item in the table.

Access Level
Indicates the actions that users in the role can perform on the view or page.

Setting up users to modify event lists

To enable a TBSM user to view or modify events sent from the Netcool/OMNIbus ObjectServer, you must assign the user to certain access groups. You assign user roles to groups or to individual users.

About this task

In TBSM, a user can view and modify events from the default TBSM Netcool/OMNIbus ObjectServer. This user must have superuser privileges on the Netcool/OMNIbus ObjectServer.

Procedure

1. Add users to the Netcool/OMNIbus ObjectServer as described in the *Netcool/OMNIbus Administration Guide*.

Note: If you want a user to be able to modify the events in the AEL, the user must have `AlertsUser` or `CatalogUser` permissions on the Netcool/OMNIbus ObjectServer. These permissions are part of the Normal user group in Netcool/OMNIbus.

Note: If you selected a Netcool/OMNIbus ObjectServer as the default user repository when you installed TBSM, you can add users to the ObjectServer using the options available in the **Manage Users** page under **Users and Groups**.

2. Log in to the Dashboard Application Service Hub as a user with administration permissions.
3. In the left navigation pane click **Settings > Group Roles**, and then click **Group Roles**.
4. Assign the following additional roles to the user's group so that the user can view event lists.
   - `tbsmViewRawEvents`
   - `ncw_user`

   Note: The `tbsmViewRawEvents` role is assigned to the `tbsmAdmins` and `tbsmReadOnly` groups during the installation of TBSM. The `ncw_user` role is assigned to the `Netcool_OMNIbus_Admin` and `Netcool_OMNIbus_User` groups during the installation of TBSM.
5. In order to modify events in the AEL, assign the `netcool_rw` role to a user's group.

   Note: The `netcool_rw` role is assigned to the `Netcool_OMNIbus_Admin` group during the installation of TBSM

Modifying the default service and template privileges

If you want to change the default access privileges for service instances and service templates, you must edit the `TBSM_privs.props` file. This file is in the `$IMPACT_HOME/etc` directory.

About this task

The `TBSM_privs.props` file sets the default permissions for service instances and service templates. When you create new service instances and templates, the users and groups specified in the `TBSM_privs.props` file are granted access automatically. This does not apply to services created with ESDA or auto-population rules.

If the users or groups specified in the `TBSM_privs.props` file are removed from the user repository, the `TBSM_privs.props` file must be updated. Otherwise, service and template permissions does not function correctly. You do not need to update the `TBSM_privs.props` file when you create a user or group.

Procedure

1. Using a text editor, open the `TBSM_privs.props` file.
2. Modify the file as needed.

   For example, the following code grants the following access privileges:
   - Administration privileges to users who create a service or service template
   - Members of the `tbsmViewAllServicesUsers` group can view new services that are created manually
   - Members of the `tbsmUsers` group can view new service templates that are created manually

```
# new instance privs
impact.privs.newinstance.privname.1=tbsmServiceAdmin
impact.privs.newinstance.granttouserorgroup.1=User
impact.privs.newinstance.granttoname.1=CURRENT
impact.privs.newinstance.privname.2=tbsmViewService
impact.privs.newinstance.granttouserorgroup.2=Group
impact.privs.newinstance.granttoname.2=tbsmViewAllServicesUsers
# new template privs
impact.privs.newtemplate.privname.1=tbsmTemplateAdmin
impact.privs.newtemplate.granttouserorgroup.1=User
impact.privs.newtemplate.granttoname.1=CURRENT
impact.privs.newtemplate.privname.2=tbsmViewTemplate
```

```
impact.privs.newtemplate.granttouserorgroup.2=Group
impact.privs.newtemplate.granttoname.2=tbsmUsers
```

Changing the IBM Tivoli Business Service Manager configuration

You can change TBSM after installation. Possible changes include changes to supporting applications, such as IBM Tivoli Netcool/OMNIbus. This might be useful if you install a supporting application on a new host or change the port number that it uses.

Changing the IBM Tivoli Netcool/OMNIbus host and port

If you install Netcool/OMNIbus on another system or change the port number that it is listening on, you must reconfigure IBM Tivoli Business Service Manager to use the new values. To do so, edit the `TBSM_server.props`, `TBSM_ObjectServer_DS.ds`, and `TBSM_OutputObjectServer_DS.ds` files.

Before you begin

Before you edit the files in this procedure, create backup copies.

About this task

Procedure

1. Stop the TBSM Data server.
2. Open the `TBSM_ObjectServer_DS.ds` file. This file is in the `$IMPACT_HOME/etc` directory.
3. Change the values of the following properties to match your updated ObjectServer configuration:
   - **ObjectServer_DS.ObjectServer.PRIMARYHOST**
   - **ObjectServer_DS.ObjectServer.PRIMARYPORT**
   - **ObjectServer_DS.ObjectServer.BACKUPHOST**
   - **ObjectServer_DS.ObjectServer.BACKUPPORT**
4. Save the modified `TBSM_ObjectServer_DS.ds` file.
5. In the `/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/ etc/RAD_server.props` file, update the properties:
   - **impact.authentication.objectserver.host**
   - **impact.authentication.objectserver.port**
6. Open the `TBSM_OutputObjectServer_DS.ds` file. This file is in the `$IMPACT_HOME/etc` directory.
7. Change the values of the following properties to match your updated configuration:
   - **OutputObjectServer_DS.ObjectServer.PRIMARYHOST**
   - **OutputObjectServer_DS.ObjectServer.PRIMARYPORT**
   - **OutputObjectServer_DS.ObjectServer.BACKUPHOST**
   - **OutputObjectServer_DS.ObjectServer.BACKUPPORT**
8. Save the modified `TBSM_OutputObjectServer_DS.ds` file.
9. When you change the ObjectServer host or port, you must also update the Netcool/OMNIbus Web GUI data source included with TBSM. Open the `/opt/IBM/netcool/gui/omnibus_webgui/etc/datasources/ncwDataSourceDefinitions.xml` file.
10. Change the values of the following properties to match your updated configuration:
    - **<ncwPrimaryServer>**
    - If you have failover set up, you might need to set host and port for your backup ObjectServer with the **<ncwBackUpServer>** property

For more information about Netcool/OMNIbus Web GUI data source configuration, see Configuring a data source.

11. If the Objectserver name was changed from the default value of NCOMS, you must update the name of the **<ncwPrimaryServer>**, for example, **< ncwDataSourceEntry name="NCOMS_BKUP">** , **<ncwDataSourceDefinition type="singleServerOSDataSource" name="NCOMS_BKUP" enabled="true">**. You must also update the ObjectServer name in the `/opt/IBM/netcool/gui/omnibus_webgui/etc/default/data/system/view.xml` file.
12. Save the modified `ncwDataSourceDefinitions.xml` file.
13. Update the value of **<config:CustomProperties name="host1" value=" ">** parameter in the following file:
    - `/opt/IBM/JazzSM/profile/config/cells/JazzSMNode01Cell/wim/config/wimconfig.xml`
14. Restart the Data server and the Dashboard server.

Changing the Netcool/OMNIbus ObjectServer password or user ID

You can change the user ID or password for the IBM Tivoli Netcool/OMNIbus ObjectServer used by TBSM.

About this task

Note: Sometimes TBSM cannot configure correctly with custom user ID and password information, especially if you are connecting to a preexisting OMNIbus ObjectServer. To avoid this problem, use the default `root` and blank user ID/password combination when initially setting up TBSM.

Procedure

1. Stop the TBSM Dashboard server and Data server.

   The list below contains sample information of the Environment Variables and their path assignments.

   NETCOOL_HOME
   `C:\Program Files\IBM\tivoli\netcool`

   OMNIHOME
   `C:\Program Files\IBM\tivoli\netcool\omnibus`

   TBSM_DASHBOARD_SERVER_HOME
   `C:\Program Files\IBM\JazzSM\profile\installedApps\JazzSMNode01Cell\isc.ear`

   TBSM_HOME
   `C:\Program Files\IBM\tivoli\tbsm`

   TBSM_INSTALL_HOME
   `C:\Program Files\IBM\tivoli`
2. Create a backup copy of each file listed below.

```
<IMPACT_HOME>\etc\TBSM_OutputObjectServer_DS.ds
<IMPACT_HOME>\etc\TBSM_ObjectServer_DS.ds
<IMPACT_HOME>\etc\TBSM_sla.props
<IMPACT_INSTALL_HOME>\netcool\omnibus_webgui\etc\
datasources\ncwDataSourceDefinitions.xml
```

3. Using the OMNIbus `nco_config` tool, change the password of the ObjectServer user ID in OMNIbus. Tool can be found under the <OMNIHOME>\bin directory.
4. Using the TBSM `rad_crypt` utility, encrypt the password of the ObjectServer user ID you specified in the prior step. Take note of the encrypted value; you need it to update the files listed in the next step. To encrypt the password, enter it after the **rad_crypt** command as seen below:

```
<TBSM_HOME>\bin\rad_crypt.bat <password>
```

Note: In the following example, `usepass` is the password to be encrypted.

```
<TBSM_HOME>\bin\rad_crypt.bat usepass
7272747C8C6E7CB384F3A03914CFB7F0109D1BC0887ED6F7E5B6AC22940DE2CC
```

5. Update the following files with the new encrypted password. The variable to update is listed under each file:

   IMPACT_HOME\etc\TBSM_ObjectServer_DS.ds
   `ObjectServer_DS.ObjectServer.DBPASSWORD=`

   IMPACT_HOME\etc\TBSM_OutputObjectServer_DS.ds
   `OutputObjectServer_DS.ObjectServer.DBPASSWORD=`

   Note: If the Netcool/OMNIbusObjectServer is running in Secure mode, that is, the ObjectServer is started with the `-secure` command line option, add the following parameters to the files:

   Dashboard server: `/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_sla.props`

   Data server: `$IMPACT_HOME\etc\rad\RAD_sla.props`

   ```
   impact.objectserver.securemode=true
   impact.objectserver.secureusername=user
   impact.objectserver.securepassword=password encrypted using nco_crypt utility
   ```

6. When you change the ObjectServer password, you must also update the password for the Netcool/Webtop included with TBSM. Open the `<NCHOME>\omnibus_webgui\etc\datasources\ncwDataSourceDefinitions.xml` file.
7. Change the values of the **ncwDataSourceCredentials** property to match the new encrypted password.

   The password goes on the following stanza:

   ```
   <ncwDataSourceCredentials
            userName="root" password=""
               encrypted="true" algorithm="FIPS"
   ```

   Note: The `userName` and `password` attributes are required. The `encrypted` and `algorithm` attributes are required only if the Netcool/OMNIbus ObjectServer is running in Secure mode, that is to say, if the ObjectServer is started with the `-secure` command line option. For further information about starting the ObjectServer in secure mode, see the Netcool/OMNIbus ObjectServer *Administration Guide*.

   Note: If the password is not `encrypted`, set the value for the `encrypted` attribute to `false`. The `algorithm` attribute is not required. If the password is encrypted using FIPS 140-2, set the value for the `encrypted` attribute to `true` and the `algorithm` attribute to FIPS. If you are not using FIPS 140-2, set the `algorithm` attribute to AES.
8. Start the TBSM Dashboard server and TBSM Data server.

What to do next

If you also use Netcool/OMNIbus as a user repository, you need to complete the additional steps described in "Change the user ID or password used to access Netcool/OMNIbus as the User Repository" on page 43.

Change the user ID or password used to access Netcool/OMNIbus as the User Repository

This topic describes how to change the user ID or password used to access OMNIbus as the User Repository.

Before you begin

Before you start this task, you must complete the steps described in “Changing the Netcool/OMNIbus ObjectServer password or user ID” on page 41.

About this task

In addition to completing the steps described in the previous topic, you also need to complete the following steps to change the user ID or password of the OMNIbus User Repository:

Procedure

1. Stop TBSM Dashboard server and the TBSM Data server.
2. Create a backup copy of each file listed below.

   ```
   /opt/IBM/JazzSM/profile/config/cells/JazzSMNode01Cell/wim/config/wimconfig.xml
   ```

3. For the Dashboard server, run the script `confvmm4ncos_full.sh` on UNIX systems or the script `confvmm4ncos_full.bat` on Windows systems to configure OMNIbus as the user repository. This script is located in `/opt/IBM/JazzSM/bin`

   When this script is run without arguments, the command returns a usage message. Supply the following arguments:

   `wasdir`: The root directory location of the server installation (equivalent to `/opt/IBM/JazzSM`)
   `profile`: `profile`.
   `cell`: `JazzSMNode01Cell`.
   `username`: OMNIbus user name to connect to OMNIbus with (for example, root)
   `password`: Password for the specified user name (if you specify a null or empty password, enter two double quotation marks [""])
   `host1`: Host name of the OMNIbus server (the host name of the primary host if you are using a failover environment)
   `port1`: Port number (for example, `4100`; this number is the port number of the primary server if you are using a failover environment)
   `host2`: (Optional) Host name of a backup OMNIbus server if you are using a failover environment
   `port2`: (Optional) Port number of backup OMNIbus server if you are using a failover environment
4. Update the Data server with the OMNIbus user repository, see https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.12/com.ibm.netcoolimpact.doc/admin/imag_objectserver_user_authentication.html
5. Start the TBSM Dashboard server and TBSM Data server.

Changing the TBSM port numbers

If you want to change the port numbers that TBSM uses, you must uninstall and then reinstall the application.

Before you begin

Note: If you change the TBSM port numbers, other components might not work with TBSM correctly.

Procedure

1. Back up the TBSM configuration.
2. Uninstall TBSM.
3. Reinstall TBSM.
4. Restore the configuration that you backed up in step “1” on page 44.

Updating TBSM data sources

You can change the user ID or password and other properties of TBSM data sources with the utility described in this topic.

About this task

TBSM provides four data sources that are used by the product to access the TBSM DB2 databases. These data sources are called `TBSMDatabase`, `TBSMComponentRegistry`, `TBSMMetricHistory`, and `TBSMMarker` and provide the connection information for one or more DB2 databases you configured for TBSM.

These data sources are affected by changes to the database configuration. For example, the common scenario where the DB2 database password is changed on a required interval. Other scenarios where these data sources are affected include changing the user ID that accesses the database, or changes to connection information (like database name, hostname, and port) that result from moving the database to a new DB2 server.

TBSM includes a utility that can be used to update the TBSM DB2 data sources and test the new connection when the connection information must change. Only update these data sources when the Data server is stopped and the changes become effective on the next start of the Data server. This utility also updates connection information used by the Discovery Library Toolkit as required.

Note: The data sources `TBSMDatabase` and `TBSMComponentRegistry` require the same connection information so updates are made to both when `TBSMDatabase` is specified for update.

Each data source must be updated separately, as described in this procedure:

Procedure

1. Stop the Data server and the Discovery Library Toolkit.
2. Change to the `$TBSM_HOME/tools/etc` directory.
3. In a text editor, modify the properties file for the data source you want to update.

   For example, for the TBSM Database, open the `TBSMDatabase_ds_update.properties` file. Each properties file has comments about required values and optional values.
4. Change to the directory: `$TBSM_HOME/tools/bin`
5. Run the command:

   Windows Windows

   ```
   TBSMUpdateDatasource.bat Datasourcename
   ```

   UNIX UNIX

   ```
   TBSMUpdateDatasource.sh Datasourcename
   ```

   The value for the data source name depends on the data source you want to update. For example:

   - **TBSMUpdateDatasource.bat** TBSMDatabase
   - **TBSMUpdateDatasource.bat** TBSMMetricHistory
   - **TBSMUpdateDatasource.sh** TBSMMarker

6. Start the Data server and the Discovery Library Toolkit.

Initializing a TBSM database

Initialize a newly defined database or one that has been reset.

About this task

In some instances, you may create a new database for use with a TBSM Data server using the DbConfig database installer. Or you may use the **tbsm_db** utility to reset an existing TBSM database, clearing all the data, including the data loaded at install time to initialize the database. In this scenario, the TBSM database is not sufficiently initialized for the TBSM servers and the Discovery Library Toolkit to function correctly.

TBSM includes a utility that you can use to initialize a newly defined database or one that has been reset. This utility will perform the functions that are normally performed during the installation of the Data server. This utility assumes the database is already connected to the Data server, and requires the Data server to be active so that the **rad_radshell** utility can load required templates and services.

To run the utility:

Procedure

1. Make sure the Data server is started, as the utility will fail if the **rad_radshell** utility cannot connect to the Data server. The utility stops the Discovery Library Toolkit.
2. Change to the `$TBSM_HOME/tools/bin` directory.
3. Run the command:

   Windows Windows

   ```
   TBSMInitDB.bat dbpassword
   ```

   UNIX UNIX

   ```
   TBSMInitDB.sh dbpassword
   ```

   Where *dbpassword* is optional and specifies the password used by the TBSM data server to connect to the TBSM database. If you do not specify a password you are prompted for the password value.
4. Restart the Data server.
5. If it is not running, start the Discovery Library Toolkit or restart it if it is running.

Manually configuring TBSM for external user repositories

If you selected the file-based repository option during the installation, you can manually configure TBSM to use an LDAP repository.

You can configure TBSM to use an ObjectServer as an external user repository or authentication source during an advanced installation. However, if you want to use an LDAP user repository, you must configure it after the installation has completed. To configure TBSM to authenticate to an LDAP repository, select File Registry as your user repository during an advanced installation. You can then configure TBSM to use an LDAP authentication source after the installation has completed.

If you have an existing LDAP or OMNIbus authentication source that is already configured for secure communications over SSL, you must also select the file-based repository option during the installation and manually configure TBSM and the Dashboard Application Service Hub to communicate with your external repository over a secure (SSL) channel.

At least each Dashboard server needs to be configured to communicate with the external repository that you have chosen. You can optionally configure the Data server, including the backup Data server in a

failover configuration, to communicate with the external repository for environments in which user validation and authentication functions are required.

For more information about configuring LDAP with Dashboard Application Service Hub, see "Adding an external LDAP repository" on page 291.

Configuring TBSM to use an external user repository after installation

When you install TBSM, the `tbsmadmin` and `tbsmuser` users and 4 groups, `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsers`, are created in the target user repository. By default this is a IBM Tivoli Netcool/OMNIbus repository. After installation, you can configure TBSM to use an external user repository. However, if you select a repository after installation that contains the same user ID or groups as those already in the default repository, you must remove any duplicate groups from the default repository before you configure the external repository. If you do not, you might not be able to log into TBSM using the default administrative user ID.

About this task

To redefine the TBSM user and groups in the default repository and add them to the external user repository:

Procedure

1. Log in to TBSM as `tbsmadmin` or any user that has the authority to manage users and groups.
2. On DASH UI, go to **Settings -> WebSphere Admin Console-> Users and Groups -> Manage Users**.
3. In the **Search for** field, type the user ID for the default user `tbsmadmin` that was created during installation. Click **Search** and delete the user from the list that is displayed. Repeat this step for the `tbsmuser` user.
4. From the left navigation pane, click **Users and Groups** and then click **Manage Groups**.
5. In the **Search for** field, type the names of the default groups, `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsersthat`, that were created during the installation. Click **Search** and delete these groups from the list that is displayed.
6. In the external user repository, create the TBSM users `tbsmadmin` and `tbsmuser` and the TBSM groups `tbsmAdmins`, `tbsmReadOnly`, `tbsmUsers`, and `tbsmViewAllServicesUsers`. Add the `tbsmadmin` user to the `tbsmAdmins` group and add the `tbsmuser` user to the `tbsmReadOnly` group.

   Note: This step can be completed using the `/opt/IBM/tivoli/tbsmdash/bin/vmm_create_entities.sh` script on the Dashboard server. This is done after the new repository has been successfully configured. To ensure that the `vmm_create_entities` script is successful, the Dashboard server must have write-access to the new repository and must have been configured to specify the new repository as the default. For more information, see "Configuring the default repository for the Dashboard server" on page 47.
7. After the TBSM user and groups have been created in the external user repository, run the `$TBSM_HOME/bin/assign_group_roles.sh` script on the Dashboard server to recreate the user-to-role and group-to-role assignments for TBSM.

   Note: This procedure is required once regardless of the number of Dashboard servers.

   Note: This procedure does not address additional users and groups that you create during installation such as those created for WebGUI. Please consult OMNIbus WebGUI documentation for further information.

Configuring a Dashboard server for an external LDAP repository

You must, as a minimum configuration, configure each Dashboard server to communicate with the external repository. This procedure assumes that you selected the file-based repository option during installation.

About this task

To configure a Dashboard server to communicate with an external LDAP repository, perform the following steps:

Procedure

1. If the LDAP server is configured for SSL communications, import the signer's certificate for the LDAP server into the trust store of the dashboard server by doing one of the following:
   - Use the GUI.
   - Use the command line. See "Importing the signer certificates into the server trust store" on page 49.
2. Configure the TBSM server for LDAP using the Dashboard Application Service Hub UI.

   For more information, see "Configuring" on page 291
3. Optional: If you want to perform user management functions (add, delete, or organize users and groups) using the Dashboard Application Service Hub GUI, see "Configuring the default repository for the Dashboard server" on page 47. Also for Dashboard Application Service Hub, see the TBSM *Installation Guide*.

Configuring the default repository for the Dashboard server

If you want to create new users and groups from the user management interfaces in Dashboard Application Service Hub, you must configure the server with the target locations in which the new users and groups resides in the default repository. Although a federated repository in Dashboard Application Service Hub can read user and group information and can authenticate users from multiple registries, you can specify only one repository as the target for user and group creation.

Before you begin

If you are using Microsoft Active Directory 2003, the connection from the Dashboard server to the Microsoft Active Directory server must be over SSL. If the connection is not over SSL, you cannot create new users.

About this task

To configure the default repository so that you can create users and groups in Dashboard Application Service Hub, perform the following steps:

Procedure

1. Log in to the Dashboard Application Service Hub console as an administrative user (for example, `tbsmadmin`).
2. Click **Settings** and then click **WebSphere Administrative Console**.
3. Click **Launch WebSphere Administrative Console**. Within the WebSphere Administrative Console click **Security** ->**Global Security**.
4. In the **Global Security** page, accept the default values (**Federated repositories**), and click **Configure**.

   The **General Properties** panel displays, which contains configuration information for federated repositories.
5. In the **Additional Properties** section at the bottom of the panel, click **Supported entity types**.

   The **Supported entity types** panel displays.

6. In the table, click the **Group** entity type, and complete the **Base entry for the default parent** field.

   The base entry for the default parent is the location in the default repository where objects of that type (in this case, group objects) are created. Click **OK**.
7. Click **OrgContainer** and complete the **Base entry for the default parent** field. Click **OK**.
8. Click **PersonAccount** and complete the **Base entry for the default parent** field. Click **OK**.

   The `PersonAccount` entity type specifies where new users are created.

   Note: If you are using Microsoft Active Directory 2003, specify `cn` in the **Relative Distinguished Name properties** field.
9. Restart the Dashboard server for the changes to take effect.

What to do next

Verify the changes by logging in to the Dashboard Application Service Hub as an administrative user (for example `tbsmadmin`). Click **Users and Groups** and then click **Manage users**. Click **Create**, enter the required information for a new user, and click **Create** again. The new user must be created in the default repository.

Configuring a TBSM server for an OMNIbus user repository

You can configure OMNIbus to be the user repository after you have installed TBSM. You must configure each TBSM server in your system.

About this task

OMNIbus is the default user repository that is configured for TBSM during installation. If you want to configure and alternative user repository, perform an advanced installation and choose **File Registry** as your user repository. You can then configure an alternative user repository after the installation has completed.

Procedure

1. Change the user repository.

   For the Data server, see: https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.12/com.ibm.netcoolimpact.doc/admin/imag_objectserver_user_authentication.html

   For the Dashboard server, see: https://www.ibm.com/support/knowledgecenter/en/SSSHTQ_8.1.0/com.ibm.netcool_OMNIbus.doc_8.1.0/webtop/wip/task/web_config_vmm4objectserver.html
2. For the Data server only: Edit the `/opt/IBM/tivoli/impact/etc/TBSM_sla.props` file and ensure that the **`impact.sla.dataserver.bypassvalidation`** property is set to `false`.
3. Restart the server that you configured.
4. Dashboard server only: Verify that the Dashboard server that you configured is communicating with the OMNIbus repository by completing the following steps:
   a) Log in to the Dashboard Application Service Hub console as an administrator (for example, `tbsmadmin`).
   b) Click **Settings** and then click **Manage Users**.
   c) Click **Search**.

      The users that are defined in the OMNIbus repository appears in the list of users that is returned. The value for the **Unique Name** field for OMNIbus users must contain the suffix `o=netcoolObjectServerRepository`.
5. Data server only: Verify that the Data server that you configured is communicating with the OMNIbus repository by completing the following steps:
   a) Change to the `$TBSM_HOME/bin` directory.
   b) Issue the **`rad_radshell`** command by entering `$TBSM_HOME/bin/rad_radshell.sh` *`OMNIbus_user`*,

where *OMNIbus_user* is a user ID of a user who is defined in the OMNIbus repository. If the command connects to the Data server and correctly validates the user, a RAD shell command-line prompt returns. Type `exit();` to exit from the **radshell** utility.

What to do next

Optional for a Dashboard server: If you want to perform user management functions (add, delete, or organize users and groups) from the Dashboard Application Service Hub GUI, see "Configuring the default repository for the Dashboard server" on page 47.

Importing the signer certificates into the server trust store

You can retrieve the certificate and add it to the trust store directly from the secure port of the LDAP server by issuing **wsadmin** commands.

Procedure

1. For the Data server, execute the following commands:

```
cd /opt/IBM/tivoli/impact/sdk/bin/

./keytool -exportcert -alias default -keystore /opt/IBM/tivoli/
impact/wlp/usr/servers/TBSM/resources/security/key.jks -file cer.cer

./keytool -import -trustcacerts -file ./cer.cer -keystore /opt/IBM/tivoli/
impact/wlp/usr/servers/TBSM/resources/security/trust.jks -alias "Data server
hostname"

./keytool -import -trustcacerts -file /tmp/cer.cer -keystore ./cacerts -
alias "Data server hostname"

./keytool -import -trustcacerts -file ./cer.cer -keystore /opt/IBM/tivoli/
impact/sdk/jre/lib/security/cacerts -alias "Data server hostname"

cd $TBSM_HOME/bin

./rad_radshell -password password -tls_strict

./rad_radshell -tls
```

2. For the Dashboard Server, see:

   https://www.ibm.com/support/knowledgecenter/SSEKCU_1.1.3.0/com.ibm.psc.doc/config/psc_t_config_ssl_ldap.html

What to do next

You must restart the server before you configure a secure connection to the LDAP server.

FIPS compliance

TBSM supports the US Federal Information Processing Standard 140-2 (FIPS 140-2) when using cryptographic algorithms to encrypt and decrypt passwords.

For more information about FIPS, see http://csrc.nist.gov/publications/fips/

Enabling FIPS on the Data server

TBSM password encryption algorithms on the Data server use FIPS-approved cryptographic providers regardless of whether FIPS is enabled for the entire Data server.

To enable FIPS on the Data server, see https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.12/com.ibm.netcoolimpact.doc/admin/imag_configuring_for_fips_compliancy.html

Enabling FIPS on the application server

You can configure the application server to use a Federal Information Processing Standard (FIPS) approved cryptographic provider.

About this task

Dashboard Application Service Hub password encryption algorithms on the application server use FIPS approved cryptographic providers regardless of whether FIPS is enabled for the entire application server. However, enabling FIPS on the application server ensures that the encryption used to support SSL communications, as well as Single Sign On, uses a FIPS-approved cryptographic provider.

Follow these steps to enable FIPS 140–2 for the application server.

Procedure

1. Configure the application server to use FIPS.
   a) Log in to the Dashboard Application Service Hub.
   b) In the navigation pane, click **Settings** > **Websphere Administrative Console** and click **Launch Websphere administrative console**.
   c) In the WebSphere Application Server administrative console navigation pane, click **Security** > **SSL certificate and key management**.
   d) Select the **Use the United States Federal Information Processing Standard (FIPS) algorithms** option and click **Apply**.
      This option makes IBMJSSE2 and IBMJCEFIPS the active providers.
   e) In the Messages area at the top of the page, click the **Save** link and log out of the WebSphere Application Server console.
2. Configure the application server to use FIPS algorithms for Java clients that must access enterprise beans:
   a) Open the `/opt/IBM/JazzSM/profile/temp/ssl.client.props` file in a text editor.
   b) Change the `com.ibm.security.useFIPS` property value from `false` to `true`.
3. Configure the application server to use FIPS algorithms for SOAP-based administrative clients that must access enterprise beans:
   a) Open the `/opt/IBM/JazzSM/profile/properties/soap.client.props` file in a text editor.
   b) Add this line:`com.ibm.ssl.contextProvider=IBMJSSEFIPS`.
4. Configure `java.security` to enable IBMJCEFIPS:
   a) Open the `/opt/IBM/WebSphere/AppServer/java_1.7_64/jre/lib/security/java.security` file in a text editor.
   b) Insert the IBMJCEFIPS provider (`com.ibm.crypto.fips.provider.IBMJCEFIPS`) before the IBMJCE provider, and also renumber the other providers in the provider list.
      The IBMJCEFIPS provider must be in the `java.security` file provider list. See the example at the end of this topic.
5. Enable your browser to use Transport Layer Security (TLS) 1.0:
   a) Microsoft Internet Explorer: Start Internet Explorer and click **Tools > Internet Options**. On the **Advanced** tab, select the **Use TLS 1.0** option.
   b) Firefox: Start Firefox and click **Tools > Options**. In the toolbar, click the **Advanced** icon and select the **Encryption** tab. In the Protocols frame, select the **Use TLS 1.0** option.
6. Export Lightweight Third Party Authentication keys so applications that use these LTPA keys can be reconfigured.
   a) In the navigation pane, click **Settings** > **Websphere Admin Console** and click **Launch Websphere Admin Console**.
   b) In the WebSphere Application Server administrative console, select **Security** > **Global security**.

c) In the Global security page, from the Authentication area, click the **LTPA** link.

d) Under **Cross-cell single sign-on**, specify a key file and provide a filename and password for the file that will contain the exported LTPA keys.

e) Click **Export keys**.

By default the exported file is saved to */opt/IBM/JazzSM//*profiles/DASHProfile/

7. Reconfigure any applications that use application server LTPA keys: To reconfigure the Tivoli SSO service with the updated LTPA keys, run this script: */opt/IBM/JazzSM//*profiles/DASHProfile/bin/setAuthnSvcLTPAKeys.jacl.

a) Change directory to */opt/IBM/JazzSM//*profiles/DASHProfile/bin/

b) If the application server is not running, start it using the following command:

- Windows `startServer.bat server1`
- Linux UNIX `startServer.sh server1`

c) Run the following command:

```
wsadmin -username tbsmadmin -password tbsmadmin_password -f
setAuthnSvcLTPAKeys.jacl exported_key_path key_password
```

Where:

exported_key_path is name and full path to the key file that was exported.

key_password is the password that was used to export the key.

8. For SSO, enable FIPS for any other application server instances, then import the updated LTPA keys from the first server into these servers:

a) Copy the LTPA key file from step "6" on page 50 above to another application server computer.

b) In the navigation pane, click **Settings** > **Websphere Admin Console** and click **Launch Websphere Admin Console**.

c) In the WebSphere Application Server administrative console, select **Security** > **Global security**.

d) In the Global security page, from the Authentication area, click the **LTPA** link.

e) Under **Cross-cell single sign-on**, provide the filename and password from above for the file that contains the exported LTPA keys.

f) Click **Import keys**.

9. Run the **ConfigureCLI** command:

Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh ConfigureCLI --useFIPS true`

Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ConfigureCLI --useFIPS true`

Example

The IBM SDK `/opt/IBM/WebSphere/AppServer/java/jre/lib/security` file looks like this when IBMJCEFIPS is enabled.

```
security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.jsse.IBMJSSEProvider
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.crypto.pkcs11.provider.IBMPKCS11
security.provider.8=com.ibm.security.cmskeystore.CMSProvider
security.provider.9=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
```

Chapter 6. Administering IBM Tivoli Business Service Manager

This section contains information about administration tasks that you perform on a regular basis.

Administering your TBSM database

A database can become unusable because of hardware or software failure, or both. Protect your data against the possibility of loss by regularly creating a backup of your database. It is recommended that you back up your database daily.

You can use this backup to restore your data if an issue occurs that results in database failure. For information about backing up and restoring a DB2 database, see the DB2 Information Center located: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.doc/welcome.html.

Operating the TBSM Data and Dashboard servers

Before you can start the TBSM Data and Dashboard servers, you must start IBM Tivoli Netcool/OMNIbus.

Before you start Netcool/OMNIbus with the **nco_objserv** command, you must set the *LANG* environment variable to a different locale that exists in the `locales.dat` file or add your locale to the `locales.dat` file. You can also use one of the TBSM start-up scripts that automatically handles the *LANG* variable.

Starting the TBSM Data and Dashboard servers on UNIX systems

This topic describes how to start the TBSM servers on UNIX systems.

Procedure

1. To start the TBSM Data server, issue the following command:

   ```
   $IMPACT_HOME/bin/startImpactServer.sh server1 -profileName TBSMProfile
   ```

2. To start the TBSM Dashboard server, issue the following command:

   ```
   /opt/IBM/JazzSM/profile/bin/startServer.sh server1
   ```

3. Optional: You can view the TBSM startup logs by examining the following files:
   - Data server: **$IMPACT_HOME/logs/trace.log**
   - Dashboard server: **/opt/IBM/JazzSM/profile/logs/server1/trace.log**

   A message displays to indicate that the operation has been successful.

Stopping the TBSM Data and Dashboard servers on UNIX systems

This topic describes how to stop the TBSM servers on UNIX systems.

About this task

Procedure

1. To stop the TBSM Data server, issue the following command:

   ```
   $IMPACT_HOME/bin/stopImpactServer.sh
   ```

2. To stop the TBSM Dashboard server, issue the following command:

```
/opt/IBM/JazzSM/profile/bin/stopServer.sh server1 -username
tbsmadmin -password tbsmpassword
```

   By default, *username* is `tbsmadmin` and *password* is the `tbsmadmin` password.

Starting the TBSM Data and Dashboard services on Windows systems

This topic describes how to start TBSM Data and Dashboard services on Windows systems.

Before you begin

To start TBSM services, perform the following steps:

Procedure

1. Select **Administrative Tools** > **Services**.

   The **Services** window opens. Depending on which services you have installed, the available services are as follows:

   - Tivoli Business Service Manager: `NetcoolImpactTBSM_9080` (the Data server).
   - Dashboard Application Service Hub: `%JazzSM_HOME%\profile\bin\startServer.bat server1` (the Dashboard server)

     Note: These names are the default name. If different ports are used, the last part of the service name reflects the ports in use.
2. Right-click the service that you want to start and click **Start**.
3. Optional: You can view the TBSM startup logs by examining the following files:
   - Data server startup log: `%IMPACT_HOME\logs\server1\trace.log`
   - Dashboard server startup log: `C:\Program Files\IBM\JazzSM\profile\logs\server1\trace.log`

Stopping the TBSM Data and Dashboard services on Windows systems

You can stop the TBSM Data and Dashboard services on Windows systems using the steps shown in this topic.

About this task

To stop TBSM services, complete the following steps:

Procedure

1. Select **Administrative Tools** > **Services**.

   The **Services** window opens. Depending on which services you have installed, the available services are as follows:

   - Tivoli Business Service Manager: `TBSMProfile_Port_17310`. (Data server)
   - Dashboard Application Service Hub: `%JazzSM_HOME%\profile\bin\stopServer.bat server1`. (Dashboard server)
2. Right-click the service that you want to stop and click **Stop**.

Administering TBSM using the RAD shell tool

The RAD shell tool is available only on the TBSM Data server. The RAD shell tool allows you to administer and configure TBSM from the command line. The file that contains the RAD shell functions is named

`scriptedAPIStartup.bsh`. It is located in the `/opt/IBM/tivoli/impact/etc` directory. If you are familiar with scripting languages, you can create your own custom functions for the RAD shell tool; however, IBM Support does not assist in writing custom scripts, nor does it support them.

Before you begin

You can use WebSphere Application Server command-line tools for user management tasks. The RAD shell commands only manage permissions for TBSM-specific objects, such as service templates. For more information about WebSphere Application Server command-line tools, see Supported commands.

Note: If you use an external repository such as LDAP, user management can only be done from WebSphere command interfaces if the connection from WebSphere to LDAP is established with a user that has read and write privileges in the repository. Otherwise, if the connection to an external repository is read-only (as it is in many environments), these commands does not work. In this scenario, user management would be done by the administrator of the external repository using repository tools.

Starting the RAD shell tool

After you have started the RAD shell tool, you can administer and configure TBSM from a command-line session..

Procedure

From a command prompt, type the following command and press **Enter**:

For non-TLS config:

UNIX `$TBSM_HOME/bin/rad_radshell`

Windows `%TBSM_HOME%\bin\rad_radshell`

For TLS config:

UNIX `$TBSM_HOME/bin/rad_radshell -tls`

Windows `%TBSM_HOME%\bin\rad_radshell -tls`

For TLSv1.2 Strict config:

UNIX `$TBSM_HOME/bin/rad_radshell -tls_strict -password {keyStore password}`

Windows `%TBSM_HOME%\bin\rad_radshell -tls_strict -password {keyStore password}`

Note: The default password (WebAS) is used when no "-password" entered.

Sending file contents to the RAD shell tool

You can send the contents of a file from a UNIX or Windows command prompt to the RAD shell tool. The RAD shell tool accepts input in either US-ASCII or UTF-8 format. Byte order markers (BOMs) must not appear in the input stream.

Procedure

From a command prompt, issue the following command and press **Enter**:

UNIX `cat` *`file_name`* `| $TBSM_HOME/bin/rad_radshell`

Windows `type` *`file_name`* `| %TBSM_HOME\bin\rad_radshell`

where *file_name* is the fully qualified name of the file that contains the output that you want to process with the RAD shell tool.

Viewing help for the RAD shell tool

The RAD shell tool includes several types of online help, including general help, service-instance help, and help for functions with event attributes.

Procedure

From a rad_radshell prompt, issue one of the following commands and press **Enter**:

Description	Command
Lists all available functions	`help();`
Lists help for all functions with "token" in the name or parameter list. For example, to learn about functions used to manage service instances, type help ("ServiceInstance");	`help("token");`

RAD shell commands

You can use the RAD shell commands to create and modify service templates and service instances from the command line. The RAD Shell tool is implemented in the Java application programming interface (API) and uses the coding conventions of the Java API.

Auto-population rule commands

Auto-population rules allow TBSM to create service instances automatically based on data from incoming events.

addESDARule

Use the `addESDARule` function to add an ESDA rule to a service template.

Syntax

```
addESDARule("ruleName",
"templateName",
relation,
"policy",
"policyScript",
"dataSource",
"query",
"instanceNameExpression"
"displayNameExpression"
"descriptionExpession"
"tagNames",
"primaryTag,
"restrictionFilter",
enabled true false,
"attributeExpressions");
```

Parameter

This function has the following parameters.

Table 2. `addESDARule` Parameters

Parameter	Format	Description
ruleName	String	The name of the ESDA rule.
templateName	String	The name of the service template for the ESDA rule.

Table 2. addESDARule Parameters (continued)		
Parameter	**Format**	**Description**
relation	Integer	Identifies if the rule is used to discover child or parent services. Enter `0` for a child service and 1 for a parent service.
policy	String	Optional. The name of the policy used for the restriction filter.
policyScript	String	Optional. The full text of the policy you created for this rule.
datasource	String	The name of the external data source used for the ESDA rule.
query	String	The query used to select data for the ESDA rule from an external data source.
instanceNameExpression	String	Enter an expression for the service name. For example, enter: `""Application" +__serviceinstancename__ + applicationid"`
displayNameExpression	String	Optional. If you want a different display name other than the default name defined in the `instanceNameExpression` parameter, enter an expression in this optional field. This expression must be in the TBSM expression language. Otherwise, the name is the same as the service name defined in the service-template status rule.
descriptionExpression	String	Optional. If you want to include a description of the discovered service, enter an expression in this parameter.
tagNames	String	The names of the service template or templates you want assigned to the discovered services. If you have multiple templates assigned, separate each template name with three colons (`:::`).
primaryTag	String	The primary service template for the discovered services.
restrictionFilter	String	Optional. Enter an SQL WHERE clause for this parameter if you want to apply additional conditions when a service instance is auto-created.
enabled	Boolean	If you want to enable the new rule, enter `true`. Otherwise, enter `false`.
attributeExpressions	String	Optional. The expressions for the additional attributes (properties) for the discovered services. By default, an additional property is set for each field returned by the query.

Note: Be careful typing the field names for the name expressions such as *InstanceNameExpression*. Spelling and case errors causes your rule to malfunction. Some data sources, such as MS-SQL have case-sensitive field names. Run a query against your data source to make sure that the spelling and case are correct for the field name.

Example

The following example configures an ESDA rule with the following parent-child relationships:

1. The function name `addESDARule`.
2. The rule name is `ESDARule1`.
3. The service template name is `ESDA_Application`.
4. This rule is used to discover child services. The relation type is set to `0`.
5. The name of the default ESDA policy is `ESDA_ExecuteQuery`.
6. Blank. If you were creating your own policy, you insert the full text of the policy here.
7. The name of the external data source is `ServiceInstanceGUI`.

8. The query selects rows from a table called esda_application where the value in the `applicationname` field matches the service instance name of the parent service. The `__serviceinstance__` value is a variable that holds the name of a given parent service instance.
9. The service instance name expression uses the value from the `hostname` field to create the service name.
10. The display name expression matches the service instance name expression.
11. The description expression is hard coded to display `Hosts`.
12. The name service templates assigned to the discovered service are `ESDA_Hosts` and `Hosts`.
13. The primary template for the discovered services is `ESDA_Host`.
14. The restriction filter is blank.
15. The rule is enabled when this parameter is set to true.
16. The attributeExpression set the default value. This setting creates an additional property for each field returned by the ESDA query.

```
[1]   addESDARule(
[2]   "ESDARule1",
[3]   "ESDA_Application",
[4]   0,
[5]   "ESDA_ExecuteQuery",
[6]   null,
[7]   "ServiceInstanceGUI",
[8]   "select * from esda_application where
applicationname='__serviceinstancename__'",
[9]   "hostname",
[10]   "hostname",
[11]   "\"Hosts\"",
[12]   "ESDA_Hosts:::Hosts",
[13]   "ESDA_Hosts",
[14]   "",
[15]   true,
[16]   "AllSeedFields:::AllOrgNodeFields");
```

Note: Setting of the `attributeExpressions` to only `AllOrgNodeFields` should show the `PROP_` and `VAL_` for the level instances only in the additional parameters tab.

Note: The `addESDARule` command cannot be used to update/edit this aspect of an ESDA rule and it must be recreated after the deletion of the initial rule.

addToAutoPopulationRule

Use the `addToAutoPoplulationRule` function to add a parent relationship to an auto-population rule. These are the same parameters you specify after you click the **Auto-population Rules** button in the **Templates** tab of the **Service Navigation** panel in the IBM Tivoli Business Service Manager GUI. See the TBSM *Service Configuration Guide* for more information about auto-population rules.

Syntax

`addToAutoPoplulationRule("`*rootServiceTypeName*`",`
"*rawAttributeName*",
"*parentName*",
"*parentNameExp*",
"*displayNameExp*",
"*slaName*",
isEnabled,
"*relationshipRule*",
"*filter*"
"*StatusRuleName*"
"*RuleName*"
"*instanceNamePolicy*",
"*instanceNamePolicyScript*,
"*dataTypeForInstanceNamePolicy*",

"*queryForInstanceNamePolicy*",
"*attributeCreationPolicy*",
"*attributeCreationPolicyScript*",
"*attributeExpressions*",
"*tagNames*",
"*maintenanceWindow*",
"*groupName*",
"*dataFeed*");

Parameters

This function has the following parameters.

Table 3. addToAutoPopulationRule

Parameter	Format	Description
rootServiceTypeName	String	The name of the service template for the event-based settings of the auto-population rule. The services instances created with this root service type are the child services in the parent-child relationship you configure with this function.
rawAttributeName	String	The name of the incoming-status attribute for the child service instances.
parentName	String	The name of the parent service template you want to use for the auto-population rule.
parentNameExp	String	Enter an expression to define the instance name with this parameter. This expression must be in the Netcool/Impact expression language. If you simply want to use the existing name of a parent service instance, type the parent name in quotation marks.
displayNameExpression	String	Optional. If the parent instance does not exist yet, enter an expression in this optional field. This expression must be in the TBSM expression language. Otherwise, the name is the same as the instance name defined in the service template attribute.
SLAName	String	The SLA name for the parent service template.
isEnable	Boolean	Specifies whether the auto-population rule is enabled. The valid values are `true` to enable the rule and `false` to disable the rule. As auto-population rules affect performance, you must only have the rule enabled when it is required.
RelationshipRule	String	Specifies the rule for setting relationships if a new auto-population parent instance is detected for a previously auto-populated child instance. The valid values are as follows: • `"ADD"` creates a new parent. • `"NOTHING"` ignores the new parent. • `"SWITCH"` replaces the existing parent with the new parent service instance.

Table 3. addToAutoPopulationRule (continued)

Parameter	Format	Description
filter	String	(Optional) Enter an expression in this optional field if you want to apply additional conditions when a service instance is auto-created. For example, if you only want an instance created if the **AlertKey** field has a value, type `AlertKey<> ''` (AlertKey field value not equal to empty string) in this field.
StatusRuleName	String	The name of the incoming-status rule used to determine the status of the service instance. The default attribute name is `RawAttribute`.
RuleName	String	The name of the auto-population rule.
instanceNamePolicy	String	Optional. The policy used to determine the service name (`Autopop_InstanceName_`*Template*`_0_`*Rule*). This parameter is in the *Custom Auto-population Rule* window.
instanceNamePolicyScript	String	Optional. The full text of the policy you created for this rule. This parameter is in the *Custom Auto-population Rule* window.
dataTypeForInstanceNamePolicy	String	Optional. The data type used in the instance name policy. This parameter is in the *Custom Auto-population Rule* window.
queryForInstanceNamePolicy	String	Optional. The query used for the generic instance name policy. This parameter is in the *Custom Auto-population Rule* window.
attributeCreationPolicy	String	Optional. The policy used to create additional service attributes. The default for a template with no additional attributes is: `Autopop_AttributeCreation_`*Template* `_null_null` This parameter is in the *Custom Auto-population Rule* window.
attributeCreationPolicyScript	String	Optional. The full text of the policy you created for this rule. This parameter is in the **Custom Auto-population Rule** window.
attributeExpressions	String	Optional. The expressions for the additional attributes (properties). This parameter is in the **Custom Auto-population Rule** window.
tagNames	String	Optional. The names of the service template or templates you want assigned to the discovered services. If you have multiple templates assigned, separate each template name with three colons (`:::`).

Table 3. addToAutoPopulationRule (continued)		
Parameter	**Format**	**Description**
maintenanceWindow	String	Optional. The maintenance window you want to use for the new service instances.
groupName	String	Optional. The service group name you want for the new service. This parameter is in the *Custom Auto-population Rule* window.
dataFeed	String	Optional. The data feed for the generic-policy instance name query. This parameter is in the *Custom Auto-population Rule* window.

Example

The following example configures an auto-population rule with the following parent-child relationships:

1. The root service template for the rule is `OS_DBFARM`.
2. Null. Only used when you use a custom policy to determine the service name.
3. The parent service template is `OS_REGION`.
4. The instance name expression uses the value in the customer and region fields for the service instance name (same as display name).
5. Blank. Only used when you use a custom policy to determine the additional service attributes.
6. The SLA name is `Standard`.
7. The rule is enabled with the `true` value.
8. The rule automatically adds a relationship to a newly created parent service instance with the `ADD` value.
9. The filter expression is set to `true`.
10. The name of the auto-population rule is `DBAutopopHigh`.
11. Null. Only used when you use a custom policy to determine the service name.
12. Null. Only used when you use a custom policy to determine the service name.
13. Null. Only used when you use a custom policy to determine the service name.
14. Null. Only used when you use a custom policy to determine the service name.
15. The default attribute-creation policy name for an auto-population rule with no additional attributes (properties).
16. Null. Only used when you use a custom policy to determine the additional service attributes.
17. Blank. Only used when you use a custom policy to determine the additional service attributes.
18. Blank. Only used when you use a custom policy to determine the additional service attributes.
19. Null. No maintenance window is specified.
20. Blank. Only used when you use an expression to assign new services to a group.

```
[1]   addToAutoPopulationRule("OS_DBFARM",
[2]   null,
[3]   "OS_REGION",
[4]   "customer+region",
[5]   "",
[6]   "Standard",
[7]   true,
[8]   "ADD",
[9]   "true",
[10]   "DBAutopopHigh",
[11]   null,
[12]   null,
[13]   null,
```

```
[14]   null,
[15]   "Autopop_AttributeCreation_OS_DBFARM_null_null",
[16]   null,
[17]   "",
[18]   "",
[19]   null,
[20]   null
[21]   );
```

createAutoPopulationRule

Use the CreateAutoPoplulationRule function to configure a new rule that creates new service instances based on events from Netcool/OMNIbus. These are the same parameters you specify after you click the **Auto-population Rules** button in the **Templates** tab of the **Service Navigation** panel in the IBM Tivoli Business Service Manager GUI. See the TBSM *Service Configuration Guide* for more information about auto-population rules.

Syntax

createAutoPopulationRule("*RootServiceTemplateName*",
"*DisplayNameExpression*",
"*InstanceNameExpression*",
"*SlaName*",
EnableRule *true,false*,
Relationshiprule *"ADD","NOTHING","SWITCH"*,
Filter *"true","false"*
"*StatusRuleName*"
"*RuleName*"
"*instanceNamePolicy*",
"*instanceNamePolicyScript*,
"*dataTypeForInstanceNamePolicy*",
"*queryForInstanceNamePolicy*",
"*attributeCreationPolicy*",
"*attributeCreationPolicyScript*",
"*attributeExpressions*",
"*tagNames*",
"*maintenanceWindow*",
"*groupName*",
"*dataFeed*");

Parameter

This function has the following parameters.

Table 4. createAutPopulationRule Parameters

Parameter	Format	Description
RootServiceTemplateName	String	The name of template where you want to create an auto-population rule.
DisplayNameExpression	String	Optional. If you want a display name other than the default name defined in the event-based attribute, enter an expression for this parameter. This expression must be in the Netcool/Impact expression language. Otherwise, the name is the same as the instance name defined in the service template attribute.
InstanceNameExpression	String	Enter an expression to define the instance name with this parameter. This expression must be in the Netcool/Impact expression language.

Table 4. createAutPopulationRule Parameters (continued)		
Parameter	**Format**	**Description**
SLAName	String	If you want an SLA applied to the new service instances, specify the name of the SLA with this parameter. The default SLA name is Standard.
Enable	Boolean	Specifies whether the auto-population rule is enabled. The valid values are `true` to enable the rule and `false` to disable the rule. As auto-population rules affect performance, you must only have the rule enabled when it is required.
RelationshipRule	String	Specifies the rule for setting relationships if a new auto-population parent instance is detected for a previously auto-populated child instance. The valid values are as follows: • `"ADD"` creates a new parent. • `"NOTHING"` ignores the new parent. • `"SWITCH"` replaces the existing parent with the new parent service instance.
Filter	String	Optional. Enter an expression in this optional field if you want to apply additional conditions when a service instance is auto-created. For example, if you only want an instance created if the **AlertKey** field has a value, type `AlertKey<> "` (AlertKey field value not equal to empty string) in this field.
StatusRuleName	String	The name of the incoming-status rule used to determine the status of the service instance. The default attribute name is `RawAttribute`.
RuleName	String	The name of the auto-population rule.
instanceNamePolicy	String	Optional. The policy used to determine the service name (`Autopop_InstanceName_`*Template*`_0_`*Rule*). This parameter is in the *Custom Auto-population Rule* window.
instanceNamePolicyScript	String	Optional. The full text of the policy you created for this rule. This parameter is in the *Custom Auto-population Rule* window.
dataTypeForInstanceNamePolicy	String	Optional. The data type used in the instance name policy. This parameter is in the *Custom Auto-population Rule* window.
queryForInstanceNamePolicy	String	Optional. The query used for the generic instance name policy. This parameter is in the *Custom Auto-population Rule* window.

Table 4. createAutPopulationRule Parameters (continued)		
Parameter	**Format**	**Description**
attributeCreationPolicy	String	Optional. The policy used to create additional service attributes. The default for a template with no additional attributes is: `Autopop_AttributeCreation_`*Template* `_null_null` This parameter is in the *Custom Auto-population Rule* window.
attributeCreationPolicyScript	String	Optional. The full text of the policy you created for this rule. This parameter is in the **Custom Auto-population Rule** window.
attributeExpressions	String	Optional. The expressions for the additional attributes (properties). This parameter is in the **Custom Auto-population Rule** window.
tagNames	String	Optional. The additional attribute (property) name for each attribute expression. This parameter is in the *Custom Auto-population Rule* window.
maintenanceWindow	String	Optional. The maintenance window you want to use for the new service instances.
groupName	String	Optional. The service group name you want for the new service. This parameter is in the *Custom Auto-population Rule* window.
dataFeed	String	Optional. The data feed for the generic-policy instance name query. This parameter is in the *Custom Auto-population Rule* window.

Note: Be careful typing the field names for the name expressions such as *InstanceNameExpression*. Spelling and case errors cause your rule to malfunction. Some data sources, such as MS-SQL have case-sensitive field names. Run a query against your data source to make sure that the spelling and case are correct for the field name.

Example

The following example creates an auto-population rule with the following attributes:

1. The root service template for the rule is `OS_DBFARM`.
2. The display name expression is blank.
3. The instance name expression uses the values in the `lowlevelserviceid`, `customer`, and `region` fields for the service instance name (same as display name).
4. The SLA name is `Standard`.
5. The rule is enabled with the `true` value.
6. The rule automatically adds a relationship to a newly created parent service instance with the `ADD` value.
7. The filter expression is set to `true`.
8. The name of the source incoming-status rule is `HighDBTickets`.
9. The name of the auto-population rule is `DBAutopopHigh`.

10. Null. Only used when you use a custom policy to determine the service name.
11. Null. Only used when you use a custom policy to determine the service name.
12. Null. Only used when you use a custom policy to determine the service name.
13. Null. Only used when you use a custom policy to determine the service name.
14. The default attribute-creation policy name for an auto-population rule with no additional attributes (properties).
15. Null. Only used when you use a custom policy to determine the additional service attributes.
16. Blank. Only used when you use a custom policy to determine the additional service attributes.
17. Blank. Only used when you use a custom policy to determine the additional service attributes.
18. No maintenance window is specified.
19. Blank. Only used when you use an expression to assign new services to a group.
20. The `RawAttributeFeed` value specifies the default data feed for the incoming-status rule.

```
[1]   createAutoPopulationRule("OS_DBFARM",
[2]   "",
[3]   "lowlevelserviceid+customer+region",
[4]   "Standard",
[5]   true,
[6]   "ADD",
[7]   "true",
[8]   "HighDBTickets",
[9]   "DBAutopopHigh",
[10]   null,
[11]   null,
[12]   null,
[13]   null,
[14]   "Autopop_AttributeCreation_OS_DBFARM_null_null",
[15]   null,
[16]   "",
[17]   "",
[18]   null,
[19]   null,
[20]   "RawAttributeFeed"
[21]   );
```

Data source and data fetcher commands

Use the various data source and data fetcher commands to create a data source for IBM Tivoli Business Service Manager as well as create and maintain data fetchers.

You can configure the database connection information TBSM needs to connect to a given SQL data source and you can set up queries with data fetchers. See the IBM TBSM *Service Configuration Guide* for more information about configuring data sources and data fetchers.

clearDataFetcherCache

Use the `clearDataFetcherCache` function to clear the data cache in a given data fetcher.

```
public void clearDataFetcherCache(String name)
```

Syntax

This function has the following syntax:

```
clearDataFetcherCache("DataFetcherName");
```

Parameter

The only parameter for this function is the name of the data fetcher that you want to clear.

Example

In this example, the data cache for a data fetcher named `RegionalTickets` is cleared.

```
clearDataFetcherCache("RegionalTickets");
```

createDataFetcher

Use the `createDataFetcher` function to create a data fetcher for IBM Tivoli Business Service Manager.

Syntax

This function has the following syntax:

`createDataFetcher("`*name*`",`
minPollingInterval`,`
maxPollingInterval`,`
pollingMultiplier`,`
pollingHour`,`
pollingMin`,`
`"`*username*`",`
`"`*datasource*`",`
`"`*query*`",`
`disableQuery` *true false*`,`
`useExpression` *true false*`,`
`"`*expression*`")`

Parameters

The parameters for this function match the parameters for the **New Data Fetcher** tab in the TBSM GUI. This function has the following parameters:

Table 5. `createDataFetcher` Function Parameters

Parameter	Format	Description
name	String	The name of the data fetcher you want to create.
minPollingInterval	Long Integer	The minimum interval between data fetches. If you want to run the fetch at a specific time each day, enter a value of `-1` to disable this parameter.
maxPollingInterval	Long Integer	The maximum interval between data fetches. If you want to run the fetch at a specific time each day, enter a value of `-1` to disable this parameter.
pollingMultiplier	Integer	The polling multiplier for the polling interval. If you want to run the fetch at a specific time each day, enter the default value of `5`.
pollingHour	Integer	If you want to run the data fetcher once a day, enter the hour you want to run the query. The hour must be in 24 hour format.
pollingMin	Integer	If you want to run the data fetcher once a day, enter the minute you want to run the query.
username	String	The user name needed to access the database.
datasource	String	The data source where the database tables for the query reside.

Table 5. `createDataFetcher` Function Parameters (continued)		
Parameter	**Format**	**Description**
Query	String	The SQL query to execute against the data source.
disableQuery	Boolean	To disable the query, enter `true`. To enable the query, enter `false`.
UseExpression	Boolean	If you want to use an expression to limit the rows returned by the Data Fetcher, improving the efficiency of the fetcher processing. If you add an expression to the query, the last record from the previously fetched data will be used to build the expression. The expression is added to the fetcher query as an additional filtering clause that limits the rows returned by the fetcher. Enter `true`. If you want to check all the rows for updates, enter `false`.
Expression	String	The expression used to limit the rows returned by the Data Fetcher, improving the efficiency of the fetcher processing. Use the expression language to create this expression. For more information about using expressions, see the TBSM *Customization Guide*.

Example

This example creates a data fetcher named `RegionalTickets` that is run everyday at 12:00 pm.

```
[1]   createDataFetcher("RegionalTickets",
[2]     -1,
[3]     -1,
[4]     5,
[5]     12,
[6]     0,
[7]     "TicketDataSource",
[8]     "select customer,highlevelserviceid,lowlevelserviceid,region,severity,
count(*) as OpenTickets from tickets where status= 'Open' AND severity >= 4 group by
customer,highlevelserviceid,lowlevelserviceid,region,severity
 order by OpenTickets desc",
[9]     false,
[10]     false,
[11]     ""
[12]     ;
```

createDataSource

Use the `createDataSource` function to create a data source for IBM Tivoli Business Service Manager.

Syntax

This function has the following syntax:

`createDataSource("`*DataSourceName*`",`
`"`*DataSourcetype*`",`
`"`*primaryHost*`",`
primaryPort`,`
`"`*backupHost*`",`
backupPort`,`
`"`*username*`",`
`"`*password*`",`
`"`*primaryDatabase*`",`

" *backupDatabase*" ,
"*<failOverPolicy>*" ,
"*<enableCustomUrl>*" ,
"*primaryInformixServer*" ,
"*backupInformixServer*" ,
"*db2JDBCDriverType*" ,
"*<passwordEncrypted>*") ;

Note: Tivoli Business Service Manager supplies the following database JDBC drivers with this release:

- DB2
- HSQL
- Informix
- ObjectServer

If you want to use other databases as data sources, you need to obtain these drivers from the database manufacturer and copy them to your TBSM host. These files are typically provided by the database vendor with the database or the database client package. For example, you can find the Oracle file on your Oracle host system or as part of your Oracle client installation. For more information, see the *TBSM Installation Guide*.

Parameters

The parameters for this function match the parameters for the **New Data Source** tab in the TBSM GUI. This function has the following parameters:

Table 6. `createDataSource` function parameters

Parameter	Format	Description
DataSourceName	String	The name of the data source you want to create.
DataSourcetype	String	The type of data source you want to create. The valid values are: • ObjectServer • MySQL • Oracle • Sybase • MS-SQL • Postgres • DB2 • Informix®
primaryHost	String	The name or IP address of the host system for the primary data source.
primaryPort	Integer	The port number for the primary data source.
backupHost	String	Optional. The name or IP address of the host system for the backup data source.
backupPort	Integer	Optional. The port number for the backup data source.
username	String	The user name needed to access the database.
password	String	The password needed to access the database.

Table 6. `createDataSource` function parameters (continued)		
Parameter	**Format**	**Description**
primaryDatabase	String	For SQL data sources, the database name for the primary data source. For ObjectServer data sources, enter " ". For Oracle data sources, this field should contain the SID of the database IF enableCustomUrl is false. If enableCustomUrl is true, then it should contain the Oracle connection URL or RAC URL. Please see: http://www.ibm.com/support/knowledgecenter/SSSHYH_6.1.1.3/com.ibm.netcoolimpact.doc/user/rac_cluster_support.html?lang=en
backupDatabase	String	Optional. For SQL data sources, the database name for the backup data source. For ObjectServer data sources, enter " ".
<failoverPolicy>	String	If no backup data source is available, specify `Disabled`. To fail over to a backup data source, specify `Failover`. To ensure that the primary data source takes over from the backup data source when it becomes available again, specify `Failback`. The options are Disabled, Failover, and Failback.
<enableCustomURL>	Boolean	Indicates if the primaryDatabase field should be used as a custom URL or SID (for Oracle datasource only). When its value is true, the primaryDatabase field will be used as a custom URL connection. If it is false, it will be used as SID. For non-Oracle datasources, please set it to false.
primaryInformixServer	String	The name of primary server (for Informix only).
backupInformixServer	String	The name of backup server (for Informix only).
db2JDBCDriverType	Integer	The JDBC driver type: 2 for type-2, 4 for type-4 (for DB2 only).
<passwordEncrypted>	Boolean	Indicates whether password is encrypted or not.

Example

This example creates a data source called `TicketDataSource`:

```
createDataSource("TicketDataSource",
"DB2",
"tickethost.abc.com",
50000,
"",
50000,
```

```
"dbuser",
"dbpassword",
"ticketdb",
"",
"DISABLED",
false,
"",
"",
0,
false);
```

onDemandFetch

Use the onDemandFetch function to run the query for a data fetcher immediately.

Syntax

This function has the following syntax:

```
onDemandFetch("DataFetcherName", runInThread true false);
```

Parameters

This function has the following parameters:

Table 7. onDemandFetch Function Parameters

Parameter	Format	Description
DataFetcherName	String	The name of the data fetcher you want to run on demand.
runInThread	Boolean	Enter `true` to have the fetcher run in its own thread. If `false` is specified, the function does not return until the fetching is complete..

Example

In this example, an on-demand query is executed for a data fetcher named `RegionalTickets`:

```
onDemandFetch("RegionalTickets", true);
```

removeDataFetcher

Use the `removeDataFetcher` function to remove a data fetcher from IBM Tivoli Business Service Manager.

Syntax

This function has the following syntax:

`removeDataFetcher("`*DataFetcherName*`");`

Parameter

The only parameter for this function is the name of the data fetcher you want to remove.

Example

In this example, the data fetcher named `RegionalTickets` is removed.

```
removeDataFetcher("RegionalTickets");
```

clearAllESDAsForTag

Use the `clearAllESDAsForTag` function to clear all the ESDAs for a given tag.

Syntax

This function has the following syntax:

```
clearAllESDAsForTag("tagName");
```

Parameters

This function has the following parameters.

Table 8. `clearAllESDAsForTag` parameters

Parameter	Format	Description
tagName	String	Tag from which to clear all ESDAs.

Example

The following example clears all the ESDAs for the tag `ESDA_Hosts`.

```
radshell> clearAllESDAsForTag("ESDA_Hosts");
```

createMigrated42xDataSource

Use the `createMigrated42xDataSource` function to create a datasource from a migrated 42x datasource.

Syntax

This function has the following syntax:

```
createMigrated42xDataSource("name",
"type",
"primaryHost",
primaryPort,
"backupHost",
backupPort,
"username",
"password",
"primaryDatabase",
"backupDatabase",
disableFailOver,
"primaryInformixServer",
"backupInformixServer",
db2JDBCDriverType,
passwordEncrypted);
```

Parameters

This function has the following parameters.

Table 9. createMigrated42xDataSource parameters		
Parameter	**Format**	**Description**
name	String	Datasource name. This must not exceed 50 characters and must not contain spaces nor any of the following characters: = : \ " % & ! $
type	String	Type of datasource. This parameter takes the following values: • `DataBrowserIfc.OBJECTSERVER_SOURCE` • `DataBrowserIfc.SYBASE_SOURCE` • `DataBrowserIfc.ORACLE_SOURCE` • `DataBrowserIfc.POSTGRES_SOURCE` • `DataBrowserIfc.MYSQL_SOURCE` • `DataBrowserIfc.MSSQL_SOURCE` • `DataBrowserIfc.INFORMIX_SOURCE` • `DataBrowserIfc.DB2_SOURCE`
primaryHost	String	Primary host for this datasource.
primaryPort	Integer	Primary port for this datasource.
backupHost	String	Backup host for this datasource.
backupPort	Integer	Backup port for this datasource.
username	String	Username for this datasource.
password	String	Password for this datasource.
primaryDatabase	String	Primary database for this datasource.
backupDatabase	String	Backup database for this datasource.
disableFailOver	Boolean	Indicates whether failover is disabled.
primaryInformixServer	String	Name of the primary server (for Informix only).
backupInformixServer	String	Name of the backup server (for Informix only).

Table 9. createMigrated42xDataSource parameters (continued)		
Parameter	**Format**	**Description**
db2JDBCDriverType	Integer	The driver type. For JDBC driver, specify type 2. For DB2, specify type 4.
passwordEncrypted	Boolean	Indicates whether the password is encryped.

Example

The following example creates a datasource from a migrated 42x datasource using the data indicated.

```
radshell> createMigrated42xDataSource(Webfarm_DB2, DataBrowserIfc.DB2_SOURCE, Webfarm_host, 22,
Webfarm_backup_host, 22, DB2_User, DB2_User, Webfarm_primary, Webfarm_backup,  "", "", 4,
false);
```

createPolicyDataFetcher

Use the createPolicyDataFetcher function to create a policy-based data fetcher.

Syntax

This function has the following syntax:

```
createPolicyDataFetcher("name",
minPollingInterval,
maxPollingInterval,
pollingMultiplier,
pollingHour,
pollingMin,
"policyName",
"columnNames",
disabled,
"policyScript");
```

Parameters

This function has the following parameters.

Table 10. createPolicyDataFetcher parameters		
Parameter	**Format**	**Description**
name	String	Name of the data fetcher.
minPollingInterval	Integer	Minimum polling interval. Set this parameter to -1 if you are not using it with this data fetcher.
maxPollingInterval	Integer	Maximum polling interval. Set this parameter to -1 if you are not using it with this data fetcher.
pollingMultiplier	Integer	Polling interval multiplier.
pollingHour	Integer	Polling hour in 24 hour format. Set this parameter to -1 if you are not using it with this data fetcher.

Table 10. createPolicyDataFetcher parameters (continued)		
Parameter	**Format**	**Description**
pollingMin	Integer	Polling minute. Set this parameter to -1 if you are not using it with this data fetcher.
policyName	String	Name of the policy to execute for data fetching.
columnNames	String	Names of the columns with ; used as the delimiter. A sample columnNames has the following format: Column1#Integer;Column2#Boolean;Column3#Long;Column4#Long
disabled	Boolean	Indicates whether this dbpoller is disabled.
policyScript	String	A string containing the actual policy text.

Example

The following example creates a policy based data fetcher using the data indicated.

```
radshell> createPolicyDataFetcher("TestPolicyFetcher", 30, 300, 5, 0,
0,"Test_DataPolicyFetcher", "Column1#Integer;Column2#String", false, "Policy_Script");
```

disableDataFetcher

Use the disableDataFetcher function to stop or start the specified data fetcher.

Syntax

This function has the following syntax:

```
disableDataFetcher("dataFetcherName",
set);
```

Parameters

This function has the following parameters.

Table 11. disableDataFetcher parameters		
Parameter	**Format**	**Description**
dataFetcherName	String	Name of the data fetcher to stop or start.
set	Boolean	Indicates whether to disable the data fetcher specified. This parameter has the following valid values: true: Stops the date fetcher. false: Starts the date fetcher.

Example

The following example stops the data fetcher called `MyDB2Fetcher`.

```
radshell> disableDataFetcher("MyDB2Fetcher",true);
```

listITMPolicyDataFetchers

Use the `listITMPolicyDataFetchers` function to list the ITM policy based data fetchers.

Syntax

This function has the following syntax:

```
listITMPolicyDataFetchers();
```

Parameters

This function has no parameters.

Example

The following example lists the ITM policy based data fetchers.

```
radshell> listITMPolicyDataFetchers();
```

setDataFetcher

Use the `setDataFetcher` function to set the data fetcher parameters.

Syntax

This function has the following syntax:

```
setDataFetcher("name",
minPollingInterval,
maxPollingInterval,
pollingMultiplier,
pollingHour,
pollingMin,
"datasource",
"query",
disabled,
useExpression,
"expression");
```

Parameters

This function has the following parameters.

Table 12. `setDataFetcher` parameters

Parameter	Format	Description
name	String	Data fetcher name.
minPollingInterval	Integer	Minimum polling interval. Set this parameter to -1 if you are not using it with this data fetcher.

Table 12. `setDataFetcher` parameters (continued)		
Parameter	**Format**	**Description**
maxPollingInterval	Integer	Maximum polling interval. Set this parameter to `-1` if you are not using it with this data fetcher.
pollingMultiplier	Integer	Polling interval multiplier.
pollingHour	Integer	Polling hour in 24 hour format. Set this parameter to `-1` if you are not using it with this data fetcher.
pollingMin	Integer	Polling minutes. Set this parameter to `-1` if you are not using it with this data fetcher.
datasource	String	`dbpoller` datasource.
query	String	`dbpoller` query to use.
disabled	Boolean	Indicates whether this `dbpoller` is disabled.
useExpression	Boolean	Indicates whether to use an expression in the query.
parameterName	String	Expression string to be appended to the end of the query.

Example

The following example sets the parameters for the data fetcher `MyDataFetcher`.

```
radshell> setDataFetcher("MyDataFetcher",  30000,  30000, 5,  10, 15,"MyDataSource",  "select *
from mytable" , false, false, null);
```

testConnection

Use the `testConnection` function to test the database connection for a given datasource.

Syntax

This function has the following syntax:

```
testConnection("datasourceName"
serverType);
```

Parameters

This function has the following parameters.

Table 13. `testConnection` parameters		
Parameter	**Format**	**Description**
datasourceName	String	Name of the datasource.

Table 13. testConnection parameters (continued)		
Parameter	**Format**	**Description**
serverType	Boolean	Set this parameter to `true` for the primary server and `false` for the backup server.

Example

The following example tests the connection of the primary server to the MyOracleDSA datasource.

```
radshell> testConnection("MyOracleDSA",true);
```

The following example tests the connection of the backup server to the MyOracleDSA datasource.

```
radshell> testConnection("MyOracleDSA",false);
```

updateITMPolicyDataFetchers

Use the updateITMPolicyDataFetchers function to update the ITM policy-based data fetchers, replacing the ITM server contact information.

Syntax

This function has the following syntax:

```
updateITMPolicyDataFetchers("datafetcher",
"protocol",
"web_service_host_name",
"web_service_port_number",
"web_service_name",
"user_id",
"user_id_password_encrypted",
"single_sign_on");
```

Parameters

This function has the following parameters.

Table 14. updateITMPolicyDataFetchers parameters		
Parameter	**Format**	**Description**
datafetcher	String	Name of the data fetcher.
protocol	String	Protocol used by the data fetcher. Valid values are HTTP and HTTPS.
web_service_host_name	String	Host name or IP address of the TEMS/TEPS sever where the ITM Charting Web Service was installed on.
web_service_port_number	String	Port number the ITM Charting Web Service is listening on.
web_service_name	String	ITM Charting Web Service name.
user_id	String	ITM user ID used to connect to the ITM Charting Web Service.
user_id_password_encrypted	String	Encrypted password for the user ID.

Table 14. updateITMPolicyDataFetchers parameters (continued)		
Parameter	**Format**	**Description**
single_sign_on	String	Indicates whether single signon is supported.

Export commands

Use the RAD shell export commands to copy an existing service configuration to a RAD shell script. You then can import the RAD shell script into another TBSM host.

These commands create output files in the $TBSM_HOME/export directory.

export

Use the **export** function to export your IBM Tivoli Business Service Manager service configuration to a file. When you use the **export** function, all of your service templates, services, data fetchers, and data sources are written to file named export.radsh in the $TBSM_HOME/export directory.

Syntax

This function has the following syntax:

```
export();
```

Example

The **export** function is only used to export an entire service configuration as shown in this example:

```
export();
```

exportCSV

Use the **exportCSV** function to export your IBM Tivoli Business Service Manager service instances and dependencies to a comma-separated value (.csv) file.

When you use the **exportCSV** function, all of your service instances, and the service-instance dependencies are written to two files: radexport_instances.csv and radexport_dependencies.csv in the $TBSM_HOME/export directory. The radexport_instances.csv file contains all the service-instance attributes. The radexport_dependencies.csv file contains the service dependencies for each service instance in your service configuration.

Syntax

This function has the following syntax:

```
exportCSV();
```

Example

This function is only used to export an entire service configuration as shown in this example:

```
exportCSV();
```

exportFromStartingInstance

Use the `exportFromStartingInstance` function to export a service instance and all the service instances and service templates related to the service instance. You export the service instance, its service template, and all of its child services and service templates.

You must use the **Service Name** field value, not the **Display Name** field value, when you use this function to export service instance data. When you use the **exportFromStartingInstance** function, the service data is written to a file named `export.radsh` in the `$TBSM_HOME/export` directory.

Syntax

This function has the following syntax:

```
exportFromStartingInstance("ServiceName");
```

Parameter

The **ServiceName** parameter is the value of the **Service Name** field on the **Edit Service** tab in the IBM Tivoli Business Service Manager GUI.

Example

In this example, the function exports the service data for a service instance called `ABCOnlineBanking`.

```
exportFromStartingInstance("ABCOnlineBanking");
```

exportMeta

Use the **exportMeta** function to export your IBM Tivoli Business Service Manager configuration to a file. When you use the **exportMeta** function, all of your templates, data fetchers, data sources, and auto-population rules are written to a file named `exportMeta.radsh` in the `$TBSM_HOME/export` directory. This function is like the **export** function, except that service instances and service dependencies are not exported by the **exportMeta** function.

Syntax

This function has the following syntax:

```
exportMeta();
```

Example

The **exportMeta** function is used to export an entire configuration (not including service instances and service dependencies) as shown in this example.

```
exportMeta();
```

exportDataFetchers

Use the `exportDataFetchers` function to export a specific data fetcher.

Syntax

This function has the following syntax:

```
exportDataFetchers("df",
"pw");
```

Parameters

This function has the following parameters.

Table 15. exportDataFetchers parameters		
Parameter	**Format**	**Description**
df	String	Name of the data fetcher to export.
pw	String	The print writer to use.

exportDataFetchersForImport

Use the `exportDataFetchersForImport` function to export selected user created datasources for import.

Syntax

This function has the following syntax:

```
exportDataFetchersForImport("regexPattern",
"exportDir");
```

Parameters

This function has the following parameters.

Table 16. exportDataFetchersForImport parameters		
Parameter	**Format**	**Description**
regexPattern	String	A regular expression pattern specifying which datasources to export.
exportDir	String	Fully qualified directory in which to put the exported file.

exportDataSource

Use the `exportDataSource` function to export all configuration details for a given datasource.

Note: ESDA and data fetchers depend on this function.

Syntax

This function has the following syntax:

```
exportDataSource("datasource",
"pw");
```

Parameters

This function has the following parameters.

Table 17. exportDataSource parameters		
Parameter	**Format**	**Description**
datasource	String	The datasource from which to export configuration details.
pw	String	The printer writer to use.

exportDataSourcesForImport

Use the `exportDataSourcesForImport` function to export selected user created datasources.

Syntax

This function has the following syntax:

```
exportDataSourcesForImport("regexPattern",
"exportDir");
```

Parameters

This function has the following parameters.

Table 18. `exportDataSourcesForImport` parameters

Parameter	Format	Description
regexPattern	String	A regular expression pattern specifying which datasources to export.
exportDir	String	Directory in which to write the export file.

exportForMigration

Use the `exportForMigration` function to export for migration the IDs of instances, templates and SLA rules so that they will work with existing security manager configuration and SLA tracking data and canvases.

Syntax

This function has the following syntax:

```
exportForMigration("host",
"port",
"remoteHOME",
"reomteUser",
"remotePassword");
```

Parameters

This function has the following parameters.

Table 19. `exportForMigration` parameters

Parameter	Format	Description
host	String	The host RAD server to export from.
port	String	HTTP port of the host RAD server.
remoteHOME	String	Name of the host machine. If reading from the local server, set this parameter to `null`.
reomteUser	String	Username to log on to the host with. If reading from the local server, set this parameter to `null`.

Table 19. exportForMigration parameters (continued)		
Parameter	**Format**	**Description**
remotePassword	String	Password to use. If reading from the local server, set this parameter to `null`.

exportFromStartingInstanceNoMeta

Use the `exportFromStartingInstanceNoMeta` function to export an instance to a script file along with a specified number of levels of its dependencies.

The name of the script file is `tbsm/export/export_<instanceName>.radsh`.

Note: ESDA instances and non-persistent autpopulated instances are not exported as children.

Syntax

This function has the following syntax:

```
exportFromStartingInstanceNoMeta("instanceName",
levels);
```

Parameters

This function has the following parameters.

Table 20. exportFromStartingInstanceNoMeta parameters		
Parameter	**Format**	**Description**
instanceName	String	Name of instance to export.
levels	Integer	Levels of the instance's dependencies to export. For example, if you set levels to 3, the function exports the instance itself, its children and its grandchildren.

exportInstanceNoMeta

Use the `exportInstanceNoMeta` function to export an instance (without its dependencies) to a script file.

The name of the script file is `tbsm/export/export_<instanceName>.radsh`.

Syntax

This function has the following syntax:

```
exportInstanceNoMeta("instanceName");
```

Parameters

This function has the following parameters.

Table 21. exportInstanceNoMeta parameters		
Parameter	**Format**	**Description**
instanceName	String	Name of the instance to export.

exportInstanceRelationshipAttributes

Use the `exportInstanceRelationshipAttributes` function to export the relationship attributes of an instance.

Syntax

This function has the following syntax:

```
exportInstanceRelationshipAttributes("instanceName",
"pw");
```

Parameters

This function has the following parameters.

Table 22. `exportInstanceRelationshipAttributes` parameters

Parameter	Format	Description
instanceName	String	Name of the instance to export.
pw	String	The Printer Writer to use.

exportMaintSchedules

Use the `exportMaintSchedules` function to export the full RAD configuration to a file called `export.radsh`.

Syntax

This function has the following syntax:

```
exportMaintSchedules("pw",
"remoteHost",
"remotePort",
"remoteHOME",
"remoteUser",
"remotePassword");
```

Parameters

This function has the following parameters.

Table 23. `exportMaintSchedules` parameters

Parameter	Format	Description
pw	String	Print Writer with which to write.
remoteHost	String	The host RAD server to export from.
remotePort	String	HTTP port of the host RAD server.
remoteHOME	String	Name of the host machine. If reading from the local server, set this parameter to `null`.

Table 23. exportMaintSchedules parameters (continued)		
Parameter	**Format**	**Description**
remoteUser	String	Username to log on to the host with. If reading from the local server, set this parameter to `null`.
remotePassword	String	Password to use. If reading from the local server, set this parameter to `null`.

exportMetaWithArtifactMerge

Use the `exportMetaWithArtifactMerge` function to export your configuration to a file.

This function merges artifacts from the export with the existing values.

Syntax

This function has the following syntax:

```
exportMetaWithArtifactMerge();
```

Parameters

This function has no parameters.

Example

The following example exports your configuration to a file.

```
radshell> exportMetaWithArtifactMerge();
```

exportTemplateForReplace

Use the `exportTemplateForReplace` function to export all configuration for a specific template and its dependent templates to the file `exportTemplate_templateName.radsh`.

Syntax

This function has the following syntax:

```
exportTemplateForReplace("tagName",
"exportDir");
```

Parameters

This function has the following parameters.

Table 24. exportTemplateForReplace parameters		
Parameter	**Format**	**Description**
tagName	String	The name of the root template to export.
exportDir	String	Fully qualified directory in which to put the exported file.

exportTemplatesForImport

Use the `exportTemplatesForImport` function to export the configuration for the selected user created templates.

Syntax

This function has the following syntax:

```
exportTemplatesForImport("regexPattern",
"exportDir");
```

Parameters

This function has the following parameters.

Table 25. `exportTemplatesForImport` parameters

Parameter	Format	Description
regexPattern	String	A regular expression pattern specifying which templates to export.
exportDir	String	Fully qualified directory in which to put the exported file.

Maintenance schedule commands

Maintenance schedules allow you to account for scheduled downtime in your service models.

createAbsoluteTimeWindow

Use the `createAbsoluteTimeWindow` function to add a one-time window for maintenance schedules. Absolute time windows are maintenance periods with a set date and time and do not repeat on a regular basis. See the TBSM *Service Configuration Guide* for more information about maintenance schedules.

Syntax

```
createAbsoluteTimeWindow("absoluteTimeWindowName",
startMonth,
startDay,
startYear,
startHour,
startMinute,
endMonth,
endDay,
endYear,
endHour,
endMinute);
```

Parameters

This function has the following parameters:

Note: For this function, the months start with `0` for January. As a result use the numbers `0` through `11` to specify the months January through December.

Restriction: For months, hours and minutes, do not include a leading zero as part of the value. That is, for the number 1, **do not** enter 01, since this will cause the command to malfunction.. For example, to create a window which runs on August 8 from 8 am to 8:08 am, enter:

```
createAbsoluteTimeWindow("8 Aug 8AM - 8 Aug 8:08",
7,8,2011,8,0,7,8,2011,8,8);
```

Table 26. createAbsoluteTimeWindow Parameters

Parameter	Format	Description
absoluteTimeWindowName	String	The name of the absolute time window you want to create. You can add this time window to a schedule as described for the `createScheduleMaintenanceWindow` command..
startMonth	Integer	The month when you want the time window to start in number format. For example, enter 0 for January, 1 for February, and so on.
startDay	Integer	The day of the month when you want the time window to start.
startYear	Integer	The year when you want the time window to start. Enter in *YYYY* format (for example, 2004).
startHour	Integer	The hour of the day when you want the time window to start.
startMinute	Integer	The minute in the hour when you want the time window to start.
endMonth	Integer	The month when you want the time window to end in number format. For example, enter 0 for January, 1 for February, and so on.
endDay	Integer	The day of the month when you want the time window to end.
endYear	Integer	The year when you want the time window to end. Enter in *YYYY* format (for example, 2004).
endHour	Integer	The hour of the day when you want the time window to end.
endMinute	Integer	The minute in the hour when you want the time window to end.

Example

The following example creates an absolute time window for June 23, 2011 between 9:00 pm and 12:50 pm. The time window is specified as follows:

1. The time window name is `23 June 2011`.
2. The start month for the time window is June (5).
3. The start day of the month for the time window is (23.)
4. The start year for the time window is the 2011. As a result, the start date for the time window is set to June 23, 2011.
5. The start hour for the time window is 9:00 pm (21).

6. The start minute for the time window is the 00. As a result, the start time for the time window is set to 9:00 pm.
7. The end month for the time window is June (5).
8. The end day of the month for the time window is (23).
9. The end year for the time window is 2011.
10. The end hour for the time window is 10:00 pm (22).
11. The end minute for the time window is 50.

```
[1]   createAbsoluteTimeWindow("23 June 2011",
[2]                            5,
[3]                            23,
[4]                            2011,
[5]                            21,
[6]                            0,
[7]                            5,
[8]                            23,
[9]                            2011,
[10]                            22,
[11]                            50;)
```

createRecurringTimeWindow

Use the createRecurringTimeWindow function to add a recurring time window for maintenance schedules. Recurring time windows are maintenance periods that occur on a weekly basis. See the TBSM *Service Configuration Guide* for more information about maintenance schedules.

Syntax

createRecurringTimeWindow("*recurringTimeWindowName*",
dayOfWeek,
startHour,
startMinute,
endHour,
endMinute);

This second syntax allows a different start day and end day to be specified.

createRecurringTimeWindow(" *recurringTimeWindowName* ",
startDayOfWeek ,
endDayOfWeek ,
startHour ,
startMinute ,
endHour ,
endMinute);

Parameters

This function has the following parameters:

Restriction: For months, hours and minutes, do not include a leading zero as part of the value. That is, for the number 1, **do not** enter 01, since this will cause the command to malfunction.. For example, to create a window which runs on Monday from 8 am to 8:08 am, enter:

```
createRecurringTimeWindow("Mon 8AM - 8:08AM",2,2,8,0,8,8);
```

Table 27. createRecurringTimeWindow Parameters		
Parameter	**Format**	**Description**
recurringTimeWindowName	String	The name of the time window you want to create. You can add this time window to a schedule as described in “createScheduleMaintenanceWindow” on page 89.
dayOfWeek	Integer	The number representing the day of the week for the time window, where 1 represents Sunday and 7 represents Saturday.
startdayOfWeek	Integer	The number representing the day of the week for start of the time window, where 1 represents Sunday and 7 represents Saturday.
enddayOfWeek	Integer	The number representing the day of the week for end of the time window, where 1 represents Sunday and 7 represents Saturday.
startHour	Integer	The hour of the day when you want to the time window to start. Use 24-hour time. For example, 14 equals 2:00 pm.
startMinute	Integer	The minute when you want the time window to start.
endHour	Integer	The hour of the day when you want to the time window to end. Use 24-hour time. For example, 15 equals 3:00 pm.
endMinute	Integer	The minute when you want the time window to end.

Examples

The following example creates a recurring time window with the following characteristics:

1. The time window name is `Thursday 7:50 PM - 8:50 PM`
2. The day of the week is set to Thursday (5).
3. The start hour for the time window is 7:00 pm (19)
4. The start minute for the time window is the 50. As a result, the start time for the time window is set to 7:50 pm.
5. The end hour for the time window is 8:00 pm (20).
6. The end minute for the time window is the 50. As a result, the end time for the time window is set to 8:50 pm.

```
[1]   createRecurringTimeWindow("Thursday 7:50 PM - 8:50 PM",
[2]                             5,
[3]                             19,
[4]                             50,
[5]                             20,
[6]                             50;)
```

The following example creates a recurring time window with different start and end days:

1. The time window name is Mon7:50 PM - Wed 8:50 PM
2. The start day of the week is set to Monday (2).
3. The end day of the week is set to Wednesday (4).

4. The start hour for the time window is 7:00 pm (19)
5. The start minute for the time window is the 50. As a result, the start time for the time window is set to 7:50 pm.
6. The end hour for the time window is 8:00 pm (20).
7. The end minute for the time window is the 50. As a result, the end time for the time window is set to 8:50 pm.

```
1] createRecurringTimeWindow("Mon7:50 PM - Wed 8:50 PM",
[2] 2,
[3] 4,
[4] 19,
[5] 50,
[6] 20,
[7] 50;)
```

createScheduleMaintenanceWindow

Use the `createScheduleMaintenanceWindow` function to create maintenance schedules.

These are the same parameters you specify after you click the **New** button for maintenance schedules in the **Edit Service** panel of the IBM Tivoli Business Service Manager GUI. See the TBSM *Service Configuration Guide* for more information about maintenance schedules.

Syntax

```
createScheduleMaintenanceWindow("scheduleName",
new String[] {"timeWindows"});
```

Parameters

This function has the following parameters:

Table 28. `createScheduleMaintenanceWindow` Parameters

Parameter	Format	Description
scheduleName	String	The name of the schedule you want to create.
timeWindows	String Array	The time window or windows you want to add to the maintenance window. You must use an existing time window. See “createRecurringTimeWindow” on page 87 and “createAbsoluteTimeWindow” on page 85 for more information.

Example

The following example creates a time window called `WebFarm_Maintenance-1` with time windows on Thursday between 7:00 and 8:50 PM and from October 23 at 10:00 PM to October 24 at 10:01 pm.

```
[1]   createScheduleMaintenanceWindow("WebFarm_Maintenance-1",
[2]   new String[] {"Thursday 7:00 PM - 8:50 PM","23 Oct 2003 10:00 PM - 24
 Oct 2003 10:01 PM"});
```

createDailyDateTimeWindow

Use the `createDailyDateTimeWindow` function to construct a daily-date-time window.

Syntax

This function has the following syntax:

```
createDailyDateTimeWindow("dailyDateTimeWindowName",
"dateTimeWindowName",
"dailyTimeWindowName");
```

Parameters

This function has the following parameters.

Table 29. createDailyDateTimeWindow parameters

Parameter	Format	Description
dailyDateTimeWindowName	String	Name of the daily-date-time window to construct.
dateTimeWindowName	String	Name of the date-time window to use.
dailyTimeWindowName	String	Name of the daily-time window to use.

Example

The following example constructs a daily-date-time window called `31 May 2005 05:58 PM - 31 May 2005 06:01 PM` from the `30 April 2005 - 31 May 2005` date-time and `05:58 - 06:01 PM` daily-time windows.

```
radshell> createDailyDateTimeWindow("31 May 2005 05:58 PM - 31 May 2005 06:01 PM", "30 April
2005 - 31 May 2005", "05:58 - 06:01 PM");
```

createDailyTimeWindow

Use the `createDailyTimeWindow` function to create a daily-time window.

Syntax

This function has the following syntax:

```
createDailyTimeWindow("dailyTimeWindowName",
startHour,
startMinute,
endHour,
endMinute);
```

Parameters

This function has the following parameters.

Table 30. createDailyTimeWindow parameters

Parameter	Format	Description
dailyTimeWindowName	String	Name for the daily-time window.
startHour	Integer	Hour at which the daily time window starts.
startMinute	Integer	Minute at which the daily time window starts.
endHour	Integer	Hour at which the daily time window ends.
endMinute	Integer	Minute at which the daily time window ends.

Example

The following example creates a daily-time window called `05:58 - 06:01 PM` which is defined as running from `17:58` to `18:01` each day.

```
radshell> createDailyTimeWindow("05:58 - 06:01 PM",17,58,18,01);
```

createDateTimeWindow

Use the `createDateTimeWindow` function to create a date-time window.

Syntax

This function has the following syntax:

```
createDateTimeWindow("dateTimeWindowName",
startMonth,
startDay,
startYear,
endMonth,
endDay,
endYear);
```

Parameters

This function has the following parameters.

Table 31. `createDateTimeWindow` parameters

Parameter	Format	Description
dateTimeWindowName	String	Name for the date-time window.
startMonth	Integer	Month on which the date-time window starts.
startDay	Integer	Day on which the date-time window starts.
startYear	Integer	Year on which the date-time window starts.
endMonth	Integer	Month on which the date-time window ends.
endDay	Integer	Day on which the date-time window ends.
endYear	Integer	Year on which the date-time window ends.

Example

The following example creates a date-time window called `30 April 2005 - 31 May 2005` which is defined as running from `April 30,2005` to `May 31, 2005`

```
radshell> createDateTimeWindow("30 April 2005 - 31 May 2005", 4,30,2005,5,31,2005);
```

Service instance commands

Service instances represent actual pieces of hardware or software on a network that behave according to the rules defined by its service template or templates. Before you can add a service instance with the RAD shell, you must have a preexisting service template.

addGISCoordinatesForInstance

Use the `addGISCoordinatesForInstance` function to set GIS coordinates for a service instance.

These are the same parameters you specify in the **Additional** tab of the **Edit Service** panel in the IBM Tivoli Business Service Manager GUI.

Syntax

`addGISCoordinatesForInstance(("`*instanceName*`",`
`"`*longitude*`","`*latitude*`");`

Parameters

This function has the following parameters:

Table 32. addGISCoordinatesForInstance

Parameter	Format	Description
instanceName	String	The name of the service instance in the database.
longitude	String	The GIS longitude for the service instance.
latitude	String	The GIS latitude for the service instance.

Example

The following example sets the GIS coordinates for Berlin, Germany for a service instance called `webserver1`.

```
addGISCoordinatesForInstance("webserver1",
                             "13.1","52.45");
```

addHourlySLAPenaltyToInstance

Use the `addHourlySLAPenaltyToInstance` function to set an hourly SLA penalty for a service instance.

These are the same parameters you specify in the IBM Tivoli Business Service Manager GUI.

Syntax

`addHourlySLAPenaltyToInstance ("`*instanceName*`","`*hourlySLAPenalty*`");`

Parameters

This function has the following parameters:

Table 33. addHourlySLAPenaltyToInstance Parameters		
Parameter	**Format**	**Description**
instanceName	String	The name of the service instance in the database.
hourlySLAPenalty	String	The hourly dollar amount associated with an SLA violation.

Examples

The following examples show how to add an hourly penalty of $1000.

```
addHourlySLAPenaltyToInstance ("webserver1","1000");
```

addScheduleToInstance

Use the addScheduleToInstance function to assign a maintenance schedule to a service instance.

These are the same parameters you specify in the **Edit Service** panel of the IBM Tivoli Business Service Manager GUI. See the TBSM *Service Configuration Guide* for more information about maintenance schedules.

Syntax

addScheduleToInstance("*instanceName*", "*scheduleName*",scope);

Parameters

This function has the following parameters:

Table 34. addScheduleToInstance Parameters		
Parameter	**Format**	**Description**
instanceName	String	The name of the service instance in the database.
scheduleName	String	The name of the schedule you want to assign to the service instance. You must create the schedule you want to assign before using this function. See “createScheduleMaintenanceWindow” on page 89 for more information.
scope	Integer	Specifies the scope of schedule assignment. When set to 0 (default), the schedule is added to the specified service. When set to 1, the schedule is added to the service and all its child services. When set to 2, the schedule is added to the child services, but is not added to the service itself.

Examples

The following example shows how to add the schedule Sundaybackup to the webserver1 service instance.

```
addScheduleToInstance("webserver1", "Sundaybackup", 0);
```

The following example shows how to add the schedule Sundaybackup to the webserver1 service and all its child service instances.

```
addScheduleToInstance("webserver1", "Sundaybackup", 1);
```

The following example shows how to add the schedule Sundaybackup only to the child services of the webserver1 service. The schedule is not added the webserver1 service.

```
addScheduleToInstance("webserver1", "Sundaybackup", 2);
```

addServiceInstance

Use the addServiceInstance function to create a new service instance in the IBM Tivoli Business Service Manager database. You can only create a service instance for a preexisting service template.

Syntax

addServiceInstance("*templateName*", "*instanceName*", "*displayName*", "*description*", "*slaName*", String[] {"*rawEventFieldExprs*"}, String[] {"*rawEventFieldValues*"});

Parameters

This function has the following parameters:

Table 35. addServiceInstance parameters

Parameter	Format	Description
templateName	String	The name of the template or templates assigned to the service instance. If you want to assign more than one template to the new service instance, you must enter an array of templates as follows: `addServiceInstance(String[] {"template1","template2"}` The first template specified in the template names array is the Primary template.
instanceName	String	The name of the service instance in the database.
displayName	String	The name of the service instance as it appears in the user interface.
description	String	Optional. Description of the service instance as it appears in the **Service Instance Properties** panel in the user interface.

Table 35. addServiceInstance parameters (continued)		
Parameter	**Format**	**Description**
slaName	String	The name of the service-level agreement (SLA) for the service instance. If you have not configured an SLA for the instance, you must enter `Standard` for this parameter.
rawEventFieldExprs	String Array	Optional. An array of event field expressions that correspond to one or more event mapping rules defined in service templates associated with the service instance. It automatically determines how the event field expressions correspond to the event attributes defined in the templates.
rawEventFieldValues	String Array	Optional. An array of event field values that correspond to one or more event mapping rules defined in service templates associated with the service instance. It automatically determines how the event field values correspond to the event attributes defined in the templates.

Examples

The following examples show how to add a simple service instance without event attributes and a complex service instance with event attributes.

Simple Service Instance Example

The following example creates a new service instance with the following characteristics:

1. The template for the service instance is `Webserver`.
2. The service instance name is `webserver2`.
3. The description is `L-R Webserver`.
4. The SLA is `Standard`.

```
[1]   addServiceInstance ("Webserver",
[2]   "webserver2",
[3]   "L-R Webserver",
[4]   "Standard");
```

Complex Service Instance Example

The following example creates a new service instance with the following characteristics:

1. The primary template is `Webserver` and the second template is `Box`.
2. The service instance name is `webserver1`.
3. The display name is `Web Server 1`.
4. The description is `Web Server Prime`.
5. The SLA name is `Standard`.
6. The raw event field expressions set the **Location** and **Node** fields as the service instance identifier fields for the `webserver1` service. The & character is stripped from the value of the **Node** field.
7. Sets the array for the **Location** and **Node** field values that identify an event for the `webserver1` service instance. Lines 8, 9, 10, and 11 set four possible value combinations which can identify an event for the `webserver1` service instance.
8. On this line the values `ny` and `ws1` identify an event for the `webserver1` service.
9. On this line the values `ny.us` and `ws1` identify an event for the `webserver1` service.

10. On this line the values nyc and ws1 identify an event for the webserver1 service.
11. On this line the values nyc and ws1.micromuse.com identify an event for the webserver1 service.

```
[1]   addServiceInstance(new String[] {"WebServer", "Box"},
[2]      "webserver1",
[3]      "Web Server 1",
[4]      "Web Server Prime",
[5]      "Standard",
[6]       new String[] {"Location", "strip(Node, \"&\")"},
[7]       new String[][] {
[8]       new String[] {"ny", "ws1"},
[9]       new String[] {"ny.us","ws1"},
[10]       new String[] {"nyc","ws1"},
[11]       new String[] {"nyc","ws1.micromuse.com"}
[12]   });
```

addServiceInstanceDependency

Use the addServiceInstanceDependency function to add a child service instance to a parent service instance.

These are the same parameters that you specify in the **Edit Dependents** tab of the **Edit Service** panel in the IBM Tivoli Business Service Manager GUI.

Syntax

addServiceInstanceDependency("*parentInstanceName*", "*childInstanceName*");

Parameters

This function has the following parameters:

Table 36. addServiceInstancedependency Parameters

Parameter	Format	Description
ParentinstanceName	String	The name of the service instance in the database where you want to add a service dependency.
ChildinstanceName	String	The name of service instance you want to add as a dependent of the parent service instance.

Example

The following example adds the service instance called webserver1 as a dependent of the webfarm1 service instance.

```
addServiceInstanceDependency("webfarm1","webserver1");
```

clearStartingInstance

The clearStartingInstance function is included for compatibility with earlier versions of TBSM; this function is no longer supported. Use the clearStartingInstance function to clear the service that displays when a user logs in to IBM Tivoli Business Service Manager. You can also clear the starting service for a user group. When you clear the starting service, the user sees the default display according to their user or group privileges.

Syntax

```
clearStartingInstance(String "userOrGroupType",
String "userOrGroupName");
```

Parameter

This function has the following parameters.

Table 37. clearStartingInstance Parameters

Parameter	Format	Description
userOrGroupType	String	Specifies whether to clear the starting instance set for a user or a group.
userOrGroupName	String	The name of the user or group.

deleteServiceInstance

Use the deleteServiceInstance function to delete service instances from the IBM Tivoli Business Service Manager database.

Syntax

```
deleteServiceInstance(new String[] {"InstanceName1", "InstanceName2"});
```

Parameter

This function has the following parameters:

Table 38. deleteServiceInstance Parameters

Parameter	Format	Description
InstanceName	String Array	The names of the service instances you want to delete.

Example

The following example deletes the service instance named dbserver1.

```
deleteServiceInstance (new String[] {"dbserver1"});
```

dumpInstanceInfo

Use the dumpInstanceInfo function to display information about a service instance from the IBM Tivoli Business Service Manager database.

Syntax

```
dumpInstanceInfo("InstanceName");
```

Parameter

This function has the following parameters:

Table 39. dumpInstanceInfo Parameters		
Parameter	**Format**	**Description**
InstanceName	String	The name of the service instance whose information you want to display.

Example

The following example dumps the service instance named dbserver1.

```
dumpInstanceInfo ("dbserver1");
```

removeScheduleFromInstance

Use the removeScheduleFromInstance function to remove a maintenance schedule from a service or services.

These are the same parameters you specify in the **Edit Service** panel of the IBM Tivoli Business Service Manager GUI. See the TBSM 4.2.1 *Service Configuration Guide* for more information about maintenance schedules.

Syntax

removeScheduleFromInstance("*instanceName*", scope);

Parameters

This function has the following parameters:

Table 40. removeScheduleFromInstance Parameters		
Parameter	**Format**	**Description**
instanceName	String	The name of the service instance in the database.
scope	Integer	Specifies the scope of schedule assignment. When set to 0 (default), the schedule is removed from the specified service. When set to 1, the schedule is removed from the service and all its child services. When set to 2, the schedule is removed from the child services, but is not removed from the service itself.

Examples

The following example shows how to remove a maintenance schedule from the webserver1 service instance.

```
removeScheduleFromInstance("webserver1", 0);
```

The following example shows how to remove a maintenance schedule from the webserver1 service and all its child service instances.

```
removeScheduleFromInstance("webserver1", 1);
```

The following example shows how to remove a maintenance schedule from only from the child services of the webserver1 service. The schedule is not removed from webserver1 service.

```
removeScheduleFromInstance("webserver1", 2);
```

setGISMapNameForInstance

Use the setGISMapNameForInstance to specify a given GIS map to for a given service instance in the ***Service Viewer***.

Syntax

setGISMapNameForInstance("*InstanceName*","*mapName*");

Parameters

This function has the following parameters:

Table 41. setGISMapNameforInstance Parameters

Parameter	Format	Description
InstanceName	String	The name of the service instance where you want to have a custom GIS map.
mapName	String	The file name for the map you want to use.

Example

The following example sets the GIS map Asiamap.svg for a service instance named ABCBankAsia:

```
setGISMapNameForInstance("ABCBankAsia","Asiamap.svg");
```

printCountOfServiceInstances

Use the printCountOfServiceInstances function to print the total number of service instances.

Syntax

This function uses the following syntax:

```
printCountOfServiceInstances();
```

Parameters

This function has no parameters.

Service template commands

Use these commands to configure service templates. These templates define the common behavior for a given service type. You must create the template before you can use the other RAD Shell functions.

addAverageDependencyAttributeToTemplate

Use the addAverageDependencyAttributeToTemplate function to add a numerical-aggregation rule to a service template, using the Average aggregation function. For more information about numerical-aggregation rules, see the TBSM *Service Configuration Guide*.

Syntax

```
addAverageDependencyAttributeToTemplate
("parentTemplateName",
"childTemplateName",
"childRuleName"
"ruleName"
"multiplierExpression");
```

Parameters

This function has the following parameters.

Table 42. addAverageDependencyAttributeToTemplate parameters

Parameter	Format	Description
parentTemplateName	String	The name of the template for the dependency rule.
childTemplateName	String	The name of the child template that supports the parent template.
chileRuleName	String	The name of the child-service-template rule that provides the output values you want to aggregate with this rule.
ruleName	String	Optional. The name of the dependency rule. If you do not enter anything for this parameter, the system creates a name based on the other settings.
multiplierExpression	String	Optional. The multiplier you want to apply to the average.

Example

The following example shows how to create a dependency rule that calculates the average values for the child services assigned to the OS_WEBFARM service template.

This example is configured as follows:

1. The parent service template is OS_REGION.
2. The child service template is OS_WEBFARM. The child services assigned to the OS_WEBFARM are used in the calculations for this aggregation rule.
3. The child rule name is CriticalWebTickets. This aggregation rule calculates the average output value for the CriticalWebTickets child rule.

4. The rule name is `AvgCriticalWebTickets`.
5. The multiplier expression is blank.

```
[1]   addAverageAttributeToTemplate("OS_REGION",
[2]   "OS_WEBFARM",
[3]   "CriticalWebTickets",
[4]   "AvgCriticalWebTickets"
[5]   "");
```

addBooleanExpressionsAttributeToTemplate

Use the `addBooleanExpressionsAttributeToTemplate` function to create a new secondary output expression when status changes of a service instance to bad or marginal. You can combine the status of multiple rules using this function.

These are the same parameters you specify in the **Output Expressions** tab for the **Edit Template** panel in the IBM Tivoli Business Service Manager GUI.

Syntax

`addBooleanExpressionsAttributeToTemplate("`*parentTemplateName*`",`
`"`*badOutputExpression*`",`
`"`*marginalOutputExpression*`");`

Parameters

This function has the following parameters.

Table 43. addBooleanExpressionsAttributeToTemplate Parameters

Parameter	Format	Description
parentTemplateName	String	The name of the template where you want an output expression.
badOutputExpression	String	The expression that, when evaluated to be `true`, causes the service status to be set to bad.
marginalOutputExpression	String	The expression that, when evaluated to be `true`, causes the service status to be set to `marginal`.

Example

The following example adds additional conditions to the `WebServerWorst` event rule for the `WebFarm` template.

1. The parent template is `WebFarm`.
2. The bad output expression sets the `WebFarm` status to bad when the `WebserverWorst` status is Bad and the `RawAttribute` status is `Marginal`.
3. The marginal output expression sets the `WebFarm` status to marginal when the `WebserverWorst` status is `Marginal` and the `RawAttribute` status is Bad.

```
[1]   addBooleanExpressionsAttributeToTemplate "WebFarm",
[2]   "(WebServerWorst.Value = Bad" AND "RawAttribute.Value <= Marginal)",
[3]   "(WebServerWorst.Value = Marginal" AND "RawAttribute.Value = Bad)";
```

addCummulativeDurationSlaAttributeToTemplate

Use the addCummulativeDurationSlaAttributeToTemplate function to add cumulative-outage time SLA settings to a service template.

These are the same parameters you specify when you check the **Calculate cumulative duration SLA violations** settings in the **SLA** tab for the **Edit Template** panel in the IBM Tivoli Business Service Manager GUI. With this function, you set the amount of time a service can be unavailable within a given calendar time period before TBSM sends a violation or warning event. The cumulative time violation calculations track the cumulative time for all outages within the specified time period. See the TBSM *Service Configuration Guide* for more information about configuring SLAs.

Syntax

```
addCumulativeDurationSlaAttributeToTemplate("TemplateName,
"timePeriod",
"slaName",
String[] {"violationThresholdValues"},
String[] {"warningThresholdValues"},
"timeBoundaryString");
```

Parameter

This function has the following parameters.

Table 44. addCummulativeDurationSlaAttributeToTemplate function parameters

Parameter	Format	Description
TemplateName	String	The name of the template where you want to add an SLA attribute.
timePeriod	String	Sets the time-period selection to month, day, hour, and minute.
slaName	String	Name of the SLA (Standard, Gold, and so on) associated with the attribute.
violationThresholdValues	String Array	The time period that triggers an SLA violation in hours, minutes, and seconds. Enter the values in the format *HH,MM,SS*.
warningThresholdValues	String Array	The time period that triggers an SLA warning in hours, minutes, and seconds. Enter the values in the format *HH,MM,SS*.
timeBoundaryString	String	If you set a monthly SLA tracking period, set the day of the month when the tracking starts with this parameter. Specify the start and end days in the following format: "1st to 1st", "2nd to 2nd", "15th to 15th", and so on.

Example

The following example creates an SLA attribute with the following settings:

1. The service template where the attribute is added in WebFarm.
2. The time period for the SLA tracking is a month.
3. The SLA where the attribute is added is WebFarmSlaGold.
4. The violation threshold value is set to 15 minutes.
5. The warning threshold value is set to 10 minutes.

6. The day of the month when the SLA tracking starts is on the third day of the month (`3rd to 3rd`).

```
[1]   addCummulativeDurationSlaAttributeToTemplate("WebFarm",
[2]                                                "month",
[3]                                                "WebFarmSlaGold",
[4]                                                 new String[] {"0","15","0"},
[5]                                                 new String[] {"0","10","0"},
[6]                                                "3rd to 3rd");
```

addDurationCountSlaAttributeToTemplate

Use the `addDurationCountSlaAttributeToTemplate` function to configure duration-based SLA settings for a service template.

These are the same parameters you specify when you check **Calculate duration based SLA violations** in the **SLA** tab for the **Edit Template** panel in the IBM Tivoli Business Service Manager GUI. See the TBSM *Service Configuration Guide* for more information about configuring SLAs.

Syntax

```
addDurationCountSlaAttributeToTemplate("TemplateName",
"slaName",
String[] "violationThresholdValues",
String[] "warningThresholdValues");
```

Parameters

This function has the following parameters.

Table 45. `addDurationCountSlaAttributeToTemplate` function parameters

Parameter	Format	Description
TemplateName	String	The name of the template where you want to add a violation count SLA rule.
slaName	String	Name of the SLA (Standard, Gold, and so on) associated with the attribute.
violationThresholdValues	String Array	The time period that triggers an SLA violation in hours, minutes, and seconds. Enter the values in the format *HH,MM,SS*.
warningThresholdValues	String Array	The times period that triggers an SLA warning in hours, minutes, and seconds. Enter the values in the format *HH,MM,SS*.

Example

The following example creates a duration-based SLA attribute with the following settings:

1. The service template where the new SLA settings are added is `WebFarm`.
2. The SLA name is `WebFarmSLAGold`.
3. The violation threshold value is set to 5 minutes.
4. The warning threshold value is set to 2 minutes.

```
[1]   addDurationCountSlaAttributeToTemplate("WebFarm",
[2]   "WebFarmSlaGold",
[3]   new String[] {"0","5","0"},
[4]   new String[] {"0","2","0"});
```

addIncidentCountSlaAttributeToTemplate

Use the addIncidentCountSlaAttributeToTemplate function to add violation-count SLA settings to a service template.

These are the same parameters you specify when you check **Calculate number of violations in given time period** settings in the **SLA** tab for the **Edit Template** panel in the IBM Tivoli Business Service Manager GUI. See the TBSM *Service Configuration Guide* for more information about configuring SLAs.

Syntax

```
addIncidentCountSlaAttributeToTemplate("TemplateName",
"slaName",
String[]violationThresholdValues,
String[] warningThresholdValues,
String[] rollingTimeWindows);
```

Parameters

This function has the following parameters.

Table 46. addIncidentCountSlaAttributeToTemplate function parameters

Parameter	Format	Description
TemplateName	String	The name of the service template where you want to add an SLA attribute.
slaName	String	Name of the SLA (Standard, Gold, and so on) associated with the attribute.
violationThresholdValues	String Array	The number of outage incidents that trigger an SLA violation.
warningThresholdValues	String Array	The number of outage incidents that trigger an SLA violation warning.
rollingTimeWindows	String	Enter the time period for the cumulative violation calculation. Enter the values in the format *HH,MM,SS*.

Example

The following example creates an incident-count SLA attribute with the following settings:

1. The service template where the attribute is added is WebFarm.
2. The SLA name is WebFarmSLAGold.
3. The number of outages that trigger an SLA violation is 3.
4. The number of outages that trigger an SLA violation warning is 2.
5. The rolling time window is set to 5 minutes and 10 seconds.

```
[1]   addIncidentCountSlaAttributeToTemplate("WebFarm",
[2]                                          "WebFarmSlaGold",
[3]                                           new String[] {"3"},
[4]                                           new String[] {"2"},
[5]                                           new String[] {"0","5","10"});
```

addInternalAttributeToTemplate

Use the addInternalAttributeToTemplate function to add a formula rule that combines the output of multiple rules for a given service template. For more information about formula rules, see the TBSM *Service Configuration Guide*.

Syntax

```
addInternalAttributeToTemplate
("templateName"
"ruleName",
"formula",
"policyName",
"policyScript",
"policyType",
"badThreshold",
"marginalThreshold",
textRule);
```

Parameters

This function has the following parameters.

Table 47. addInternalAttributeToTemplate function parameters

Parameter	Format	Description
templateName	String	The name of the template for the formula rule.
ruleName	String	Optional. The name of the dependency rule. If you do not enter anything for this parameter, the system creates a name based on the other settings.
formula	String	The formula that combines the output values from multiple rules.
policyName	String	Optional. The name of the policy used to calculate the rule-output value you want.
policyScript	String	Optional. The full text of the policy you created for this rule.
policyType	String	Optional. The class name for the policy. For IPL policies this is com.micromuse.response.common.parser.IPLPolicy. For javascript policies this is com.micromuse.response.common.parser.JavaScriptPolicy. The default is com.micromuse.response.common.parser.IPLPolicy"
badThreshold	String	Optional. The output value that changes the service status to bad.
marginalThreshold	String	Optional. The output value that changes the service status to marginal.
textRule	Boolean	Optional. Set this parameter to true to indicate that this is a text rule. The default is false which indicates that it is a numerical rule.

Example 1

The following example creates an internal-formula rule that adds the output values from three other rules.

This example is configured as follows:

1. The parent service template is `OS_REGION`.
2. The rule name is `RegSumCrit`.
3. The rule values for the `NetSumCritical`, `WebSumCritical`, and `DBSumCrit` rules are added to calculate the rule-output value for this rule.
4. No custom policy is used for this rule.
5. No custom policy is used for this rule.
6. No custom policy is used for this rule.
7. The bad-status threshold is `50`.
8. The marginal-status threshold is 25.
9. This is a numerical rule.

```
[1]   addInternalAttributeToTemplate("OS_REGION",
[2]    "RegSumCrit",
[3]    "NetSumCritical.Value +  WebSumCritical.Value + DBSumCrit.Value",
[4]    null,
[5]    null,
[6]    null,
[7]    null,
[8]    null,
[9]    false );
```

Example 2

The following example creates an internal-formula rule that uses a formula.

This example is configured as follows:

1. The parent service template is `myTemplate`.
2. The rule name is `newPolicyRule`.
3. No rule values are used directly.
4. The name of the custom policy used for this rule.
5. The policy statements for policy is used for this rule.
6. The policy type is ipl
7. No status is set for this rule, so the bad threshold is not required.
8. No status is set for this rule, so the marginal threshold is not required.
9. This is not a text rule.

```
[1]   addInternalAttributeToTemplate("myTemplate",
[2]    "newPolicyRule",
[3]    null,
[4]    "myPolicyName",
[5]    " ENDLINE Log(\"something\"); ENDLINE Status = 3; ENDLINE Log(\"something else\");
ENDLINE ",
[6]    "com.micromuse.response.common.parser.IPLPolicy",
[7]   null,
[8]   null,
[9]   false );
```

For more information about creating policies for formula rules, see https://www.ibm.com/support/knowledgecenter/SSSPFK_6.2.0/com.ibm.tivoli.itbsm.doc/customization/bsmc_polc_num_formrule_createcustom.html

Additional call to addInternalAttributeToTemplate to set optional Status

When creating a formula, there is an option to create a `Status` based on `Bad` and `Marginal` thresholds. If this is required, then the rule should be created, as above, and an additional `radshell` call to `addInternalAttributeToTemplate` is required for the status/thresholds.

For the additional call to `addInternalAttributeToTemplate`, the template name is the same as before but the rule name has `_Status` appended. The third parameter should be set to the value attribute of the original rule name, namely: `rulename.value`

The parameters are:

1. The service template name.
2. The rule name as below, with `_Status` appended, namely: `RuleName_Status`
3. The value attribute of the rule name, namely: `RuleName.value`
4. Set to `DontTestFormula`
5. No custom policy is used for this rule. The default `com.micromuse.response.common.parser.IPLPolicy` can be used.
6. The bad-status threshold is `50`.
7. The marginal-status threshold is `25`.
8. This is a numerical rule.

Example 3

The following example creates an internal-formula rule that uses a formula.

After adding the policy attribute, an example of the second call to set the status thresholds is below:

```
addInternalAttributeToTemplate(
"myTemplate",
"newPolicyRule_Status",
"newPolicyRule.Value",
"DontTestFormula",
null,
"com.micromuse.response.common.parser.IPLPolicy",
"50",
"25",
false
);
```

addMaxDependencyAttributeToTemplate

Use the `addMaxDependencyAttributeToTemplate` function to add a numerical-aggregation rule to a service template, using the `Max` aggregation function. For more information about numerical-aggregation rules, see the TBSM *Service Configuration Guide*.

Syntax

```
addMaxDependencyAttributeToTemplate
("parentTemplateName",
"childTemplateName",
"childRuleName"
"ruleName");
```

Parameters

This function has the following parameters.

Table 48. addMaxDependecyAttributeToTemplate Parameters		
Parameter	**Format**	**Description**
parentTemplateName	String	The name of the template for the dependency rule.
childTemplateName	String	The name of the child template that supports the parent template.
chileRuleName	String	The name of the child-service-template rule that provides the output values you want to aggregate with this rule.
ruleName	String	Optional. The name of the dependency rule. If you do not enter anything for this parameter, the system creates a name based on the other settings.

Example

The following example shows how to create a dependency rule that calculates the maximum output values for the child services assigned to the OS_WEBFARM service template.

This example is configured as follows:

1. The parent service template is OS_REGION.
2. The child service template is OS_WEBFARM. The child services assigned to the OS_WEBFARM are used in the calculations for this aggregation rule.
3. The child rule name is CriticalWebTickets. This aggregation rule calculates the maximum output value from the CriticalWebTickets child rule.
4. The rule name is MaxCriticalWebTickets.

```
[1]   addMaxAttributeToTemplate("OS_REGION",
[2]   "OS_WEBFARM",
[3]   "CriticalWebTickets",
[4]   "MaxCriticalWebTickets");
```

addMinDependencyAttributeToTemplate

Use the addMinDependencyAttributeToTemplate function to add a numerical-aggregation rule to a service template, using the Min aggregation function. For more information about numerical-aggregation rules, see the TBSM *Service Configuration Guide*.

Syntax

```
addMinDependencyAttributeToTemplate
("parentTemplateName",
"childTemplateName",
"childRuleName"
"ruleName");
```

Parameters

This function has the following parameters.

Table 49. addMinDependencyAttributeToTemplate function parameters		
Parameter	**Format**	**Description**
parentTemplateName	String	The name of the template for the dependency rule.
childTemplateName	String	The name of the child template that supports the parent template.
chileRuleName	String	The name of the child-service-template rule that provides the output values you want to aggregate with this rule.
ruleName	String	Optional. The name of the aggregation rule. If you do not enter anything for this parameter, the system creates a name based on the other settings.

Example

The following example shows how to create a dependency rule that calculates the minimum values for the child services assigned to the OS_WEBFARM service template.

This example is configured as follows:

1. The parent service template is OS_REGION.
2. The child service template is OS_WEBFARM. The child services assigned to the OS_WEBFARM are used in the calculations for this aggregation rule.
3. The child rule name is CriticalWebTickets. This aggregation rule calculates the minimum output value from the CriticalWebTickets child rule.
4. The rule name is MinCriticalWebTickets.

```
[1]   addMinAttributeToTemplate("OS_REGION",
[2]   "OS_WEBFARM",
[3]   "CriticalWebTickets",
[4]   "MinCriticalWebTickets");
```

addNewRawAttribute

Use the addNewRawAttribute function to add new incoming-status rules to a service template.

You configure the following settings for each template event attribute (rule) with this function:

- Service template
- Rule name
- Event discriminator field values used to identify events
- Event fields used to name a service instance tagged with the template
- Rule type (numeric or good, marginal, or bad)
- Data feed

These parameters match the fields on the top of the **Status Rule Configuration** window. After you add the rule with this function, you must specify the threshold or numeric filters for the rule with the addRawAttributeThresholdSet function as described in “addRawAttributeThresholdSet” on page 121.

Syntax

This function has the following syntax

```
addNewRawAttribute("TemplateName","RuleName",
new String[]{"eventDiscriminatorNames"},
new String[]{"instanceExpressionFields"},
rule type,
"data feed"
);
```

Parameters

This function has the following parameters:

Table 50. addNewRawAttribute function parameters

Parameter	Format	Description
TemplateName	String	The name of the template you want to create. In the GUI, these values appear in the **Template Name** field in the **Edit Rule Set** panel.
RuleName	String	The name of the event rule you want to create for the service template in the GUI.
eventDiscriminatorNames	String Array	The value of the event **Class** field used to identify an event for the event rule. If you are defining a rule that uses a data fetcher data feed, enter: `new String[] { "Default Class(0)" }` for this parameter.
instanceExpressionFields	String Array	An array of event fields used to create the name for service instances tagged with this template. You can also use Netcool/Impact version 3.1 expressions to extract specific values from these fields.
rule type	Integer	Specifies whether the rule is a numeric or a good, marginal, bad incoming-status rule. 0 = good, marginal, bad 1 = numeric
data feed	String	Specifies the data feed for the rule. The values can be `ObjectServer`, or the name of a data fetcher.

Examples

The examples in this section show how to configure both incoming-status rule types using the `addNewRawAttribute` function.

Numeric Rule Example

This example adds a new numeric incoming-status rule named `HighNetworkTickets`.

1. Calls the addNewRawAttribute function.
2. The new rule is added to the OS_NETWORK service template.
3. The rule name is HighNetworkTickets.
4. This rule uses data from a data fetcher. As a result, the default event discriminator { "Default Class(0)" } must be specified.
5. This rule uses the following fields to identify the service instance names: "highlevelserviceid","lowlevelserviceid","region","customer".
6. The rule type of 1 is sets this as a numeric rule.
7. The data feed is a data fetcher named RegionalTickets.

Lines 10-18 show the addRawAttributeThresholdsSet function you need to create to finish the rule configuration. For more information about the addRawAttributeThresholdsSet function, see "addRawAttributeThresholdSet" on page 121.

```
[1]   addNewRawAttribute(
[2]   "OS_NETWORK",
[3]   "HighNetworkTickets",
[4]   new String[] { "Default Class(0)" },
[5]   new String[] { "highlevelserviceid","lowlevelserviceid","region","customer" },
[6]   1,
[7]   "RegionalTickets"
[8]   );
[9]
[10]   addRawAttributeThresholdSet(
[11]   "OS_NETWORK",
[12]   "HighNetworkTickets",
[13]   "opentickets",
[14]   null,
[15]   new String[] { "lowlevelserviceid",:severity" },
[16]   new String[] { "=","=" },
[17]   new String[] { "Network","'4'" },
[18]   0
[19]   );
```

Good, Marginal, Bad Rule Example

This example adds a new good, marginal, bad incoming-status rule named DBEvent_status.

1. Calls the addNewRawAttribute function.
2. The new rule is added to the OS_DBFARM service template.
3. The rule name is DBEvent_status.
4. The event discriminator is { "Default Class(0)" }.
5. This rule uses the Node fields to identify the service instance names.
6. The rule type of 0 is sets this as a good, marginal, bad rule.
7. The data feed is the default IBM Tivoli Business Service Manager ObjectServer.

Lines 10-30 show the addRawAttributeThresholdsSet function you need to finish the rule configuration, as described in "addRawAttributeThresholdSet" on page 121.

```
[1]   addNewRawAttribute(
[2]   "OS_DBFARM",
[3]   "DBEvent_status"
[4]   new String[] { "Default Class(0)" },
[5]   new String[] { "Node" },
[6]   0,
[7]   "ObjectServer"
[8]   );
[9]
[10]   addRawAttributeThresholdSet(
[11]   "OS_DBFARM",
[12]   "DBEvent_status",
[13]   "Bad",
[14]   null,
[15]   new String[] { "Severity","AlertKey" },
```

```
[16]   new String[] { ">=","=" },
[17]   new String[] { "5","DBFarm" },
[18]   0
[19]   );
[20]
[21]   addRawAttributeThresholdSet(
[22]   "OS_DBFARM",
[23]   "DBEvent_status",
[24]   "Marginal",
[25]   null,
[26]   new String[] { "Severity","AlertKey" },
[27]   new String[] { ">=","=" },
[28]   new String[] { "3","DBFarm" },
[29]   0
[30]   );
```

addPercentageOfChildrenDependencyAttributeToTemplate

Use the addPercentageOfChildrenDependencyAttributeToTemplate to configure the percentage of child service dependency rule for the service template.

With this type of rule, the status of the parent service instance changes when the specified percentage of child service instances tagged with the child template report a bad or marginal state. These are the same parameters you specify when you click the **% of Children** button in the **Good, Marginal, Bad Aggregation** Rule window of the IBM Tivoli Business Service Manager GUI.

Syntax

addPercentageOfChildrenDependencyAttributeToTemplate
("*parentTemplateName*",
"*childTemplateName*",
"*ruleName*"
"*childThresholdState*",
badPercentageThreshold,
marginalPercentageThreshold, ,
isChildInstancePropagation,
"*weightPropertyName*" ,
weightDefaultValue);

Parameter

This function has the following parameters.

Table 51. addPercentageOfChildrenDependencyAttributeToTemplate function parameters

Parameter	Format	Description
parentTemplateName	String	The name of the template where you want to add a dependency.
childTemplateName	String	The name of the child template that supports the parent template.
ruleName	String	Optional. The name of the dependency rule. If you do not enter anything for this parameter, the system creates a name based on the other settings.

Table 51. addPercentageOfChildrenDependencyAttributeToTemplate function parameters (continued)

Parameter	Format	Description
childThresholdState	String	The valid state values are Bad or `Marginal`. The rule monitors the dependent child for this state. If a specified percentage of the children are in this threshold state or worse, the parent service status changes. The *isChildInstancePropagation* parameter determines whether the child template state or the overall child instance status is monitored.
badPercentageThreshold	Integer	A percentage threshold for child service instances reporting the status specified in the *childThresholdState* parameter. When this threshold is crossed, the parent instance status changes to bad.
marginalPercentageThreshold	Integer	A percentage threshold for child service instances reporting the status specified in the *childThresholdState* parameter. When this threshold is crossed, the parent instance status changes to marginal.
isChildInstancePropagation	Boolean	Specifies whether the overall status of the parent service is determined by either the status of *childTemplateName*, or the overall status of the child services tagged with *childTemplateName*. If the value of this attribute is set to `false` the status is dependent on the status of *childTemplateName*. If the value is set to `true`, the status is dependent on the net overall status of the child services tagged with *childTemplateName*. The true option is useful for children with multiple templates. The child instance status propagates regardless of which child template changed the child instance state. **Note:** This option only applies to the Good, Marginal, Bad Aggregation rule.

Table 51. addPercentageOfChildrenDependencyAttributeToTemplate function parameters (continued)		
Parameter	**Format**	**Description**
weightPropertyName	String	(Optional) This is a service instance property name. The value is used to calculate the weighted effect of this instance.
weightDefaultValue	Boolean	(Optional) Default value is -1. This is the default weighted value applied to the service instance if no property value exists or if it is not numeric.

Example

The following example shows how to create a dependency rule where the parent `WebFarm` service instance's status changes as follows:

- Status is bad when 70% of its child Webservers report a marginal state
- Status is marginal when 30% of its child Webservers report a marginal state.

This example is configured as follows:

1. The parent template is `WebFarm`.
2. The child template is `WebServer`.
3. The child threshold state is `Marginal`. That is, this rule tracks the percentage of child Webservers with a state of marginal or worse.
4. The bad threshold percentage for child Webserver instances that report a marginal status is `70`. That is, when the 70% of the Webserver instances report a marginal state or worse, the status of the parent WebFarm instance changes to bad.
5. The marginal threshold percentage for child Webserver instances that report a marginal status is `30`. That is, when the 30% of the Webserver instances report a marginal state or worse, the status of the parent WebFarm instance changes to marginal.

```
[1]   addPercentageOfChildrenDependencyAttributeToTemplate ("WebFarm",
[2]   "WebServer",
[3]   "Marginal",
[4]   70,
[5]   30,
[6]          false,
[7]      "",
[8]   -1);
```

Example

The following example shows how to create a weighted dependency rule where the status of the parent `Hub_Manager` service changes as follows:

- Status is bad when 60% of its child `Hub_Services` report a marginal state.
- Status is marginal when 30% of its child `Hub_Services` report a marginal state.

This example is configured as follows:

1. The parent template is `Hub_Manager`.
2. The child template is `Hub_Service`.
3. The child threshold state is `Marginal`. That is, this rule tracks the percentage of child `Hub_Services` with a state of marginal or worse.

4. When 30% of the weighted Hub_Service instances are marginal state or worse, then the parent Hub_Manager instance status is marginal, and critical when 60% of the weighted Hub_Service instances are marginal.
5. With an unweighted rule, if we have five Hub_Service instances, and Hub_Service instance 2 and 3 are marginal, the percentage calculation would be 40%. Therefore, Hub_Manager instance would be marginal.
6. The weight property name is Hub_weight.
7. If we assign values to the Hub_weight property for each instance as follows:
   - Hub_Service 1 - 10
   - Hub_Service 2 - 70
   - Hub_Service 3 - 20
   - Hub_Service 4 - 20
   - Hub_Service 5 - 30
8. The percentage calculation would be 60% ((70 + 20) /150), and the Hub_Manager instance would be critical.

```
[1]   addPercentageOfChildrenDependencyAttributeToTemplate ("Hub_Manager",
[2]   "Hub_Service",
[3]   "Marginal",
[4]   60,
[5]   30,
[6]       false,
[7]      "Hub_weight",
[8]   -1);
```

addPercentileDependencyAttributeToTemplate

Use the addPercentileDependencyAttributeToTemplate function to returns the rule output value associated with the child service closest to the specified percentile. For more information about numerical-aggregation rules, see the TBSM *Service Configuration Guide*.

Syntax

addPercentileDependencyAttributeToTemplate
("*parentTemplateName*",
"*childTemplateName*",
"*childRuleName*",
"*ruleName*",
"*Percentile*");

Parameters

This function has the following parameters.

Table 52. addPercentileDependecyAttributeToTemplate function parameters

Parameter	Format	Description
parentTemplateName	String	The name of the template for the dependency rule.
childTemplateName	String	The name of the child template that supports the parent template.

Table 52. addPercentileDependecyAttributeToTemplate function parameters (continued)		
Parameter	**Format**	**Description**
chileRuleName	String	The name of the child-service-template rule that provides the output values you want to aggregate with this rule.
ruleName	String	Optional. The name of the dependency rule. If you do not enter anything for this parameter, the system creates a name based on the other settings.
Percentile	String	The percentile you want to calculate for the group of rule-output values.

Example

The following example shows how to create a dependency rule that calculates the 50th percentile of the output values for the child services assigned to the OS_WEBFARM service template.

This example is configured as follows:

1. The parent service template is OS_REGION.
2. The child service template is OS_WEBFARM. The child services assigned to the OS_WEBFARM are used in the calculations for this aggregation rule.
3. The child rule name is CriticalWebTickets. This aggregation rule calculates the 50th percentile of the output value from the CriticalWebTickets child rule.
4. The rule name is 50PercentileCriticalWebTickets.
5. The percentile is 50.

```
[1]   addPercentileAttributeToTemplate("OS_REGION",
[2]   "OS_WEBFARM",
[3]   "CriticalWebTickets",
[4]   "50percentileCriticalWebTickets"
[5]   "50");
```

addSlaAttributeToTemplate

Use the addSlaAttributeToTemplate function to add SLA settings to a service template. You can add duration-based, incident count, or cumulative SLA settings with this function.

These are the same parameters you specify in the **SLA** tab for the **Edit Template** panel in the IBM Tivoli Business Service Manager GUI. See the TBSM *Service Configuration Guide* for more information about configuring SLAs.

Syntax

This function has the following syntax:

```
addSlaAttributeToTemplate("TemplateName",
"slaType",
"slaName",
String[] "violationThresholdValues",
String[] "warningThresholdValues",
"timeBoundaryString,"
String[] "rollingTimeWindows",
makePersistent);
```

Parameters

This function has the following parameters.

Table 53. addSlaAttributeToTemplate function parameters

Parameter	Format	Description
TemplateName	String	The name of the template where you want to add an SLA attribute.
slaType	String	The SLA type, such as `DurationCount`.
slaName	String	Name of the SLA (Standard, Gold, and so on) associated with the attribute.
violationThresholdValues	String Array	The times that trigger an SLA violation or warning in hours, minutes, and seconds. Enter the values with the syntax *HH: MM: SS*
warningThresholdValues	String Array	The times that trigger an SLA violation warning in hours, minutes, and seconds. Enter the values with the syntax *HH: MM: SS*.
timeBoundaryString	String	The time period or periods where you track the cumulative duration SLA. The following values are valid: `Month: dates, Day, Hour, Minute`.
rollingTimeWindow	String	Rolling time window for tracking the number of violations over a given time period.
makePersistent	Boolean	Determines whether the changes are persisted to the database.

Example

The following example creates an SLA attribute with the following settings and persists the changes to the database:

1. The service template for the SLA attribute is `WebFarm`.
2. The SLA type is `DurationCount`.
3. The SLA name is `WebFarmESlaGold`.
4. The violation threshold value is 33 minutes.
5. The warning threshold value is 22 minutes.
6. The time boundary does not apply to duration count SLA settings and is set to null.
7. The rolling time window does not apply to duration count SLA settings and is set to null.

```
[1]   addSlaAttributeToTemplate("WebFarm",
[2]                           "DurationCount",
[3]                           "WebFarmESlaGold",
[4]                           new String[] {"00","33","00"},
[5]                           new String[] {"00","22","00"},
[6]                           null,
[7]                           null,
                              true);
```

addSumDependencyAttributeToTemplate

Use the addSumDependencyAttributeToTemplate function to add a numerical-aggregation rule to a service template, using the Sum aggregation function. For more information about numerical-aggregation rules, see the *Service Configuration Guide*.

Syntax

addSumDependencyAttributeToTemplate
("*parentTemplateName*",
"*childTemplateName*",
"*childRuleName*"
"*ruleName*");

Parameters

This function has the following parameters.

Table 54. addDependecyAttributeToTemplate function parameters

Parameter	Format	Description
parentTemplateName	String	The name of the template for the dependency rule.
childTemplateName	String	The name of the child template that supports the parent template.
chileRuleName	String	The name of the child-service-template rule that provides the output values you want to aggregate with this rule.
ruleName	String	Optional. The name of the dependency rule. If you do not enter anything for this parameter, the system creates a name based on the other settings.

Example

The following example shows how to create a dependency rule that adds output values for the child services assigned to the OS_WEBFARM service template.

This example is configured as follows:

1. The parent service template is OS_REGION.
2. The child service template is OS_WEBFARM. The child services assigned to the OS_WEBFARM are used in the calculations for this aggregation rule.
3. The child rule name is CriticalWebTickets. This aggregation rule sums the output values from the CriticalWebTickets child rule.
4. The rule name is SumCriticalWebTickets.

```
[1]   addMinAttributeToTemplate("OS_REGION",
```

```
[2]   "OS_WEBFARM",
[3]   "CriticalWebTickets",
[4]   "SumCriticalWebTickets");
```

addWorstChildDependencyAttributeToTemplate

Use the addWorstChildDependencyAttributeToTemplate function to create the worst child dependency rule for a service template.

This type of rule changes the status of the parent service template when any child service instance tagged with the child template reports a bad or marginal state. These are the same parameters you specify when you click the **Any Child** button in the **Good, Marginal, Bad Aggregation Rule** of the TBSM GUI.

Syntax

addWorstChildDependencyAttributeToTemplate ("*parentTemplateName*",
"*childTemplateName*",
"*ruleName*",
" *badThreshold*",
"*marginalThreshold*",
isChildInstancePropagation);

Parameters

This function has the following parameters.

Table 55. addWorstChildDependencyAttributeToTemplate function parameters

Parameter	Format	Description
parentTemplateName	String	The name of the template where you want to add a dependency.
childTemplateName	String	The name of the child template that supports the parent template. If any child service instance assigned to the parent service reports a bad or marginal state, the status of the parent service changes based on the *isChildInstancePropagation* parameter.
ruleName	String	(Optional) Name of the dependency rule you want to add. If you do not specify this parameter, the system creates a rule name based on the parent template name.
badThreshold	String	Specifies the state (Bad, Marginal, None) for the parent service when the worst state of a child service instance is Bad.
marginalThreshold	String	Specifies the state (Bad, Marginal) for the parent service when the worst state of a child service instance is Marginal.

Table 55. addWorstChildDependencyAttributeToTemplate function parameters (continued)		
Parameter	**Format**	**Description**
isChildInstancePropagation	Boolean	Specifies whether the overall status of the parent service is determined by either the status of a child template specified in the dependency rule in the parent template, or the overall status of the child services tagged with the child template. If the value of this attribute is set to `false` the status is dependent on the status of the specified child template. If the value is set to `true`, the status is dependent on the net overall status of the child services tagged with the specified child template. The true option is useful for children with multiple templates. The child instance status propagates regardless of which child template changed the child instance state. **Note:** This option only applies to the Good, Marginal, Bad Aggregation rule.

Example

The following example adds the service dependency rule that has a dependency on the DBServer service template called DBServerWorst to the DBFarm service template. The state of the DBFarm service changes to bad or marginal whenever the state of any child DBService changes to bad or marginal.

```
addWorstChildDependencyAttributeToTemplate ("DBFarm","DBServer",
"DBServerWorst" "Bad", "Marginal", false);
```

Example

In the following example, the DBFarm service changes whenever the state of any child tagged with Key_Farm_Element changes regardless of what child template changed the state of the child status.

```
addWorstChildDependencyAttributeToTemplate ("DBFarm","Key_Farm_Element",
"DBServerWorst" "Bad", "Marginal", true);
```

addRawAttributeThresholdSet

Use the addRawAttributeThresholdSet function to add the field-value pairs for the threshold-filters in an incoming status rule.

Syntax

This function has the following syntax:

```
addRawAttributeThresholdSet("TemplateName","RuleName",
"ThresholdName",
"ThresholdCustomFilterExpression",
new String[]{"ThresholdFieldNames"},
new String[]{""ThresholdOperators""},
new String[]{"ThresholdFieldValues"},
countvalue
);
```

Parameters

This function has the following parameters:

Table 56. addRawAttributeThresholdSet function parameters

Parameter	Format	Description
TemplateName	String	Name of the template containing the rule being defined. The template must already exist.
RuleName	String	The name of the event rule you want to create for the service template in the GUI. In the GUI, these values appear in the **Rule Name** field in the **Status Rule Configuration** window.
ThresholdName	String	The threshold output name. For good, marginal, bad rules, this value is Bad or Marginal. For numeric rules, this value is the name of the data field that contains the output value for the rule. In the GUI, these values appear in the Filter section of the **Status Rule Configuration** window.
ThresholdCustomFilter Expression	String Array	Optional. Custom filter expression to set the threshold field and value expressions in a single line. If you use a custom filter, you must set the arrays for the *ThresholdName* fields, values, and comparators to null. Conversely, if you use the regular field and value and comparator arrays, set this array to null. In the GUI, these values appear in the **Advanced** window.

Table 56. addRawAttributeThresholdSet function parameters (continued)		
Parameter	**Format**	**Description**
ThresholdFieldNames	String Array	An array of fields used to determine when an event or other data matches the rule. If you use the custom filter expression, set this field array to `null`. In the GUI, these values appear in **Selected Filter Fields** lists in the **Status Rule Configuration** window.
ThresholdOperators	String Array	An array of Boolean comparators used to determine the threshold for a service. The valid values for alphabetic fields are: `=`, `IN`, and `LIKE`. The valid values for numeric fields such as **Severity** are: • `>=` • `=` • `IN` • `>` • `<` • `<=` In the GUI, these values appear in the **Threshold Filter(s)** table in the **Status Rule Configuration** window.
ThresholdFieldValues	String Array	An array of event field values used to determine when a service instance's status changes to bad. This array must be set to `null` if you use custom threshold filter expressions. In the GUI these values appear at the bottom of the Filter section of the **Status Rule Configuration** window.
countvalue	Integer	Value that returns an exception.

Examples

The examples in this section show how to configure both incoming-status rule types using the `addRawAttributeThresholdSet` function.

Numeric rule example

This example adds the threshold set for a numeric incoming-status rule named `HighNetworkTickets`.

You cannot add a threshold set until you have created the rule with the `addNewRawAttribute` function. For more information about the `addNewRawAttribute` function see “addNewRawAttribute” on page 109.

1. Calls the `addRawAttributeThresholdSet` function.
2. The new rule threshold set is added to the `OS_NETWORK` service template.
3. The rule name is `HighNetworkTickets`.

4. This rule uses data from a data fetcher. As a result, the database field that contains the rule-output value is `opentickets`.
5. The custom filter expression is `null`. The threshold set does not use a custom filter expression.
6. The threshold field names are `lowlevelserviceid` and `severity`.
7. The threshold operators are = for the `lowlevelserviceid` field and = for the `severity` field.
8. The threshold field values are `Network` for the `lowlevelserviceid` field and 4 for the `severity` field.
9. The count value is set is `0`.

```
[1]   addRawAttributeThresholdSet(
[2]   "OS_NETWORK",
[3]   "HighNetworkTickets",
[4]   "opentickets",
[5]   null,
[6]   new String[] { "lowlevelserviceid",severity" },
[7]   new String[] { "=","=" },
[8]   new String[] { "Network","'4'" },
[9]   0
[10]   );
```

Good, marginal, bad rule example

This example adds a new good, marginal, or bad threshold set for a rule named `DBEvent_status`.

1. Calls the `addRawAttributeThresholdSet` function.
2. The new rule threshold set is added to the `OS_DBFARM` service template.
3. The rule name is `DBEvent_status`.
4. This the threshold name is `Bad`. These are the settings for the bad rule-output value.
5. The custom filter expression is `null`. The threshold set does not use a custom filter expression.
6. The threshold field names are `severity` and `AlertKey`.
7. The threshold operators are >= for the `severity` field and = for the `AlertKey` field.
8. The threshold field values are 5 for the `severity` field and `DBFarm` for the `AlertKey` field.
9. The count value is set is `0`.

Lines 12-21 configure the threshold set for the `Marginal` threshold in the same rule.

```
[1]   addRawAttributeThresholdSet(
[2]   "OS_DBFARM",
[3]   "DBEvent_status",
[4]   "Bad",
[5]   null,
[6]   new String[] { "Severity","AlertKey" },
[7]   new String[] { ">=","=" },
[8]   new String[] { "5","DBFarm" },
[9]   0
[10]   );
[11]
[12]   addRawAttributeThresholdSet(
[13]   "OS_DBFARM",
[14]   "DBEvent_status",
[15]   "Marginal",
[16]   null,
[17]   new String[] { "Severity","AlertKey" },
[18]   new String[] { ">=","=" },
[19]   new String[] { "3","DBFarm" },
[20]   0
[21]   );
```

Custom filter expression example

The following example sets the thresholds for an incoming-status rule using custom-filter expressions as follows:

1. The `addRawAttributeThresholdSet` function is called and the template name is `WebServer`.
2. The rule name is `WebServerIsOk`.
3. The instance name expression fields are `Node` and `Location`. The & character is stripped from the value of the **Node** field.
4. The threshold name is `Bad`. These are the settings for the bad rule-output value.
5. The custom filter expression changes the service status to bad. The value of the **AlertKey** field is `PingFailure` and the value of the **Severity** field >=5.
6. The parameters for fields, values, and comparators are set to null.
7. The parameters for fields, values, and comparators are set to null.
8. The parameters for fields, values, and comparators are set to null.
9. The count value is set is `0`.

Lines 12-21 configure the threshold set for the `Marginal` threshold in the same rule.

```
[1]   addRawAttributeThresholdSet("WebServer",
[2]   "WebServerIsOK",
[3]   new String[] {"strip(Node, \"&\")", "Location"},
[4]   "Bad",
[5]   "AlertKey = 'PingFailure' AND Severity >= 5",
[6]   null,
[7]   null,
[8]   null,
[9]   0,
[10]   );
[11]   addRawAttributeThresholdSet("WebServer",
[12]   "WebServerIsOK",
[13]   new String[] {"strip(Node, \"&\")", "Location"},
[14]   "Marginal",
[15]   AlertKey = 'PingFailure' AND Severity >= 3",
[16]   null,
[17]   null,
[18]   null,
[19]   0,
[20]   );
```

copyPropertiesToTemplate

Use the `copyPropertiesToTemplate` function to copy additional properties from one service template to another.

The following custom attributes can be defined on the **Additional** tab in the IBM Tivoli Business Service Manager **Edit Template** tab.

copyPropertiesToTemplate(String fromTemplate, String toTemplate)

Syntax

This function has the following syntax:

`copyPropertiesToTemplate("`*fromTemplateName*`", "`*toTemplateName*`")`

Parameters

This function has the following parameters:

Table 57. `copyPropertiesToTemplate` function parameters

Parameter	Format	Description
fromTemplateName	String	The source template for the additional properties you want to copy.

Table 57. copyPropertiesToTemplate function parameters (continued)		
Parameter	**Format**	**Description**
toTemplateName	String	The destination template for the additional properties you want to copy.

Example

The following example copies the additional properties from a service template called OS_DBFarm to a service template called OS_NETWORK.

```
copyPropertiesToTemplate("OS_DBFARM","OS_NETWORK");
```

createSlaForTemplate

Use the createSLAForTemplate function to create a new SLA for a template.

The default service level is Standard. This function allows you to create other service levels such as Gold and Silver.

Syntax

createSlaForTemplate("*TemplateName*", "*slaName*");

Parameters

This function has the following parameters.

Table 58. createSLAForTemplate function parameters		
Parameter	**Format**	**Description**
TemplateName	String	The name of the template where you want a new service level.
slaName	String	The name of the SLA you want to create.

Example

The following example creates an SLA called Gold for the WebCustomers template.

```
createSlaForTemplate ("WebCustomers", "Gold");
```

createTemplate Function

Use the createTemplate function to create a new service template in the IBM Tivoli Business Service Manager database.

Syntax

This function has the following syntax:

createTemplate("*TemplateName*", "*IconName*")

Parameters

This function has the following parameters:

Table 59. `createTemplate` function parameters		
Parameter	**Format**	**Description**
TemplateName	String	The name of the template you want to create.
IconName	String	The name of the file for the icon you want to use for the service template in the GUI. You can choose from the icons described in Table 60 on page 126.

Example

The following example creates a template called `DBFarm` that uses the icon file called `man_svg.gif`.

```
createTemplate ("DBFarm","man_svg.gif");
```

Template Icons

The template icon graphics are in the `$TBSM_DASHBOARD_SERVER_HOME/icons` directory. If you want to add custom icon graphics, see "Adding custom icons" on page 29 for more information.

Table 60. Service template icon names	
Icon	**Filename**
	`blocks_svg.gif`
	`box_svg.gif`
	`bridge_svg.gif`
	`cellphone_svg.gif`
	`cloud_svg.gif`
	`database_svg.gif`
	`db_farm_svg.gif`
	`email_svg.gif`
	`endnode_svg.gif`
	`ethernet_svg.gif`
	`firewall_svg.gif`
	`folder_svg.gif`

Table 60. Service template icon names (continued)	
Icon	**Filename**
	headset_svg.gif
	hub_svg.gif
	man_svg.gif
	pda_svg.gif
	plug_svg.gif
	rack_server_farm_svg.gif
	rack_server_svg.gif
	report_svg.gif
	router_svg.gif
	satellite_dish_svg.gif
	satellite_svg.gif
	server_farm_svg.gif
	server_svg.gif
	switch_svg.gif
	web_svg.gif
	women_svg.gif

deleteTemplate

Use the deleteTemplate function to delete a service template from the IBM Tivoli Business Service Manager database.

Syntax

```
deleteTemplate(new String[] {"TemplateName1", "TemplateName2")};
```

Parameter

This function has the following parameters.

Table 61. deleteTemplate function parameters		
Parameter	**Format**	**Description**
TemplateName	String Array	The name of the templates you want to delete.

Example

The following example deletes the service template named DBServer.

```
deleteTemplate (new String[] {"DBServer")};
```

dumpTemplateInfo

Use the dumpTemplateInfo function to display information about a service template from the IBM Tivoli Business Service Manager database.

Syntax

dumpTemplateInfo("*TemplateName*");

Parameter

This function has the following parameters.

Table 62. dumpTemplateInfo function parameters		
Parameter	**Format**	**Description**
TemplateName	String	The name of the template for which you want information

Example

The following example dumps the service template named DBServer.

```
dumpTemplateInfo ("DBServer");
```

removeDependentAttributeFromTemplate

Use the removeDependentAttributeFromTemplate function to delete a dependency rule from a service template.

Syntax

removeDependentAttribute FromTemplate("*parentTemplateName*",
"*childTemplateName*");

Parameters

This function has the following parameters:

Table 63. removeDependentAttributeFromTemplate function parameters		
Parameter	**Format**	**Description**
parentTemplateName	String	The name of the template where you want to remove a dependency rule.

Table 63. removeDependentAttributeFromTemplate function parameters (continued)		
Parameter	**Format**	**Description**
childTemplateName	String	The name of the service template that you want to remove as a dependency for the parent template.

Example

The following example removes the DBServer service dependency from the WebFarm service template regardless of the rule type.

```
removeDependentAttributeFromTemplate ("WebFarm","DBServer");
```

setHeartBeatPeriodAttributeToTemplate

Use the setHeartBeatPeriodAttributeToTemplate function to set the time period threshold that changes the service instance status to unknown.

Syntax

setHeartBeatPeriodAttributeToTemplate("*TemplateName*", *timeperiod*);

Parameter

This function has the following parameters.

Table 64. setHeartBeatPeriodAttributeToTemplate function parameters		
Parameter	**Format**	**Description**
TemplateName	String	The name of the template where you want to set a heart beat time period.
timeperiod	Int	The heart beat time period threshold in seconds.

Example

The following example sets the heartbeat period to 300 seconds (5 minutes) for the service template called WebServer.

```
setHeartBeatPeriodAttributeToTemplate("WebServer", "300");
```

setTemplateUsesGIS

Use the setTemplateUsesGIS function to enable the GIS options for a service template.

Syntax

setTemplateUsesGIS("*TemplateName*");

Parameter

This function has the following parameters.

Table 65. setTemplateUsesGIS function parameters		
Parameter	**Format**	**Description**
TemplateName	String	The name of the template you want.

Example

The following example enables the GIS options for the WebServer service template.

```
setTemplateUsesGIS("WebServer");
```

refreshRawAttributesEventDiscriminators

Use the `refreshRawAttributesEventDiscriminators` function to refresh all `IncomingStatusRule` event discriminator display names from ObjectServer. An event class name change in ObjectServer cannot be synchronized with TBSM automatically. This function must be executed when you modify an event class name in ObjectServer.

Syntax

```
refreshRawAttributesEventDiscriminators();
```

Parameters

This function has no parameters.

Example

```
refreshRawAttributesEventDiscriminators();
```

validateAllInternalDependentAttributes

Use the `validateAllInternalDependentAttributes` function to validate data integrity of all service template `internalDependentAttributes` in the TBSM database.

Syntax

```
validateAllInternalDependentAttributes
("clearTag");
```

Parameters

This function has the following parameters.

Table 66. validateAllInternalDependentAttributes parameters		
Parameter	**Format**	**Description**
clearTag	Boolean	Determines whether corrupt attributes are removed from TBSM database. Enter `false` to check and display the corrupt data. Enter `true` to check, display, and remove corrupt data.

Example

The following example validates the integrity of all `internalDependentAttributes` in all service templates contained in the TBSM database. It also displays any corrupt data found during this validation:

```
validateAllInternalDependentAttributes (false);
```

addUserPreferencesForInstance

Use the `addUserPreferencesForInstance` function to add a field-value pair (for user preference) to the instance.

Syntax

This function has the following syntax:

```
addUserPreferencesForInstance("instanceName",
"fieldName",
"fieldValue");
```

Parameters

This function has the following parameters.

Table 67. `addUserPreferencesForInstance` parameters

Parameter	Format	Description
instanceName	String	The name of the service instance in the database.
fieldName	String	The name of the field to add.
fieldValue	String	The value to set the field to.

Example

The following example adds user preferences to the service instance `my_services2`, with the name of the field `my_services2_fieldName` and a value of 2.

```
radshell> addUserPreferencesForInstance("my_services2","my_services2_fieldName","2");
```

addInstanceIDFieldValuePair

Use the `addInstanceIDFieldValuePair` function to create a new field-value pair for raw attribute instance identification for the instance.

Syntax

This function has the following syntax:

```
addInstanceIDFieldValuePair("templateName",
"attributeName",
"fieldExpression",
"expressionValue",
comboID,
"instanceName");
```

Parameters

This function has the following parameters.

Table 68. addInstanceIDFieldValuePair parameters		
Parameter	**Format**	**Description**
templateName	String	The name of the template.
attributeName	String	The name of the raw attribute.
fieldExpression	String	The field expression of the field-value pair.
expressionValue	String	The value expression of the field-value pair.
comboID	Integer	A unique integer that no other field-value pair has for this template/attribute/instance combination.
instanceName	String	The name of the instance.

Example

The following example creates the new field-value pair of the field `myfieldexp=` with value 2 for the attribute `IncomingStatusRule2` identification for the instance `my_services2`.

```
radshell>
addInstanceIDFieldValuePair("mynewtemplate2","IncomingStatusRule2","myfieldexp=","2",1,"my_servi
ces2");
```

clearAllInstanceIDFieldValuePairs

Use the `clearAllInstanceIDFieldValuePairs` function to clear out all `instanceid` field-value pairs for the instance.

Syntax

This function has the following syntax:

```
clearAllInstanceIDFieldValuePairs("instanceName");
```

Parameters

This function has the following parameters.

Table 69. clearAllInstanceIDFieldValuePairs parameters		
Parameter	**Format**	**Description**
instanceName	String	Name of the instance.

Example

The following example clears out all field-value pairs for the service instance `my_services2`.

```
radshell> clearAllInstanceIDFieldValuePairs("my_services2");
```

updateAttributeInstanceName

Use the updateAttributeInstanceName function to update the attribute name of an instance.

Syntax

This function has the following syntax:

```
updateAttributeInstanceName("templateName",
"attributeName",
"InstanceNames");
```

Parameters

This function has the following parameters.

Table 70. updateAttributeInstanceName parameters

Parameter	Format	Description
templateName	String	Name of the template.
attributeName	String	Attribute name to update.
InstanceNames	String	The name of the instances in which to change the attribute name. **Note:** Separate instance names by a comma (,).

setStartingInstance

Use the setStartingInstance function to set the starting instance to display for a user or group.

Syntax

This function has the following syntax:

```
setStartingInstance("userOrGroupType",
"userOrGroupName",
"instanceName");
```

Parameters

This function has the following parameters.

Table 71. setStartingInstance parameters

Parameter	Format	Description
userOrGroupType	String	User or group type. This should be USER or GROUP.
userOrGroupName	String	Name of the user or group.
instanceName	String	The instance name to show when the user or group logs in.

Example

The following example sets the starting instance to display WebFarm27 for the group type WebAdmins.

```
radshell> setStartingInstance(GROUP, "WebAdmins", "WebFarm27");
```

setTemplateForInstance

Use the `setTemplateForInstance` function to associate an existing service to an existing template.

Syntax

This function has the following syntax:

```
setTemplateForInstance("serviceName",
"templateName");
```

Parameters

This function has the following parameters.

Table 72. `setTemplateForInstance` parameters

Parameter	Format	Description
serviceName	String	Name of the service.
templateName	String	Name of the template.

Example

The following example associates the service `webserver1` to the template `WebFarm`.

```
radshell> setTemplateForInstance("webserver1","WebFarm");
```

resetServiceInstancePrivilege

Use the `resetServiceInstancePrivilege` function to reset the service instance privilege for users or groups. The service instance will inherit privileges from template.

Syntax

This function has the following syntax:

```
resetServiceInstancePrivilege("userOrGroupType",

"serviceInstanceName",
"privilege");
```

Parameters

This function has the following parameters.

Table 73. `resetServiceInstancePrivilege` parameters

Parameter	Format	Description
userOrGroupType	String	User or group type. This should be USER or GROUP.
serviceInstanceName	String	Name of the service instance.
privilege	String	Privilege to reset.

Example

The following example resets the service instance `WebFarm1` privilege for the user to `tbsmServiceAdmin`.

```
radshell> resetServiceInstancePrivilege(USER, "WebFarm1", "tbsmServiceAdmin");
```

The following example resets the service instance `WebFarm2` privilege for the group to `tbsmViewService`.

```
radshell> resetServiceInstancePrivilege(GROUP, "WebFarm2", "tbsmViewService");
```

renameInstance

Use the `renameInstance` function to change the name of an instance.

Syntax

This function has the following syntax:

```
renameInstance("oldName",
"newName");
```

Parameters

This function has the following parameters.

Table 74. `renameInstance` parameters

Parameter	Format	Description
oldName	String	Current name of the instance.
newName	String	New name for the instance.

printTopLevelInstances

Use the `printTopLevelInstances` function to print a list of top level instances.

Syntax

This function has the following syntax:

```
printTopLevelInstances();
```

Parameters

This function has no parameters.

deleteAllInstances

Use the `deleteAllInstances` function to delete all instances in TBSM.

Syntax

This function has the following syntax:

```
deleteAllInstances();
```

Parameters

This function has no parameters.

addPolicyDependencyAttributeToTemplate

Use the addPolicyDependencyAttributeToTemplate function to create a new policy rule.

Syntax

This function has the following syntax:

```
addPolicyDependencyAttributeToTemplate("parentTemplateName",
"childTemplateName",
"childAttributeName",
"ruleName",
"policyName",
"policyScript",
"policyType",
isTextRule true/false);
```

Parameters

This function has the following parameters.

Table 75. addPolicyDependencyAttributeToTemplate parameters

Parameter	Format	Description
parentTemplateName	String	Parent template name.
childTemplateName	String	Child template name.
childAttributeName	String	Child attribute name.
ruleName	String	Name of the rule.
policyName	String	Name of the policy to create.
policyScript	String	The policy script.
policyType	String	Optional. The policy type.
isTextRule	Boolean	Optional. Specifies whether this is a text rule. The valid values are true or false.

addRawEventAttributeToTemplate

Use the addRawEventAttributeToTemplate function to add a new threshold raw event attribute.

Syntax

This function has the following syntax:

```
addRawEventAttributeToTemplate("templateName",
"attributeName",
"eventDiscriminatorNames",
"instanceExpressionFields",
"badThresholdCustomFilterExpression",
"badThresholdFieldNames",
"badThresholdComparators",
"badThresholdFieldValues",
"marginalThresholdCustomFilterExpression",
"marginalThresholdFieldNames",
```

```
"marginalThresholdComparators",
"marginalThresholdFieldValues");
```

Parameters

This function has the following parameters.

Table 76. addRawEventAttributeToTemplate parameters

Parameter	Format	Description
templateName	String	Name of the template.
attributeName	String	Name of the attribute.
eventDiscriminatorNames	String array	List of event discriminator names.
instanceExpressionFields	String array	List of individual ObjectServer field names used to identify instances.
badThresholdCustomFilterExpression	String	The bad threshold custom filter expression.
badThresholdFieldNames	String	The bad threshold field names.
badThresholdComparators	String array	The bad threshold comparators.
badThresholdFieldValues	String array	The bad threshold field values.
marginalThresholdCustomFilterExpression	String array	The marginal threshold custom filter expression.
marginalThresholdFieldNames	String	The marginal threshold field names.
marginalThresholdComparators	String array	The marginal threshold comparators.
marginalThresholdFieldValues	String array	The marginal threshold field values.

Example

The following example sets up an attribute called `WebServerIsOK` where:

- it is affected by events where `Class` is `0`
- the instance is determined by the `Node` and `Location` fields
- it goes Bad if `AlertKey = 'PingFailure'` and `Severity` is `5`
- it goes Marginal if `AlertKey = 'PingFailure'` and `Severity > = 3`

```
radshell> addRawEventAttributeToTemplate("WebServer",
 "WebServerIsOK",
 new String[] { "Default Class(0)" },
 new String[] { "strip(Node, \"&\")", "Location" },
  null,
  new String[] { "AlertKey", "Severity" },
  new String[] { "=", ">=" },
  new String[] { "PingFailure", "5" },
  null,
  new String[] { "AlertKey", "Severity" },
  new String[] { "=", ">=" },
  new String[] { "PingFailure", "3" } );
```

The following example sets up same thing, but uses custom filters instead of separate fields. Although it is simpler to just use custom filters all the time, the downside is that if you use a custom filter, when you look at the attribute in the GUI, it will just show the filter as Advanced and will place the values in the table. This makes it harder to edit and to understand using the GUI later.

```
radshell> addRawEventAttributeToTemplate("WebServer",
 "WebServerIsOK",
 new String[] { "Default Class(0)" },
 new String[] { "strip(Node, \"&\")", "Location" },
 "AlertKey = 'PingFailure' AND Severity >= 5",

 null,
 null,
 null,

 "AlertKey = 'PingFailure' AND Severity >= 3",
 null,aaa
 null,
 null );
```

Note: If you use a custom filter, you should set all the other arrays for marginal/bad (whichever you are entering values for) to null and vice versa: if you are using the arrays to set the fields separately, you should set the custom filter to null.

addSumAttributeToTemplate

Use the addSumAttributeToTemplate function to add a sum attribute.

Syntax

This function has the following syntax:

```
addSumAttributeToTemplate("parentTemplateName",
"childTemplateName",
"childAttributeName",
"ruleName");
```

Parameters

This function has the following parameters.

Table 77. addSumAttributeToTemplate parameters

Parameter	Format	Description
parentTemplateName	String	The name of the parent template.
childTemplateName	String	The name of the child template.
childAttributeName	String	The name of the child attribute.
ruleName	String	The name of the rule.

addUserPreferencesForTemplate

Use the addUserPreferencesForTemplate function to add a field-value pair (user preference) for the template.

Syntax

This function has the following syntax:

```
addUserPreferencesForTemplate("templateName",
"fieldName",
"fieldValue");
```

Parameters

This function has the following parameters.

Table 78. addUserPreferencesForTemplate parameters		
Parameter	**Format**	**Description**
templateName	String	The name of the template.
fieldName	String	Name of the field to add to the template.
fieldValue	String	Value to which to set the field.

createDummyParentForTemplate

Use the createDummyParentForTemplate function to create a dummy parent for the template.

Syntax

This function has the following syntax:

```
createDummyParentForTemplate("template");
```

Parameters

This function has the following parameters.

Table 79. createDummyParentForTemplate parameters		
Parameter	**Format**	**Description**
template	String	The name of the template.

deleteAttributeFromTemplate

Use the deleteAttributeFromTemplate function to delete an attribute from the template.

Syntax

This function has the following syntax:

```
deleteAttributeFromTemplate("templateName",
"attributeName");
```

Parameters

This function has the following parameters.

Table 80. deleteAttributeFromTemplate parameters		
Parameter	**Format**	**Description**
templateName	String	The name of the template.
attributeName	String	The name of the attribute to delete.

Example

The following example deletes the IncomingStatusRule4 attribute from the template MyWebServer.

```
radshell> deleteAttributeFromTemplate("MyWebServer","IncomingStatusRule4");
```

listAllTemplates

Use the listAllTemplates function to list all the templates.

Syntax

This function has the following syntax:

```
listAllTemplates();
```

Parameters

There are no parameters for this function.

removeSlaForTemplate

Use the removeSlaForTemplate function to remove an SLA from a template.

Syntax

This function has the following syntax:

```
removeSlaForTemplate("templateName",
"slaName");
```

Parameters

This function has the following parameters.

Table 81. removeSlaForTemplate parameters

Parameter	Format	Description
templateName	String	The template name.
parameterName	String	The SLA to be removed.

Example

The following example removes the SLA WebFarm from the template WebFarmSlaGold.

```
radshell> removeSlaForTemplate("WebFarm","WebFarmSlaGold");
```

renameSlaForTemplate

Use the renameSlaForTemplate function to rename the SLA of a template.

Syntax

This function has the following syntax:

```
renameSlaForTemplate("templateName",
"oldSlaName",
"newSlaName");
```

Parameters

This function has the following parameters.

Table 82. renameSlaForTemplate parameters		
Parameter	**Format**	**Description**
templateName	String	The template name.
oldSlaName	String	The SLA to be renamed.
newSlaName	String	New name for the SLA.

Example

The following example renames the SLA of the template `WebFarm` from `WebFarmSlaGold` to `WebFarmSlaSilver`.

```
radshell> renameSlaForTemplate("WebFarm","WebFarmSlaGold","WebFarmSlaSilver");
```

setTemplateDescription

Use the `setTemplateDescription` function to set the description for an existing template with the specified template name.

Syntax

This function has the following syntax:

```
setTemplateDescription("templateName",
"description");
```

Parameters

This function has the following parameters.

Table 83. setTemplateDescription parameters		
Parameter	**Format**	**Description**
templateName	String	The template name.
description	String	The description to set for the template.

Example

The following example sets the description for the template `WebFarm` to `Web Farm Template`.

```
radshell> setTemplateDescription("WebFarm","Web Farm Template");
```

setTemplateForInstance

Use the `setTemplateForInstance` function to associate an existing service to an existing template.

Syntax

This function has the following syntax:

```
setTemplateForInstance("serviceName",
"templateName");
```

Parameters

This function has the following parameters.

Table 84. setTemplateForInstance parameters		
Parameter	**Format**	**Description**
serviceName	String	Name of the service.
templateName	String	Name of the template.

Example

The following example associates the service `webserver1` to the template `WebFarm`.

```
radshell> setTemplateForInstance("webserver1","WebFarm");
```

updatePercentageOfChildrenDependencyAttributeToTemplate

Use the `updatePercentageOfChildrenDependencyAttributeToTemplate` function to update a percentage of children in a `State` dependency attribute.

Note: You can specify a new value, or `" "` to indicate no change, for all parameters other then the `isChildInstancePropagation`, `badPercentageThreshold`, and `marginalPercentageThreshold` parameter.

Syntax

This function has the following syntax:

```
updatePercentageOfChildrenDependencyAttributeToTemplate("parentTemplateName",
"childTemplateName",
"ruleName",
"newRuleName",
"childThresholdState",
badPercentageThreshold,
marginalPercentageThreshold,
"setChildInstancePropagation",
"weightProperty",
weightDefaultValue);
```

Parameters

This function has the following parameters.

Table 85. updatePercentageOfChildrenDependencyAttributeToTemplate parameters		
Parameter	**Format**	**Description**
parentTemplateName	String	Required. The name of the parent template.
childTemplateName	String	The name of the child template. Specify `" "` for no change.
ruleName	String	Required. The rule name.

Table 85. updatePercentageOfChildrenDependencyAttributeToTemplate parameters (continued)		
Parameter	**Format**	**Description**
newRuleName	String	New rule name. Specify "" for no change.
childThresholdState	String	The childThresholdState parameter can be set to the following valid values: • Either • Bad • Marginal • All **Note:** This parameter is case sensitive. Specify 0 for no change.
badPercentageThreshold	Integer	The percentage of children that must be bad in order for this attribute to be Bad. Specify 0 for no change.
marginalPercentageThreshold	Integer	The percentage of children that must be marginal in order for this attribute to be Marginal. Specify 0 for no change.
setChildInstancePropagation	String	Indicates whether the status is dependent on the child status. This parameter takes the following values: yes: Status is dependent on the child status NetOverallAttribute. no: Status is dependent on the OverallAttribute of the specified child template. **Note:** If this parameter is set to any value other than yes or no, then no change is made to the attribute.
weightProperty	String	Name of the property that has a weighting factor. Specify "" for no change.
weightDefaultValue	Integer	Weight value for the weightProperty. Specify 0 for no change.

Example

The following example updates the following properties for the PercentRule rule of the WebFarm template:

- Changes the value set for the childTemplateName property to WebServer.
- Changes the value set for the childThresholdState property to Bad.
- Changes the value set for the weightProperty property to weight_property_name.

The remaining properties will remain unchanged.

```
radshell> updatePercentageOfChildrenDependencyAttributeToTemplate("WebFarm", "WebServer",
"PercentRule", "", "Bad", 0, 0, "", "weight_property_name", 0);
```

updatePolicyDependencyAttributeToTemplate

Use the updatePolicyDependencyAttributeToTemplate function to update the dependency attribute of a policy rule.

Syntax

This function has the following syntax:

```
updatePolicyDependencyAttributeToTemplate("parentTemplateName",
"childTemplateName",
"childAttributeName",
"ruleName",
"policyName",
"policyScript",
"policyType",);
```

Parameters

This function has the following parameters.

Table 86. updatePolicyDependencyAttributeToTemplate parameters

Parameter	Format	Description
parentTemplateName	String	Required. The name of the parent template.
childTemplateName	String	The name of the child template. Specify "" for no change.
childAttributeName	String	Dependency attribute of the policy rule.
ruleName	String	Required. The rule name.
policyName	String	The name of the policy.
policyScript	String	The name of the policy.
policyType	String	The policy type.

Example

The following example sets the following values for the WebServer child template of the WebFarm template:

- Sets the childAttributeName parameter to IncomingStatusRule_2.
- Sets the ruleName parameter to PercentRule.
- Sets the policyName parameter to PolicyFetcher.
- Sets the policyScript parameter to PolicyFetcherScript.
- Sets the policyType parameter to PolicyType_A.

```
radshell> updatePolicyDependencyAttributeToTemplate("WebFarm", "WebServer",
"IncomingStatusRule_2", "PercentRule", "PolicyFetcher", "PolicyFetcherScript", "PolicyType_A");
```

updateWorstChildDependencyAttributeToTemplate

Use the updateWorstChildDependencyAttributeToTemplate function to update a `Worst State` of all children in a `State` dependency attribute.

Note: You can specify a new value, or `""` to indicate no change, for all parameters other then the `isChildInstancePropagation`, `badPercentageThreshold`, and `marginalPercentageThreshold` parameter.

Syntax

This function has the following syntax:

```
updateWorstChildDependencyAttributeToTemplate("parentTemplateName",
"childTemplateName",
"ruleName",
"newRuleName",
"badThreshold,"
"marginalThreshold,"
"setChildInstancePropagation");
```

Parameters

This function has the following parameters.

Table 87. updateWorstChildDependencyAttributeToTemplate parameters

Parameter	Format	Description
parentTemplateName	String	Required. The name of the parent template.
childTemplateName	String	The name of the child template. Specify `""` for no change.
ruleName	String	Required. The rule name.
newRuleName	String	New rule name. Specify `""` for no change.
badThreshold	Integer	This should be Bad or `Marginal`. Specify `0` for no change.
marginalThreshold	Integer	This should be Bad or `Marginal`. Specify `0` for no change.
setChildInstancePropagation	String	Indicates whether the status is dependent on the child status. This parameter takes the following values: `yes`: Status is dependent on the child status `NetOverallAttribute`. no: Status is dependent on the `OverallAttribute` of the specified child template. **Note:** If this parameter is set to any value other than `yes` or no, then no change is made to the attribute.

Example

The following example updates the following properties for the `PercentRule` rule of the `WebFarm` template:

- Changes the value set for the `childTemplateName` property to `WebServer`.
- Changes the value set for the `childThresholdState` property to `Bad`.

```
radshell> updateWorstChildDependencyAttributeToTemplate("WebFarm", "WebServer", "PercentRule",
"", "Bad", "", "");
```

addNumericFunctionsPolicy

Use the `addNumericFunctionsPolicy` function to add the numeric attributes to a policy script.

Syntax

This function has the following syntax:

```
addNumericFunctionsPolicy("policyScript");
```

Parameters

This function has the following parameters.

Table 88. addNumericFunctionsPolicy parameters

Parameter	Format	Description
policyScript	String	Policy script to which to add the new functions.

Example

The following example adds the numeric attributes function to the policy script `PolicyScript2`.

```
radshell> addNumericFunctionsPolicy("PolicyScript2");
```

countOfChildren

Use the `countOfChildren` function to display the total number of children for a given instance.

Syntax

This function has the following syntax:

```
countOfChildren("instanceName");
```

Parameters

This function has the following parameters.

Table 89. countOfChildren parameters

Parameter	Format	Description
parameterName	String	Name of the instance over which to run the function.

Example

The following example displays the total number of children for the instance `WebFarm`.

```
radshell> countOfChildren("WebFarm");
```

createSchedulesIfNecessary

Use the `createSchedulesIfNecessary` function to export schedules for a set of instances.

Syntax

This function has the following syntax:

```
createSchedulesIfNecessary("instancesToExport",
"pw");
```

Parameters

This function has the following parameters.

Table 90. `createSchedulesIfNecessary` parameters

Parameter	Format	Description
instancesToExport	String	The instances from which to export the schedules.
pw	String	The PrinterWriter to use.

invalidateServiceName

Use the `invalidateServiceName` function to invalidate a service instance by its service instance name.

Syntax

This function has the following syntax:

```
invalidateServiceName("serviceInstanceName");
```

Parameters

This function has the following parameters.

Table 91. `invalidateServiceName` parameters

Parameter	Format	Description
serviceInstanceName	String	Name of the service instance to be invalidated.

Example

The following example invalidates the service instance with the name `webserver1`.

```
radshell> invalidateServiceName("webserver1");
```

listServicesWithSpecialCharacters

Use the `listServicesWithSpecialCharacters` function to list the names of services containing special characters.

Syntax

This function has the following syntax:

```
listServicesWithSpecialCharacters();
```

Parameters

This function has no parameters.

Example

The following example lists the names of services containing special characters.

```
radshell> listServicesWithSpecialCharacters();
```

printStatusOfRule

Use the `printStatusOfRule` function to print the status of a rule for an instance.

Syntax

This function has the following syntax:

```
printStatusOfRule("instanceName",
"templateName"
"ruleName");
```

Parameters

This function has the following parameters.

Table 92. `printStatusOfRule` parameters

Parameter	Format	Description
instanceName	String	Instance whose rule status you want to print.
templateName	String	Template from rule status you want to print.
ruleName	String	Name of the rule whose status you want to print.

Example

The following example prints the status of the rule `my_rule` from the template `my_template` for the instance `my_services2`.

```
radshell> printStatusOfRule("my_services2", "my_template", "my_rule");
```

provisionNewDNSMon

Use the `provisionNewDNSMon` function to provision a new DNS Monitor.

Note: This method does not require that previous ISM rules be created in RAD.

Syntax

This function has the following syntax:

```
provisionNewDNSMon("ismServer",
"ismPort",
"ismUserName",
"ismPassword",
"profile",
"server",
"port",
"hostname",
"localip",
"timeout",
"pollInterval",
"retestNumber",
"retestInterval",
"recursivelookups",
"retries",
"description");
```

Parameters

This function has the following parameters.

Table 93. provisionNewDNSMon parameters

Parameter	Format	Description
ismServer	String	Hostname of the ISM Server to connect to.
ismPort	String	Port that the ISM Server is running on.
ismUserName	String	Username to use when logging into the ISM Server.
ismPassword	String	Password to use when logging into the ISM Server.
profile	String	Profile to add this monitor to on the ISM Server.
server	String	DNS Server to connect to.
port	String	Port the DNS server is running on.
hostname	String	Hostname to resolve on the DNS Server.
localip	String	IP address that the hostname should resolve to.
timeout	String	Timeout in seconds. The default is 10.
pollInterval	String	Time in seconds between polls. The default is 600.
retestNumber	String	The number of retests to try on timeout.
retestInterval	String	The time (in seconds) to wait between retries.

Table 93. provisionNewDNSMon parameters (continued)		
Parameter	**Format**	**Description**
recursivelookups	String	Indicates whether recursive lookup is performed. This parameter takes the following values: yes: Recursive lookup is performed. no: Recursive lookup is not performed.
retries	String	Number of retries available on failure.
description	String	Description field for administration purposes to describe the monitor.

provisionNewHTTPMon

Use the provisionNewHTTPMon function to provision a new HTTP monitor on the ISM Server.

Note: This method does not require that previous ISM rules be created in RAD.

Syntax

This function has the following syntax:

```
provisionNewHTTPMon("ismServer",
"ismPort",
"ismUserName",
"ismPassword",
"host",
"profile",
"port",
"page",
"timeout",
"pollInterval",
"retestNumber",
"retestInterval",
"description");
```

Parameters

This function has the following parameters.

Table 94. provisionNewHTTPMon parameters		
Parameter	**Format**	**Description**
ismServer	String	Hostname of the ISM Server to connect to.
ismPort	String	Port that the ISM Server is running on.
ismUserName	String	Username to use when logging into the ISM Server.
ismPassword	String	Password to use when logging into the ISM Server.
host	String	Host to monitor HTTP on. This can be a hostname or an IP address.
profile	String	Profile to add this monitor to on the ISM Server.
port	String	Port the DNS server is running on.

Table 94. provisionNewHTTPMon parameters (continued)		
Parameter	**Format**	**Description**
page	String	Page to monitor. The default is "/".
timeout	String	Timeout in seconds. The default is 10.
pollInterval	String	Time in seconds between polls. The default is 600.
retestNumber	String	The number of retests to try on timeout.
retestInterval	String	The time (in seconds) to wait between retries.
description	String	Description field for administration purposes to describe the monitor.

provisionNewHTTPSMon

Use the provisionNewHTTPSMon function to provision a new HTTP monitor on the ISM Server.

Note: This method does not require that previous ISM rules be created in RAD.

Syntax

This function has the following syntax:

```
provisionNewHTTPSMon("ismServer",
"ismPort",
"ismUserName",
"ismPassword",
"host",
"profile",
"port",
"page",
"timeout",
"pollInterval",
"retestNumber",
"retestInterval",
"description");
```

Parameters

This function has the following parameters.

Table 95. provisionNewHTTPSMon parameters		
Parameter	**Format**	**Description**
ismServer	String	Hostname of the ISM Server to connect to.
ismPort	String	Port that the ISM Server is running on.
ismUserName	String	Username to use when logging into the ISM Server.
ismPassword	String	Password to use when logging into the ISM Server.

Table 95. provisionNewHTTPSMon parameters (continued)		
Parameter	**Format**	**Description**
host	String	Host to monitor HTTP on. This can be a hostname or an IP address.
profile	String	Profile to add this monitor to on the ISM Server.
port	String	Port the DNS server is running on.
page	String	Page to monitor. The default is "/".
timeout	String	Timeout in seconds. The default is 10.
pollInterval	String	Time in seconds between polls. The default is 600.
retestNumber	String	The number of retests to try on timeout.
retestInterval	String	The time (in seconds) to wait between retries.
description	String	Description field for administration purposes to describe the monitor.

provisionNewICMPMon

Use the provisionNewICMPMon function to provision a new ICMP Monitor.

Note: This method does not require that previous ISM rules be created in RAD.

Syntax

This function has the following syntax:

```
provisionNewICMPMon("ismServer",
"ismPort",
"ismUserName",
"ismPassword",
"profile",
"hostname",
"numberofpackets"
"packetinterval"
"packetsize"
"timeout",
"pollInterval",
"retestNumber",
"retestInterval",
"retries",
"description");
```

Parameters

This function has the following parameters.

Table 96. provisionNewICMPMon parameters		
Parameter	**Format**	**Description**
ismServer	String	Hostname of the ISM Server to connect to.
ismPort	String	Port that the ISM Server is running on.
ismUserName	String	Username to use when logging into the ISM Server.
ismPassword	String	Password to use when logging into the ISM Server.
profile	String	Profile to add this monitor to on the ISM Server.
hostname	String	Hostname to resolve on the DNS Server.
numberofpackets	String	Number of packets to send, The default is 5.
packetinterval	String	Time (in seconds) between packets, The default is 1.
packetsize	String	Size (in bytes) of each packet sent. The default is 64.
timeout	String	Timeout (in seconds) . The default is 10.
pollInterval	String	Time in seconds between polls. The default is 600.
retestNumber	String	The number of retests to try on timeout. The default is 0.
retestInterval	String	The time (in seconds) to wait between retries. The default is 0.
retries	String	Number of retries on timeout 0.
description	String	Description field for administration purposes to describe the monitor.

provisionNewLDAPMon

Use the provisionNewLDAPMon function to provision a new LDAP Monitor.

Note: This method does not require that previous ISM rules be created in RAD.

Syntax

This function has the following syntax:

```
provisionNewLDAPMon("ismServer",
"ismPort",
"ismUserName",
```

```
"ismPassword",
"profile",
"server",
"port",
"searchbase",
"filter",
"timeout",
"pollInterval",
"retestNumber",
"retestInterval",
"authtype",
"sasl",
"ismUserName",
"ismPassword",
"description");
```

Parameters

This function has the following parameters.

Table 97. `provisionNewLDAPMon` parameters

Parameter	Format	Description
ismServer	String	Hostname of the ISM Server to connect to.
ismPort	String	Port that the ISM Server is running on.
ismUserName	String	Username to use when logging into the ISM Server.
ismPassword	String	Password to use when logging into the ISM Server.
profile	String	Profile to add this monitor to on the ISM Server.
server	String	LDAP server to monitor.
port	String	Port the LDAP server is running on.
searchbase	String	LDAP search base to monitor. For example `ou=People,dc=Micromuse`
filter	String	Filter to use. For example `uid=jdoe`.
timeout	String	Timeout in seconds. The default is `10`.
pollInterval	String	Time in seconds between polls. The default is `600`.
retestNumber	String	The number of retests to try on timeout. The default as `0`.
retestInterval	String	The time (in seconds) to wait between retries. The default as `0`.
authtype	String	Authentication type to use. The default as `SIMPLE`.

Table 97. provisionNewLDAPMon parameters (continued)		
Parameter	**Format**	**Description**
sasl	String	Simple authentication and security layer.
username	String	LDAP username to use when querying the LDAP server.
password	String	LDAP password to use when querying the LDAP server.
description	String	Description field for administration purposes to describe the monitor.

removeServiceRelationship

Use the `removeServiceRelationship` function to remove the link between the instances with instance names of the parent and child parameters.

Syntax

This function has the following syntax:

```
removeServiceRelationship("parent",
"child");
```

Parameters

This function has the following parameters.

Table 98. removeServiceRelationship parameters		
Parameter	**Format**	**Description**
parent	String	Name of the parent instance.
child	String	Name of the child instance.

setRelationshipAttribute

Use the `setRelationshipAttribute` function to set a relationship attribute between two instances.

Syntax

This function has the following syntax:

```
setRelationshipAttribute("instanceName",
"childName",
"attributeName",
"attributeValue");
```

Parameters

This function has the following parameters.

Table 99. setRelationshipAttribute parameters		
Parameter	**Format**	**Description**
instanceName	String	Name of the instance.

Table 99. setRelationshipAttribute parameters (continued)		
Parameter	**Format**	**Description**
childName	String	Name of the child.
attributeName	String	Name of the attribute.
attributeValue	String	Value to set the attribute to.

returnTopLevelInstances

Use the returnTopLevelInstances function to return the list of top level instances.

Syntax

This function has the following syntax:

```
returnTopLevelInstances();
```

Parameters

This function has no parameters.

Example

The following example returns the list of top level instances.

```
radshell> returnTopLevelInstances();
```

Time Window Analyzer commands

The Time Window Analyzer commands allow you to add, enable, modify, and delete metric rules for the Time Window Analyzer window.

While much of the system-wide configuration of the Metric Collection feature is administered through its property file, the collection of each particular metric is controlled through RAD shell commands.

addMetricMeta

Use the addMetricMeta command to add a numeric rule as a metric for the Time Window Analyzer.

Syntax

This function has the following syntax:

```
addMetricMeta("templatename.ruleName", "metricDisplayName",
 "metricDescription", metricFrequency,
 metricEnabled true false,metricMarkerEnabled true false)
```

Parameters

This function has the following parameters:

Table 100. addMetricMeta function parameters		
Parameter	**Format**	**Description**
templatename.ruleName	String	The name of the rule you want to add in the form of *templatename.rulename*. This is the unique rule name and service template name for a rule and template that was previously configured in TBSM. When a metric definition is being added, radshell issues a warning if the template name does not exist. The metric meta definition is still added.
metricDisplayName	String	Optional. The Time Window Analyzer display name of the metric you want to add. The default is the value of *templatename.ruleName*.
metricDescription	String	Optional. If you want to include a description of the rule, enter it for this parameter.
metricFrequency	Integer	The time period in seconds between metric history data points. The default for this value is `900` seconds (15 minutes).
metricEnabled	Boolean	If true, the metric data is being collected in the metric history data store. This value defaults to `true`.
metricMarkerEnabled	Boolean	If true, markers are created whenever a service status changes. This value defaults to `false`.

Example

This example sets up a metric called `OSRegion.RegionTicketsSum` that adds a metric for the `RegionTicketsSum` rule in the `OSRegion` service template. The display name is set to `RegionTickets`.

1. The *template.rule* name is `OSRegion.RegionTicketsSum`.
2. The *metricDisplayName* is `RegionTickets`.
3. The *metricDescription* is blank.
4. The *metricFrequency* is `900` seconds.
5. The *metricEnabled* parameter is set to true.
6. The *metricMarkerEnabled* is set to true.

```
[1]   addMetricMeta("OSRegion.RegionTicketsSum",
[2]                       "RegionTickets",
[3]                       "",
[4]                       900,
[5]                       true,
[6]                       true);
```

enableMetricCollection

Use the `enableMetricCollection` function to start data collection for a Time Window Analyzer metric.

Syntax

You can enable data collection for a previously defined metric with this command.

This function has the syntax:

enableMetricCollection ("*metricName*");

Parameter

The only parameter for this function is the metric name.

Example

This example enables data collection for a metric called OSRegion.RegionTicketsSum.

```
enableMetricCollection ("OSRegion.RegionTicketsSum");
```

disableMetricCollection

Use the disableMetricCollection function to stop data collection for a Time Window Analyzer metric.

Syntax

You can disable metric collection for a previously defined metric with this command.

This function has the syntax:

disableMetricCollection ("*metricName*");

Parameter

The only parameter for this function is the metric name.

Example

This example disables data collection for a metric called OSRegion.RegionTicketsSum.

```
disableMetricCollection ("OSRegion.RegionTicketsSum");
```

setHeartBeatPeriodAttributeToTemplate

Use the setHeartBeatPeriodAttributeToTemplate function to set the time period threshold that changes the service instance status to unknown.

Syntax

setHeartBeatPeriodAttributeToTemplate("*TemplateName*", *timeperiod*);

Parameter

This function has the following parameters.

Table 101. setHeartBeatPeriodAttributeToTemplate function parameters

Parameter	Format	Description
TemplateName	String	The name of the template where you want to set a heart beat time period.
timeperiod	Int	The heart beat time period threshold in seconds.

Example

The following example sets the heartbeat period to 300 seconds (5 minutes) for the service template called WebServer.

```
setHeartBeatPeriodAttributeToTemplate("WebServer", "300");
```

disableMetricMarkers

Use the disableMetricMarkers function to stop marker creation for a Time Window Analyzer metric.

Syntax

This function has a similar syntax for each parameter:

```
disableMetricMarkers ("metricName");
```

Parameter

The only parameter for this function is the metric name.

Example

This example disables creation for a metric called OSRegion.RegionTicketsSum.

```
    disableMetricMarkers ("OSRegion.RegionTicketsSum");
```

enableMetricMarkers

Use the enableMetricMarkers function to start marker creation for a Time Window Analyzer metric.

Syntax

This function has the syntax:

```
enableMetricMarkers ("metricName");
```

Parameter

The only parameter for this function is the metric name.

Example

This example enables marker creation for a metric called OSRegion.RegionTicketsSum.

```
    enableMetricMarkers ("OSRegion.RegionTicketsSum");
```

disableServiceMetricCollection

Use the disableServiceMetricCollection function to stop metric collection for a specific TBSM service instance.

Syntax

You specify the metric and TBSM service instance name.

This function has the syntax:

```
disableServiceMetricCollection ("metricName","serviceName");
```

Parameters

The parameters for this function are described in this table.

Table 102. disableServiceMetricCollection

Parameter	Format	Description
metricName	String	The metric where you want to disable metric data collection.
serviceName	String	The name of the TBSM service instance where you want to disable data collection.

Example

This example disables data collection for a metric called `OSRegion.RegionTicketsSum` and a service named `AsiaGetThereFast`.

```
disableServiceMetricCollection ("OSRegion.RegionTicketsSum","AsiaGetThereFast");
```

enableServiceMetricCollection

Use the `enableServiceMetricCollection` function to start metric collection for a specific TBSM service instance.

Syntax

You specify the metric and TBSM service instance name.

This function has the syntax:

`enableServiceMetricCollection ("`*`metricName`*`","`*`serviceName`*`");`

Parameters

The parameters for this function are described in this table.

Table 103. enableServiceMetricCollection

Parameter	Format	Description
metricName	String	The metric where you want to enable metric data collection.
serviceName	String	The name of the TBSM service instance where you want to enable data collection.

Example

This example enables data collection for a metric called `OSRegion.RegionTicketsSum` and a service named `AsiaGetThereFast`.

```
enableServiceMetricCollection ("OSRegion.RegionTicketsSum","AsiaGetThereFast");
```

disableServiceMetricMarkers

Use the `disableServiceMetricMarkers` function to stop marker creation for a specific TBSM service instance.

Syntax

You specify the metric and TBSM service instance name.

This function has the syntax:

`disableServiceMetricMarkers ("metricName","serviceName");`

Parameters

The parameters for this function are described in this table.

Table 104. disableServiceMetricMarkers function parameters		
Parameter	**Format**	**Description**
metricName	String	The metric where you want to disable marker creation.
serviceName	String	The name of the TBSM service instance where you want to disable marker creation.

Example

This example disables marker creation for a metric called `OSRegion.RegionTicketsSum` and a service named `AsiaGetThereFast`.

```
disableServiceMetricMarkers ("OSRegion.RegionTicketsSum","AsiaGetThereFast");
```

enableServiceMetricMarkers

Use the `enableServiceMetricMarkers` function to start marker creation for a specific TBSM service instance.

Syntax

This function has the syntax:

`enableServiceMetricMarkers ("metricName","serviceName");`

Parameters

This table describes the parameter for this function.

Table 105. enableServiceMetricMarkers function parameters		
Parameter	**Format**	**Description**
metricName	String	The metric where you want to enable marker data creation for the service.
serviceName	String	The name of the TBSM service instance where you want to enable marker creation.

Example

This example enables marker creation for a metric called `OSRegion.RegionTicketsSum` and a service named `AsiaGetThereFast`.

```
enableServiceMetricMarkers ("OSRegion.RegionTicketsSum","AsiaGetThereFast");
```

List metric data functions

Use the list functions to display parameters for a Time Window Analyzer metric.

Syntax

With these functions, you can list data about all metrics or a single metric.

You can run these list functions with the syntax:

```
listMetricMetaData();
listMetricMetaData("metricName");
listDisabledServicesMetricCollection();
listDisabledServicesMetricCollection("metricName");
listDisabledServicesMarkers();
listDisabledServicesMarkers("metricName");
```

List functions

This table describes the list functions:

Table 106. `listMetricMetaData` function parameters

Function	Description
listMetricMetaData	Shows data for all metrics or for a single metric you specify.
listDisabledServicesMetricCollection	Shows all of the services where metric data collection is disabled for any metric or all the services where collection is disabled for the single metric you specify as a parameter.
listDisabledServicesMarkers	Shows all of the services where metric marker creation is disabled for any metric or all the services where creation is disabled for the single metric you specify as a parameter.

Example

This example lists all the metric data configured for Time Window Analyzer.

```
  listMetricMetaData();

  Name=OSRegion.RegionTicketsSum
   Display Name=RegionTickets
   Description=
   Priority=0
   Type={"type":"unspecified"}
   Range Low=0.0
   Range High=100.0
   Frequency=900
   Enabled=true
   Marker Enabled=true


Name=OverallAttribute
  Display Name=Status Changes
  Description=TBSM Status
  Priority=0
  Type={"type":"ENUM","ENUM",{"0":"Good","3":"Marginal","5":"Critical"}}
  Range Low=0.0
  Range High=5.0
  Frequency=900
  Enabled=true
  Marker Enabled=true
```

exportMetricMetaData

Use the export functions to export Time Window Analyzer metric configuration data to a file.

The following functions export metric metadata and the services disabled for collection and marker creation for each metric. Use this function to back up your metric configuration.

By default, the export output is written to file `exportMetricMetaData.radsh`, in the directory `TBSM_HOME\export`. Optionally, you can specify an alternative file. If you specify `null` for the `metricName` parameter, all meta data is exported.

Syntax

This function has the syntax:

```
exportMetricMetaData ("metricName", "exportFileName",
exportDisabledServicesMetricCollection True|False,exportDisabledServicesMarkers
True|False);
```

Parameters

This function has the following parameters:

Table 107. exportMetricMetaData function parameters

Parameter	Format	Description
metricName	String	Optional. The name of the metric configuration you want to export to a file. If you do not specify a metric name, the default file name is exported.
exportFileName	String	Optional. The name of the file where you want to export metric configuration data. If you do not specify a file name, the data for the metric is exported to the file named `exportMetricMetaData.radsh` in the directory: `TBSM_HOME\export\`
exportDisabledServices MetricCollection	Boolean	Optional. This flag defaults to `true`. Set it to false if you don't want the exported data to include the disable functions for services that have been disabled for the collection of specific metrics.
exportDisabledServicesMarkers	Boolean	Optional. This flag defaults to `true`. Set it to false if you don't want the exported data to include the disable functions for services that have been disabled for marker creation for specific metrics.

Example

This example exports metric configuration data for a metric called `OSRegion.RegionTicketsSum` to a file named `myregionexport.radsh` in the directory: `\home\mydir\`.

```
  exportMetricMetaData ("OSRegion.RegionTicketsSum","\\home\\mydir
\\myregionexport.radsh");
```

Important: The double slashes (**** or **//**) are required. If you enter a single slash, it causes a parsing error in RAD shell.

Restoring from export file

You can use the exported metric meta data to restore your current system or to apply the data to another system. This is accomplished by providing the export file as input to the `rad_radshell` utility on the TBSM Data server to be updated.

This an example for a Windows system:

```
type exportMetricMetaData.radsh | %TBSM_HOME%\bin\rad_radshell
```

This an example for a UNIX system:

```
cat exportMetricMetaData.radsh | $TBSM_HOME/bin/rad_radshell
```

purgeMetricDBOnDemand

Use the purgeMetricDBOnDemand function to start the Metric Collection Purge Process.

Syntax

The **purgeMetricDBOnDemand** method asynchronously begins the Metric Collection Purge Process. If the scheduled purge process is in progress or soon to start, this method ends and lets the normal schedule execute.

This function has the syntax:

```
purgeMetricDBOnDemand();
```

Parameter

There are no parameters for this function.

updateMetric display name, description, and frequency

Use these three functions to update the display name, description, and frequency for a Time Window Analyzer metric.

Syntax

You can update a previously defined metric with these commands.

These functions require the metric name and a value for the specific update parameter.

```
updateMetricDisplayName("metricName", "DisplayName");
updateMetricDescription("metricName", "Description");
updateMetricFrequency("metricName", metricFrequency);
```

Update Functions

The updateMetric functions are described in this table.

Table 108. updateMetric functions

Function	Format	Description
updateMetricDisplayName	String	Specify the metric name and the display name.
updateMetricDescription	String	Specify the metric name and the description.
updateMetricFrequency	Integer	Specify the metric name and the update frequency.

Examples

This example changes the update frequency for a metric called OSRegion.RegionTicketsSum to 1800 seconds.

```
    updateMetricFrequency("OSRegion.RegionTicketsSum", 1800);
```

This example changes the update display name for a metric called OSRegion.RegionTicketsSum metric to RegionTickets.

```
updateMetricDisplayName("OSRegion.RegionTicketsSum", "RegionTickets");
```

revokeObjectPrivilege

Use the revokeObjectPrivilege function to remove permissions from a user or group for a given service instance or template.

Syntax

revokeObjectPrivilege("*userOrGroupType*",
"*userOrGroupName*",
"*objectType*",
"*objectName*",
"*privilege*");

Parameter

This function has the following parameters.

Table 109. revokeObjectPrivilege function parameters

Parameter	Format	Description
userOrGroupType	String	Specifies whether the privileges apply to a user or a group.
userOrGroupName	String	The name of the user or group you want to update.
objectType	String	The type of object where you want to revoke privileges.
objectName	String	The name of the object where you want to revoke privileges.
privilege	String	The privilege you want to revoke.

Example

The following example shows how to revoke user1's privileges to view service instances tagged with the WebFarm service template. The parameters are as follows:

1. The user type is set to USER.
2. The user name is user1.
3. The object type is set to TEMPLATE.
4. The service template name is set to WebFarm.
5. The privilege is set to tbsmViewTemplate.

```
[1]   revokeObjectPrivilege("USER",
[2]                         "user1",
[3]                         "TEMPLATE",
[4]                         "WebFarm",
[5]                         "tbsmViewTemplate");
```

grantPrivilegeToAllChildren

Use the grantPrivilegeToAllChildren function to give permissions to a user or group for a given service instance and its children.

Syntax

```
grantPrivilegeToAllChildren ("userOrGroupType", "userOrGroupName","objectName",
"privilege");
```

Parameters

This function has the following parameters.

Table 110. grantPrivilegeToAllChildren parameters

Parameter	Format	Description
userOrGroupType	String	Specifies whether the privileges apply to a user or a group. The valid values are USER and GROUP.
userOrGroupName	String	The name of the user or group that you want to update.
objectName	String	The name of the object to which you want to grant privileges.
privilege	String	The privilege you want to grant.

Example

The following example shows how you give user1 privileges to view service instances tagged with the WebFarm service template. The parameters in this example are set as follows:

1. The user or group type is set to USER.
2. The user name is user1.
3. The service template name is set to WebFarm.
4. The privilege is set to tbsmViewService.

```
[1]   grantPrivilegeToAllChildren("USER",
[2]                        "user1",
[3]                        "WebFarm",
[4]                        "tbsmViewService");
```

grantObjectPrivilege

Use the grantObjectPrivilege function to grants object instance privileges to users or groups.

Syntax

This function has the following syntax:

```
grantObjectPrivilege("userOrGroupType",
"userOrGroupName",
"objectType",
"objectName",
"privilege");
```

Parameters

This function has the following parameters.

Table 111. grantObjectPrivilege parameters		
Parameter	**Format**	**Description**
userOrGroupType	String	Indicates whether to grant privileges as a USER or a GROUP.
userOrGroupName	String	Name of the user or group.
objectType	String	Type of object to which privileges are being granted.
objectName	String	Name of the object to which privileges are being granted.
privilege	String	The privilege being granted.

grantPrivilegeToAllParents

Use the `grantPrivilegeToAllParents` function to grant an instance privilege (such as `tbsmViewService`) to an instance as well as all its parents recursively.

Syntax

This function has the following syntax:

```
grantPrivilegeToAllParents("userOrGroupType",
"userOrGroupName",
"objectType",
"objectName",
"privilege",
"alreadyProcessedInstanceNames");
```

Parameters

This function has the following parameters.

Table 112. grantPrivilegeToAllParents parameters		
Parameter	**Format**	**Description**
userOrGroupType	String	Indicates whether to grant privileges as a USER or a GROUP.
userOrGroupName	String	Name of the user or group.
objectType	String	Type of object to which privileges are being granted.
objectName	String	Name of the object to which privileges are being granted.
privilege	String	The privilege being granted.
alreadyProcessedInstanceNames	String	Name of the instance already processed.

listDisabledServicesMarkers

Use the `listDisabledServicesMarkers` function to list all services disabled for marker creation for a particular metric.

Syntax

This function has the following syntax:

```
listDisabledServicesMarkers("metricName");
```

Parameters

This function has the following parameters.

Table 113. `listDisabledServicesMarkers` parameters

Parameter	Format	Description
metricName	String	Metric for which to list services disabled for marker creation.

Example

The following example lists all services disabled for marker creation for the metric `OverallAttribute`.

```
radshell> listDisabledServicesMarkers("OverallAttribute");
```

listDisabledServicesMetricCollection

Use the `listDisabledServicesMetricCollection` function to list all services disabled for metric collection for all metrics.

Syntax

This function has the following syntax:

```
listDisabledServicesMetricCollection("metricName");
```

Parameters

This function has the following parameters.

Table 114. `listDisabledServicesMetricCollection` parameters

Parameter	Format	Description
metricName	String	Metric for which to list services disabled for metric collection.

Example

The following example lists all services disabled for metric collection for the metric `OverallAttribute`.

```
radshell> listDisabledServicesMetricCollection("OverallAttribute");
```

listMetricMetaData

Use the `listMetricMetaData` function to list metric meta data.

Syntax

This function has the following syntax:

```
listMetricMetaData("metricName");
```

Parameters

This function has the following parameters.

Table 115. `listMetricMetaData` parameters

Parameter	Format	Description
metricName	String	Metric for which to list meta data.

Example

The following example lists metadata for metric `OverallAttribute`.

```
radshell> listMetricMetaData("OverallAttribute");
```

updateMetricDescription

Use the `updateMetricDescription` function to update a metric's `Description` metadata.

Syntax

This function has the following syntax:

```
updateMetricDescription("metricName",
"metricDescription");
```

Parameters

This function has the following parameters.

Table 116. `updateMetricDescription` parameters

Parameter	Format	Description
metricName	String	Name of the metric.
metricDescription	String	Updated metadata description for the metric.

Example

The following example updates the `Description` metadata of the metric `OverallAttribute` to `TBSM Status`.

```
radshell> updateMetricDescription("OverallAttribute","TBSM Status");
```

updateMetricDisplayName

Use the updateMetricDisplayName function to update a metric's DisplayName metadata.

Syntax

This function has the following syntax:

```
updateMetricDisplayName("metricName",
"metricDisplayName");
```

Parameters

This function has the following parameters.

Table 117. updateMetricDisplayName parameters

Parameter	Format	Description
metricName	String	Name of the metric.
metricDisplayName	String	Updated display name for the metric.

Example

The following example updates the DisplayName metadata of the metric OverallAttribute to Status Change.

```
radshell> updateMetricDisplayName("OverallAttribute","Status Change");
```

updateMetricFrequency

Use the updateMetricFrequency function to update a metric's Frequency metadata.

Syntax

This function has the following syntax:

```
updateMetricFrequency("metricName",
metricFrequency);
```

Parameters

This function has the following parameters.

Table 118. updateMetricFrequency parameters

Parameter	Format	Description
metricName	String	Name of the metric.
metricFrequency	Integer	Updated frequency value for the metric.

Example

The following example updates the Frequency metadata of the metric OverallAttribute to 900.

```
radshell> updateMetricFrequency("OverallAttribute",900);
```

updateMetricPriority

Use the updateMetricPriority function to update a metric's Priority metadata.

Syntax

This function has the following syntax:

```
updateMetricPriority("metricName",
metricPriority);
```

Parameters

This function has the following parameters.

Table 119. updateMetricPriority parameters

Parameter	Format	Description
metricName	String	Name of the metric.
metricPriority	Integer	Updated priority value for the metric.

Example

The following example updates the Priority metadata of the metric OverallAttribute to 0.

```
radshell> updateMetricPriority("OverallAttribute",0);
```

updateMetricRangeHigh

Use the updateMetricRangeHigh function to update a metric's RangeHigh metadata.

Syntax

This function has the following syntax:

```
updateMetricRangeHigh("metricName",
metricRangeHigh);
```

Parameters

This function has the following parameters.

Table 120. updateMetricRangeHigh parameters

Parameter	Format	Description
metricName	String	Name of the metric.
metricRangeHigh	Integer	Updated upper ceiling value for the metric.

Example

The following example updates the RangeHigh metadata of the metric OverallAttribute to 5.0.

```
radshell> updateMetricRangeHigh("OverallAttribute",5.0);
```

updateMetricRangeLow

Use the updateMetricRangeLow function to update a metric's RangeLow metadata.

Syntax

This function has the following syntax:

```
updateMetricRangeLow("metricName",
metricRangeLow);
```

Parameters

This function has the following parameters.

Table 121. updateMetricRangeLow parameters

Parameter	Format	Description
metricName	String	Name of the metric.
metricRangeLow	Integer	Updated lower floor value for the metric.

Example

The following example updates the RangeLow metadata of the metric OverallAttribute to 0.

```
radshell> updateMetricRangeLow("OverallAttribute",0);
```

updateMetricType

Use the updateMetricType function to update a metric's Type metadata.

Syntax

This function has the following syntax:

```
updateMetricType("metricName",
"metricType");
```

Parameters

This function has the following parameters.

Table 122. updateMetricType parameters

Parameter	Format	Description
metricName	String	Name of the metric.
metricType	String	Updated type value for the metric.

Example

The following example updates the Type metadata of the metric OverallAttribute to C.

```
radshell> updateMetricType("OverallAttribute","C");
```

Audit logging commands

Records of service configuration changes, including the startup and shutdown of the TBSM Data server, are stored in audit logs. You can use the audit logs to assist you in understanding changes to TBSM configuration data and to more easily diagnose problems. As configuration changes are saved, audit records are created in the audit logs.

Note: The configuration steps outlined in this section can also be completed in the Dashboard Application Service Hub user interface. For more information, see the *TBSM Troubleshooting Guide*.

getAuditConfig

Use the `getAuditConfig` to display the audit file configuration.

Syntax

This function has the following syntax:

```
getAuditConfig();
```

Parameters

There are no parameters for this function.

Example

This example displays the audit file configuration:

```
getAuditConfig();
```

setAuditConfig

Use the `setAuditConfig` to set the audit file configuration.

Syntax

This function has the following syntax:

```
setAuditConfig( "auditPath", fileSize, fileCount);
```

Parameters

This function has the following parameters:

Table 123. `setAuditConfig` parameters

Parameter	Format	Description
auditPath	String	The path to the audit log location.
fileSize	Integer	The maximum size of the audit log in bytes.
fileCount	Integer	The maximum number of audit log files.

Example

The following example sets the path for the audit log file to the path provided. It sets the maximum size for each audit file to a maximum of 32 bytes, and the maximum number of audit files to 5.

```
setAuditConfig( "C:\\Program Files\\IBM\tivoli\tipv2\profiles\TBSMProfile\
logs\server1\audit", 32, 5);
```

setAuditingEnabled

Use the `setAuditingEnabled` to enable and disable auditing for the Data Server. This value is reset every time the Data Server is rebooted. By default, auditing is enabled.

Syntax

This function has the following syntax:

```
setAuditingEnabled(enable true false);
```

Parameters

This function has the following parameters:

Table 124. `setAuditingEnabled` parameters

Parameter	Format	Description
enable	Boolean	Enables and disables auditing on the Data server.

Example

The following example disables audit logging in the Data server.

```
setAuditingEnabled(false);
```

isAuditingEnabled

Use the `isAuditingEnabled` to displays whether auditing is enabled or not. The function returns `true` if auditing is enabled, otherwise `false` is returned.

Syntax

This function has the following syntax:

```
isAuditingEnabled();
```

Parameters

There are no parameters for this function.

Example

This example displays true or false to indicate whether auditing is enabled or not:

```
isAuditingEnabled();
```

Event Logging Filter commands

Event Logging Filter commands allow you to create and maintain event logging filters.

addUpdateEventLoggingFilterValue

Use the `addUpdateEventLoggingFilterValue` function to create or update an event logging filter field-value pair.

If the `fieldName` specified does not exist, it is added. If the `fieldName` specified already exists, its value is updated.

Note: The values are preserved in `eventloggingfilter` table.

Syntax

This function has the following syntax:

```
addUpdateEventLoggingFilterValue("fieldName",
"fieldValue");
```

Parameters

This function has the following parameters.

Table 125. addUpdateEventLoggingFilterValue parameters

Parameter	Format	Description
fieldName	String	The name of the field to add or update.
fieldValue	String	The value to set the field to.

Example

The following example either creates the a field called `FieldName_1` and sets it to `FieldValue_1`, *or* if `FieldName_1` already exists, it updates the value set for it to `FieldValue_1`.

```
radshell> addUpdateEventLoggingFilterValue("FieldName_1","FieldValue_1");
```

clearEventLoggingFilter

Use the `clearEventLoggingFilter` function to clear out all Event Logging Filters.

Syntax

This function has the following syntax:

```
clearEventLoggingFilter();
```

Parameters

This function has no parameters.

Example

The following example clears out all Event Logging Filters.

```
radshell> clearEventLoggingFilter();
```

displayEventLoggingFilter

Use the `displayEventLoggingFilter` function to display all fields in the Event Mapping Filter.

Syntax

This function has the following syntax:

```
displayEventLoggingFilter(");
```

Parameters

This function has no parameters.

Example

The following example displays all fields in the Event Mapping Filter.

```
radshell> displayEventLoggingFilter(");
```

removeEventLoggingFilterValue

Use the `removeEventLoggingFilterValue` function to remove an event logging filter.

Syntax

This function has the following syntax:

```
removeEventLoggingFilterValue("fieldName");
```

Parameters

This function has the following parameters.

Table 126. `removeEventLoggingFilterValue` parameters

Parameter	Format	Description
fieldName	String	Name of the field from which to remove the Event Logging Filter.

Miscellaneous commands

These are a set of commands that do not fit into the other categories.

doArtifactMerge

Use the `doArtifactMerge` function to ensure that an exception is not thrown when duplicate artifacts are created.

Syntax

This function has the following syntax:

```
doArtifactMerge(doArtifactDiffs);
```

Parameters

This function has the following parameters.

Table 127. `doArtifactMerge` parameters

Parameter	Format	Description
doArtifactDiffs	Boolean	If set to `true` an exception will not be thrown when duplicate artifacts are created.

Example

The following example ensures that an exception is not thrown when duplicate artifacts are created.

```
radshell> doArtifactMerge(true);
```

encryptMigratedPassword

Use the `encryptMigratedPassword` function to encrypt a TBSM 4.2.x password to a 6.1 password and append the pair to the properties file for migration use.

Syntax

This function has the following syntax:

```
encryptMigratedPassword("password",
"propsfilename");
```

Parameters

This function has the following parameters.

Table 128. `encryptMigratedPassword` parameters

Parameter	Format	Description
password	String	The password being migrated.
propsfilename	String	Name of the properties file to load and write the new pair to.

getMatchingEventIdentifiers

Use the `getMatchingEventIdentifiers` function to get the event identifiers for a given identifier.

Syntax

This function has the following syntax:

```
getMatchingEventIdentifiers(instanceid);
```

Parameters

This function has the following parameters.

Table 129. `getMatchingEventIdentifiers` parameters

Parameter	Format	Description
instanceid	Long	Identifier of the instance.

importInstancesFromProvisoBottomUpXMLDump

Use the `importInstancesFromProvisoBottomUpXMLDump` function to import Proviso exported XML models into RAD.

Syntax

This function has the following syntax:

```
importInstancesFromProvisoBottomUpXMLDump("xmlInputFileName");
```

Parameters

This function has the following parameters.

Table 130. importInstancesFromProvisoBottomUpXMLDump parameters		
Parameter	**Format**	**Description**
xmlInputFileName	String	File from which to import the instance's XML model exported from Proviso.

loadPolicyIPLFromDirectory

Use the `loadPolicyIPLFromDirectory` function to load the IPL policy files found in the specified directory.

Syntax

This function has the following syntax:

```
loadPolicyIPLFromDirectory("policyFileDirectory");
```

Parameters

This function has the following parameters.

Table 131. loadPolicyIPLFromDirectory parameters		
Parameter	**Format**	**Description**
policyFileDirectory	String	Directory containing policies to load. Use the UNIX style path separator (/) to avoid `rad_radshell` parsing errors. **Note:** Only files of type `ipl` will be processed.

persistifyTag

Use the `persistifyTag` function to persistify any uncommitted changes to the specified template.

Syntax

This function has the following syntax:

```
persistifyTag("templateName");
```

Parameters

This function has the following parameters.

Table 132. persistifyTag parameters		
Parameter	**Format**	**Description**
templateName	String	Template containing uncommitted changes that will be persistified.

Example

The following example persistifies any uncommitted changes to the template `WebFarm`.

```
radshell> persistifyTag("WebFarm");
```

replayTBSMEvent

Use the `replayTBSMEvent` function to replay a TBSM event.

Syntax

This function has the following syntax:

```
replayTBSMEvent();
```

Parameters

This function has no parameters.

Example

The following example replays a TBSM event.

```
radshell> replayTBSMEvent();
```

setCanvasViewerToUse

Use the `setCanvasViewerToUse` function to set the canvas viewer to use.

Valid canvas viewers are either `thin` or `full`).

Syntax

This function has the following syntax:

```
setCanvasViewerToUse("userOrGroupType",
"userOrGroupName",
"viewerTemplate");
```

Parameters

This function has the following parameters.

Table 133. `setCanvasViewerToUse` parameters

Parameter	Format	Description
userOrGroupType	String	Specify the type. This parameter takes the following values: USER or GROUP.
userOrGroupName	String	Name of the user or group.
viewerTemplate	String	Canvas viewer to use. This parameter takes the following values: THIN_VIEWER or FULL_VIEWER.

Example

The following example sets the root user to use the THIN_VIEWER canvas viewer.

```
radshell> setCanvasViewerToUse(USER, "root", THIN_VIEWER);
```

setTslaCheckbox

Use the setTslaCheckbox function to set the TSLA check box to true for a template.

Syntax

This function has the following syntax:

```
setTslaCheckbox("tagName");
```

Parameters

This function has the following parameters.

Table 134. setTslaCheckbox parameters

Parameter	Format	Description
tagName	String	Name of the template on which to set the TSLA checkbox to true.

Example

The following example sets the TSLA checkbox to true for the template ESDA_Hosts.

```
radshell> setTslaCheckbox("ESDA_Hosts");
```

waitForServerInitialization

Use the waitForServerInitialization function to wait until the service model has initialized.

Syntax

This function has the following syntax:

```
waitForServerInitialization();
```

Parameters

This function has no parameters.

Example

The following example waits until the service model has initialized.

```
radshell> waitForServerInitialization();
```

initiateSCRRefresh

Use the initiateSCRRefresh command to initiate a complete refresh of the TBSM data model data in the Service Component Repository (SCR). The processing of the refresh operation updates the SCR representations of services, attributes, and relationships that originated from the TBSM data model.

Syntax

This function has the following syntax:

```
initiateSCRRefresh();
```

Parameters

There are no parameters for this function.

Example

The following example initiates a refresh operation to refresh the TBSM data model data sent to the SCR:

```
initiateSCRRefresh();
```

For more information, see the *Synchronizing TBSM service model data with the SCR* section of the *TBSM Customization Guide*.

discoverHierarchy

Use `discoverHierarchy` to recursively discover the ESDA hierarchy from a given starting instance. This is equivalent to expanding the seed service in the Service Tree. When ESDA services are valid, running this command has no effect. Run this command after you have run the `invalidateServiceName` command, for the seed service, to run ESDA rules again.

Syntax

`discoverHierarchy("`*seedInstance*`");`

Parameter

This function has the following parameters:

Table 135. `discoverHierarchy` parameters

Parameter	Format	Description
seedInstance	String	The name of the seed service instance in the database.

Example

This example shows how to how to determine the ESDA hierarchy for a seed service instance called `Webserver`:

```
discoverHierarchy ("Webserver");
```

printChildren

Use `printChildren` to print the names of the children for the given parent instance. All child names are printed whether services are persisted or not.

Syntax

`printChildren("`*parentName*`");`

Parameter

This function has the following parameters:

Table 136. `printChildren` parameters

Parameter	Format	Description
parentName	String	The name of the parent service instance in the database.

Example

This example shows how to print the names of the child services for a service instance called `Webserver`

```
printChildren ("Webserver");
```

Creating Discovery Library Toolkit service templates

The script `BSM_Templates.radsh` is a Rad shell script that defines service templates for common TBSM services and is valid for all operating systems supported by TBSM. This script is located in the `$TBSM_HOME/install/` directory on UNIX and in the `%TBSM_HOME%\install\` directory on Windows.

About this task

Note: The Discovery Library Toolkit service templates are created when TBSM is installed. If your TBSM database becomes corrupted or if you need to reinstall it for any reason, you can use this procedure to recreate service templates for the Service Component Repository.

To create service templates, perform the following step:

Procedure

- Type the following command from a command prompt:

  UNIX `cat $TBSM_HOME/install/BSM_Templates.radsh | $TBSM_HOME/bin/rad_radshell`

  Windows `type %TBSM_HOME%\install\BSM_Templates.radsh | %TBSM_HOME%\bin\rad_radshell`

Opening a service from a URL

You can open a service directly from a URL, using this URL format: `https://`*`Tipserver:port`*`/ibm/action/launch?pageID=`*`unique page id`*. You can access information about service availability and service administration.

Before you begin

When entering the URL, you are prompted to log in to the Dashboard Application Service Hub console if you are not already logged in. After entering your user ID and password, your URL will open.

About this task

To open a service using a URL, perform the following steps:

Procedure

1. Enter a URL using the following format:

   ```
   https://TIPserver:port/ibm/action/launch?pageID=unique page id
   ```

   where *TIPserver* and *port* specify the location of the TBSM server that you want to access and *unique page id* is one of the following values:

   - `com.ibm.tbsm.navigationElement.desktop` (this opens the **Service Availability** page)
   - `com.ibm.tbsm.navigationElement.serviceAdmin` (this opens the **Service Administration** page)
2. Include one or more of the following parameters on the URL:

   - **`&ServiceInstanceID`**

- **&ServiceInstanceName**
- **&ManagedSystemName**
- **&Guid**
- **&MSSName** (required)
- **&SourceToken** (required)
- **&CDMClass** (optional with **&MSSName** and **&SourceToken**, but recommended)

Important: When entering the URL, you must specify the name of the service, *not* the display name.

These parameters are searched for in the order listed; the first parameter that is found is used for context reference, and the remainder are ignored.

In addition, the **Service Availability** page supports the **&View** parameter, which allows the view definition that is used in the ***Service Viewer*** to be changed from the default. This parameter can have a value of `BusinessImpact` or `BusinessImpactAll`.

Example

```
https://intwin4.tivlab.raleigh.ibm.com:16311/ibm/action/launch?
pageID=com.ibm.tbsm.navigationElement.desktop&ManagedSystemName=Primary:INTWIN3
:NT
```

```
https://intwin4.tivlab.raleigh.ibm.com:16311/ibm/action/launch?
pageID=com.ibm.tbsm.navigationElement.desktop&ServiceInstanceName=INTWIN1(54218
109879036F5AC021BADE2C53460)-ComputerSystem&View=BusinessImpactAll
```

Administering the Discovery Library Toolkit

The `enqueuecl.properties` and `xmltoolkitsvc.properties` files contain the properties that you can use to administer the Discovery Library Toolkit.

The Discovery Library Toolkit

The Discovery Library Toolkit enables TBSM discovery resources and to automatically build service models from Discovery Library data sources. These sources include: Tivoli Application Dependency Discovery Manager, Discovery Library books conforming to the common data model, Discovery Library books containing objects for an alternate namespace, the Discovery Library Toolkit API, or auto-pop objects.

Data discovered using the Discovery Library Toolkit can be enriched through notifications sent to Impact. This enriched data can then be used in the automatic building of the service model.

The TBSM database contains a set of SCR tables. TBSM uses an External Service Dependency Adapter (ESDA) rule to query these tables and creates new services based on the SCR data. The daemon or service is used to monitor the data source and add the data to the SCR tables.

In addition to mapping resources to templates, the Discovery Library Toolkit support also provides event mapping, which maps events received by IBM Tivoli Netcool/OMNIbus ObjectServer to resources discovered through the Discovery Library Toolkit.

Running the Discovery Library Toolkit

This topic outlines the steps to run the Discovery Library Toolkit.

Procedure

1. UNIX

Start the Discovery Library Toolkit daemon by running the **tbsmrdr_start.sh** script in the $TBSM_HOME/XMLtoolkit/bin directory.

Note for users running Linux 6: To add the daemon to the /etc/init.d directory so that the Discovery Library Toolkit is automatically restarted if the you restart the TBSM host, login as root and run the tbsmrdr_enable.sh script.

Note for users running Linux 7: Linux 7 has changed. It now uses systemd instead of init.d to manage system processes after booting. If you are running Linux 7, you must disable the XML toolkit daemon for init.d before running the tbsmrdr_start.sh and tbsmrdr_stop.sh scripts. To do this, login as root and run the tbsmrdr_disable.sh script.

2. Windows
   Start the Discovery Library Toolkit service from the **Services** window or by issuing the **net start ASICRToolkitSvc** command.
3. If you have configured TBSM to import data from Tivoli Application Dependency Discovery Manager, you can request a bulk or delta import before the configured polling interval has passed by running the **cmdbdiscovery** command.
   - To request a bulk import, specify the -b and -r flags.
   - To request a delta import, specify the -r flag.

   The import that you request begins within 60 seconds. The following message is written to the message log: GTMCL5292I Beginning CMDB import. Bulk: *x* Hostname: *taddmHostName*. The *x* value is **true** if you requested a bulk import or **false** if you requested a delta import.
4. UNIX
   Stop the Discovery Library Toolkit daemon by running the **tbsmrdr_stop.sh** script.
5. Windows
   Stop the Discovery Library Toolkit service from the **Services** window or by issuing the **net stop ASICRToolkitSvc** command.

Discovery Library Toolkit commands

The commands for the Discovery Library toolkit are located in the $TBSM_HOME/XMLtoolkit/bin directory. For help on any of these commands, run the command with the -? option.

Command output on Windows systems

On Windows systems running Western European language environments such as French, command prompt windows running an active code page of 850 might contain messages that contain corrupted characters. To fix this condition, run the **chcp 1252** command to change the active code page for the command prompt window to 1252.

cmdbdiscovery

This section describes the **cmdbdiscovery** command.

Purpose

The **cmdbdiscovery** command instructs the Discovery Library toolkit to run a bulk discovery of IBM Tivoli Application Dependency Discovery Manager. It schedules, cancels, or queries the status of an import request. The Discovery Library toolkit normally requests only changes that occurred in Tivoli Application Dependency Discovery Manager since the last query. When the **cmdbdiscovery** command is run with the **-b** flag, the next time that the toolkit begins a query, it runs a bulk discovery, requesting all resources in Tivoli Application Dependency Discovery Manager.

Syntax

```
cmdbdiscovery -b [-r] | -c | -s | -r
```

Parameters

-b
Start a bulk discovery.

-c
Cancel a bulk discovery.

-s
Query discovery status.

-r
Run the bulk or delta discovery immediately rather than waiting for the next polling interval. If this parameter is not specified with the **-b** parameter, the command performs a delta request.

The following example displays how to start a bulk discovery:

```
cmdbdiscovery -b
```

The following example displays how to cancel a bulk discovery:

```
cmdbdiscovery -c
```

The following example requests an immediate delta import from Tivoli Application Dependency Discovery Manager:

```
cmdbdiscovery -r
```

migrateguids

This section describes the **migrateguids** command.

Purpose

Run the **migrateguids** script whenever a migration has been performed on the Tivoli Application Dependency Discovery Manager server and GUID change notification files have been transmitted to the TBSM server. Processing these files migrates the TBSM database, updating the Tivoli Application Dependency Discovery Manager GUIDs with their new values.

You need to run this script only when message GTMCL5354E is written to the `msgGTM_XT.log` file.

You must stop the Discovery Library Toolkit before you issue this command. Depending on the number of resources that are affected resources, the time required to migrate the GUIDs could be significant.

Syntax

```
migrateguids -m [-?]
```

Parameter

-m
Migrates the GUIDs in the database.

setxmlaccess

This section describes the **setxmlaccess** command.

Purpose

The **setxmlaccess** command encrypts the user IDs and passwords that are used by the toolkit. The **-U** and **-P** options can be used to set all the user IDs and passwords at one time. If the user IDs and passwords have been set and need to be changed, there are options that you can use to change a particular set of values.

Syntax

```
setxmlaccess [ [ -U userIDs -P passwords ] |
    [ -tbsmid userID -tbsmpw password ] |
    [ -taddmid userId -taddmpw password ] |
    [ -tbsmdbid dbUserId -tbsmdbpw dbPassword ] |
    [ -taddmdbid dbUserId -taddmdbpw dbPassword ] ]
    [-k encryptionKey]
```

Parameters

-U
Sets the database, TBSM, and IBM Tivoli Application Dependency Discovery Manager user IDs. The format is:

```
tbsmDbUserID:tbsmUserID:taddmUserID:taddmDbUserID
```

At a minimum, the TBSM database and user IDs must be provided. If you are using Tivoli Application Dependency Discovery Manager as the data source, at four user IDs must be provided. Separate the user IDs with colons.

-P
Set the passwords associated with the **-U** flag. The format is:

```
tbsmDbPw:tbsmPw:taddmPw:taddmDbUserPw
```

The number of passwords depends on the input to the **-U** flag; two or four passwords must be provided. Separate the passwords with colons.

-k
Encryption key. This option specifies the encryption key that is used to encrypt the user IDs and passwords. If the key is not provided, an internal default key is used. The key does not need to be specified on each invocation. If a key is specified, it is saved for future use.

The following options are mutually exclusive with the **-U** and **-P** options. Use these options to set an individual user ID / password pair. Only one pair can be set per invocation of the script.

-tbsmid
TBSM user ID.

-tbsmpw
TBSM password.

-taddmid
Tivoli Application Dependency Discovery Manager user ID.

-taddmpw
Tivoli Application Dependency Discovery Manager password.

-tbsmdbid
TBSM database user ID.

-tbsmdbpw
TBSM database password.

-taddmdbid
Tivoli Application Dependency Discovery Manager database user ID.

-taddmdbpw
Tivoli Application Dependency Discovery Manager database password.

In the following example, the database user ID is set to `dbuser`, with a password of `dbpassword`, and the TBSM user ID is `admin`, with a password of `netcool`. Tivoli Application Dependency Discovery Manager is not used as a data source.

```
setxmlaccess.sh -U dbuser:admin -P dbpassword:netcool
```

In the following example, all of the users IDs and passwords are set.

```
setxmlaccess.sh -U dbuser:admin:administrator:db2inst1 -P
dbpassword:netcool:collation:db2inst1pw
```

In the following example, the Tivoli Application Dependency Discovery Manager user ID and password are set.

```
setxmlaccess.sh -taddmid administrator -taddmpw collation
```

taddmdirect

This section describes the **taddmdirect** command.

Purpose

This utility configures connection information for the Tivoli Application Dependency Discovery Manager database.

Before issuing this command, verify that the Tivoli Application Dependency Discovery Manager database user ID and password have been configured. If they have not, configure them using the **setxmlaccess** command.

Syntax

```
taddmdirect [-disable] | [-enable -t dbtype [-h hostname -d database -p port]
[-s schema] ] | [-display] [-test][-gentaddmsql]
```

Parameters

One of the following parameters is required:

-disable
Disables the JDBC connection.

-enable
Enables the JDBC connection. If the **-enable** parameter is specified, you must also specify the **-t** parameter.

-display
Displays the current settings.

-t
The type of database. Valid values are DB2 and ORACLE. This parameter is required if the **-enable** parameter is specified.

The following parameters are optional if the information has been previously configured:

-h
The Tivoli Application Dependency Discovery Manager database host name. If the default values for the **-d** and **-p** parameters are not valid, you must also specify these values.

-d
Database name. The default value is cmdb.

-p
Port that the database is listening on. The default value is 50000.

-s
Database schema or the owning user ID.

-test
Tests the JDBC connection. If the connection fails, hold **Ctrl** while pressing **C** to terminate the retry sequence.

-gentaddmsql
generates the SQL to query Tivoli Application Dependency Discovery Manager.

In this example, define the connection to a Tivoli Application Dependency Discovery Manager running DB2. This example also shows the setting of the database user ID and password.

```
taddmdirect.sh -enable -t DB2 -h ktaddm2.ibm.com -d CMDB -p 50000 -s db2inst1
```

utils

This section describes the **utils** command.

Purpose

The **utils** command provides TBSM toolkit utilities. You cannot use the **utils** script to run the migrateGuids.xml file. You must use the **migrateguids** script to run this XML file. See “migrateguids” on page 185 for more information about using the **migrateguids** script.

Syntax

```
utils -e utility [-U user -P pw -d dbname -h hostname -m schema -p port -z]
```

Parameters

-e
Executes the specified utility. Utilities must reside in the $TBSM_HOME/XMLtoolkit/scripts directory and must follow the toolkit scripting schema.

-U
Database user. Required for database access

-P
Database password. Required for database access

-d
Database name

-h
Database hostname

-m
Database schema

-p
Database port

-z
Execute the script without using RMI

If the script specified by the -e flag contains a SQL task, you must supply the database username and password. If the -U and -P flags are not specified on the command line, the script prompts you for them.

If the `-d`, `-h`, `-m`, and `-p` flags are not specified, the database configuration specified in the Discovery Library Toolkit properties is used.

Scripts that reload definitions or reevaluate data in the database are integrated with the Discovery Library Toolkit. The request is sent to the Discovery Library Toolkit and the system is quiesced before the action is taken. After the action is completed the system resume processing. You do not have to stop and start the Discovery Library Toolkit for new definitions to take effect.

If the Discovery Library Toolkit is down and you want to run one of the actions that reload definitions or reevaluate data, specify the `-z` flag so that the `utils` script does not attempt to route the request to the Discovery Library Toolkit.

Utility values

`reload_cdm_definitions.xml`
Loads changes that were made to the CDM customization into the database. The changes are then applied to the data already in the database. A bulk import is no longer required to apply CDM updates.

`reload_label_definitions.xml`
Load changes that were made to the `LabelingRules.xml` file into the database. The changes are then applied to the data in the database. A bulk import is no longer required to apply a label configuration update.

`validate_dependency_counters.xml`
This utility validates dependency counters. Incorrect dependency counters can result in resources not appearing as children of a service.

`collect_db_info.xml`
Collects database statistics helpful for debugging problems. The information is written to the `../log/db-info.out` file.

initiate_failover.xml
Initiates process to failover to the acting secondary Discovery Library Toolkit instance in a failover environment.

reevaluate_cdm_naming.xml
Reapply the naming rules to the data in the database.

toolkit_status.xml
Displays the status of the various Discovery Library Toolkit processes.

setdbschema

This section describes the **setdbschema** command.

Purpose

The **set_db_schema** command drops, truncates, and creates the toolkit database schema. Care must be taken when running this script, data loss can occur. Before running this command, always back up your database.

Syntax

```
setdbschema -U dbUser -P dbPassword -f a|s|t|i|d|v|w
```

If no options are specified, stage tables are dropped/created (`-f t`).

Parameters

-U
The database user ID.

-P
The database password

-f
function identifier, specify as follows:

a
All permanent and stage tables are dropped/created. All data is deleted

s
Permanent tables dropped/created. All data is deleted

t
Stage tables dropped/created. This is the default setting.

i
IdML stage tables dropped/created.

d
Permanent tables are truncated. All data is deleted.

v
Drop and rebuild all the views.

w
Drop all the views

m
Deletes duplicate ESDA instances from the database.

taddmconfig

This section describes the **taddmconfig** command.

Purpose

This command configures the TADDM data source. This script consolidates the invocation of the various scripts used to configure the TADDM data source. This includes calling: **taddmrmiconfig**, **taddmdirect**, and **setxmlaccess**.

This script should be used if TADDM is not currently a data source but is to be added as one. If TADDM is currently a data source for the toolkit and only a specific aspect of the configuration needs to be changed, then the script associated with that particular aspect should be called.

- The **taddmrmiconfig** script sets the properties associated with the RMI connection to the TADDM server.
- The **taddmdirect** script configures the JDBC connection to the TADDM database.
- The **setxmlaccess** script encrypts the user and password for the TADDM server and the TADDM database.

For an explanation of each prompt displayed by **taddmconfig**, run each of the above commands with the `-?` option.

Syntax

```
taddmconfig [-e] | [-d]
```

Parameters

One of the following parameters is required:

-e
Enables TADDM as a data source.

-d
Disables TADDM as a data source.

In this example, define the TADDM data source is enabled.

```
taddmconfig.sh -e
```

oslcconfig

This topic describes the options for the **oslcconfig** command.

Purpose

To configure the Data server servlet, use the Discovery Library Toolkit `oslcconfig` command located in the `$TBSM_HOME/XMLtoolkit/bin` directory.

Syntax

The `oslcconfig` command has the following syntax:

```
oslcconfig [-alter [properties]] | [-disable] | [-display] |
[-enable [properties]] | [-test]
```

UNIX/Linux: On UNIX and Linux systems, the command is: `oslcconfig.sh`

Parameters

The `oslcconfig` command has the following parameters:

alter
Updates one or more of the OSLC configuration properties. Additional properties can be assigned to this parameter.

disable
Disables OSLC enrichment support.

display
Displays the current property values.

enable
Enables OSLC enrichment support.

test
Allows you to test the connection to the Registry Service server.

Additional properties

A number of additional values can be set by applying properties to the `alter` and `enable` parameters. These properties can be applied to either the `alter` and `enable`. However, you must set values for the `-h`, `-p`, `-U`, and `-P` properties, since these properties do not have default values. If you do not set these values and attempt to run the `oslcconfig` command with the `enable` parameter, the command fails. It is only necessary to set these properties once. The remaining properties have default values and are optional.

-i
(Optional) This Discovery Library Toolkit property sets the default interval in seconds for which the Registry Service is queried. The minimum value is 5. If you enter a value lower than 5 it is set to 5. The default value is 120.

-s
(Optional) This Discovery Library Toolkit property sets the protocol used to communicate with the Data server servlet. Specify `true` to use HTTPS. The default value is `false`.

-h
This Registry Service property sets the IP address for the Registry Service.

-p
This Registry Service property sets the HTTP port on which the Registry Service listens.

-b
(Optional) This Registry Service property sets the batch size for records retrieved from the Registry Service. The default value is 1000 and the minimum value is 100. If you enter a value lower than 100 it is set to 100. Default: 1000

-r
(Optional) This Registry Service property sets the protocol used to communicate with the Registry Service. Specify `true` to use HTTPS. The default value is `false`.

-U
This Registry Service property sets the username used to connect to the Registry Service.

-P
This Registry Service property sets the password used to connect to the Registry Service. If you enter a username but do not provide a password, you are prompted for a password.

Example

This example represents the minimum configuration. The JazzSM Registry hostname, port, user, and password are set:

```
oslcconfig -alter -h myOslcHost -p 16310 -U tipuser -P tippassword
```

taddmconfig

This section describes the **taddmconfig** command.

Purpose

This command configures the TADDM data source. This script consolidates the invocation of the various scripts used to configure the TADDM data source. This includes calling: **taddmrmiconfig**, **taddmdirect**, and **setxmlaccess**.

This script should be used if TADDM is not currently a data source but is to be added as one. If TADDM is currently a data source for the toolkit and only a specific aspect of the configuration needs to be changed, then the script associated with that particular aspect should be called.

- The **taddmrmiconfig** script sets the properties associated with the RMI connection to the TADDM server.
- The **taddmdirect** script configures the JDBC connection to the TADDM database.
- The **setxmlaccess** script encrypts the user and password for the TADDM server and the TADDM database.

For an explanation of each prompt displayed by **taddmconfig**, run each of the above commands with the `-?` option.

Syntax

```
taddmconfig [-e] | [-d]
```

Parameters

One of the following parameters is required:

-e
Enables TADDM as a data source.

-d
Disables TADDM as a data source.

In this example, define the TADDM data source is enabled.

```
taddmconfig.sh -e
```

Updating IBM Tivoli Business Service Manager after installing a fix pack for IBM Tivoli Application Dependency Discovery Manager

This topic describes how to update TBSM after you install a Tivoli Application Dependency Discovery Manager fix pack or upgrade.

About this task

If a fix pack or upgrade for the Tivoli Application Dependency Discovery Manager is made, the Tivoli Application Dependency Discovery Manager jar files used by TBSM must be updated. If these jar files are not updated, random exceptions can occur when data is transferred from Tivoli Application Dependency Discovery Manager to TBSM.

If you are running Tivoli Application Dependency Discovery Manager 7.2 or later, TBSM verifies that its copy of each jar matches the copy on the Tivoli Application Dependency Discovery Manager server. This verification is completed before each import. If a discrepancy is detected, the following message is written to the Discovery Library Toolkit log:

```
GTMCL5355W: The TADDM Java API jar file taddm-api-client.jar, used by TBSM to
 communicate with  the TADDM server, has changed on the TADDM server.
 To prevent possible interruptions in service it is recommended that
 you update the file on the TBSM server.
```

If TBSM is connected to Tivoli Application Dependency Discovery Manager 7.2.1 or later, the latest jar file is downloaded automatically from Tivoli Application Dependency Discovery Manager and are made available the next time that the Discovery Library Toolkit is started.

If TBSM is connected to Tivoli Application Dependency Discovery Manager 7.2 or earlier, the jar file on the TBSM server must be updated at the earliest convenient time. In all cases, the import continues, even if the jars are detected to be out of sync.

Procedure

- Depending on the version of Tivoli Application Dependency Discovery Manager used, the location and number of client jar files differs. Please read the following for the Tivoli Application Dependency Discovery Manager version used.
  - In Tivoli Application Dependency Discovery Manager 7.1.2, the client jar files are contained in 1 jar file. This jar file is located on the Tivoli Application Dependency Discovery Manager server in `../dist/sdk/clientlib and is named taddm-api-client.jar`. Copy this file to the `$TBSM_HOME/XMLtoolkit/sdk/clientlib` directory.
  - In Tivoli Application Dependency Discovery Manager 7.2.x.x, the single client jar used in Tivoli Application Dependency Discovery Manager 7.1.2 has been separated into two files. The jar files are located on the Tivoli Application Dependency Discovery Manager server in `../dist/sdk/lib and are named taddm-api-client.jar` and `platform-model.jar`. Copy this file to the `$TBSM_HOME/XMLtoolkit/sdk/clientlib` directory.

    It is optional but strongly recommended that you also copy the `oal-topomgr.jar` jar file from the Tivoli Application Dependency Discovery Manager server to the TBSM server. The presence of this jar file allows TBSM to use JDBC to import a majority of the required data. This method of import has been shown to be significantly faster than the RMI API's. This jar file should also be placed in `$TBSM_HOME/XMLtoolkit/sdk/clientlib`.

    If Tivoli Application Dependency Discovery Manager 7.2.1 or later is being used, the three jar files mentioned above are automatically copied from the Tivoli Application Dependency Discovery Manager server to the TBSM server the first time that the TBSM Discovery Library Toolkit is started.

After the initial load of these files, the Discovery Library Toolkit shuts down. Restarting the Discovery Library Toolkit completes the configuration.

Running the Tivoli Event Integration Facility probe on UNIX systems

Starting and stopping the EIF probe on UNIX systems requires one of the following two commands:

Procedure

1. To start the EIF probe, issue the command **$OMNIHOME/probes/nco_p_tivoli_eif**.
2. To stop the EIF probe, do one of the following:
   - If you started the EIF probe as a background process, locate the process and issue the **kill** command to stop it.
   - If you started the EIF probe in the console or terminal window, hold **Ctrl** and press **C** in the console or terminal window.

Running the Tivoli Event Integration Facility probe as a Windows service

If you installed the Tivoli Event Integration Facility probe (EIF probe) to run as a Windows service, you can use two different methods to start and stop the probe.

Procedure

1. To start the EIF probe, perform one of the following steps:
   - Open a Windows command prompt and issue the command **sc start NCONONNATIVEProbe $NCOTivoliEifProbe**.
   - Select **Start** > **All Programs** > **Administrative Tools** > **Service**, and select the service that you want to start.
2. To stop the EIF probe, perform one of the following steps:
   - Open a Windows command prompt and issue the command **sc stop NCONONNATIVEProbe $NCOTivoliEifProbe**.
   - Select **Start** > **All Programs** > **Administrative Tools** > **Service**, and select the service that you want to stop.

Running the Tivoli Event Integration Facility probe on Windows systems using the command prompt

If you did not install the Tivoli Event Integration Facility probe (EIF probe) as a Windows service, you must run the probe using the command prompt.

Procedure

1. To start the EIF probe, open a Windows command prompt and issue the command **nco_p_tivoli_eif.bat**.
2. To stop the EIF probe, in the Windows command prompt in which you started the probe, hold **Ctrl** and press **C** or hold **Ctrl** and press **Break**.

Exporting and importing between TBSM servers

You can export all data and configuration from an existing TBSM 6.2.0 server to another TBSM 6.2.0 server by exporting the data and subsequently importing that data. This section covers both the TBSM Data Server and TBSM DASH Server.

Exporting Procedure

On TBSM Data Server:

Run the following command:

```
nci_export TBSM /tmp/NCI_export
output: /tmp/NCI_export/
```

For more details, see nci_export

Run the following command:

```
rad_radshell
output: /opt/IBM/tivoli/tbsm/export/export.radsh
```

For more details, see “export” on page 78.

Run the following command:

```
tbsm_export.sh -U db2inst1 -P db2inst1 -directory \temp\tbsmExportImportDir -all
output: /opt/IBM/tivoli/tbsm/XMLtoolkit/bin/temptbsmExportImportDir/
```

For more details, see “Exporting and importing customization artifacts” on page 196.

On TBSM DASH Server:

Run the following command:

```
tipcli.sh Export --username tbsmadmin --password tbsmadmin
output: /opt/IBM/JazzSM/ui/output/data.zip
```

For more details, see Exporting and importing

Importing Procedure

Copy the above output directories/files from the exporting TBSM 6.2.0 server to the importing TBSM 6.2.0 server, for example, place them under `/tmp directory`.

On TBSM Data Server:

Run the following command:

```
nci_import TBSM /tmp/NCI_export
```

For more details, see nci_import.

Run the following command:

```
rad_radshell -f /tmp/export.radsh
```

For more details, see “Sending file contents to the RAD shell tool” on page 55.

Run the following command:

```
tbsm_import.sh -U db2inst1 -P db2inst1 -directory /tmp/temptbsmExportImportDir
```

For more details, see “Exporting and importing customization artifacts” on page 196.

On TBSM DASH Server:

Copy `/tmp/data.zip /opt/IBM/JazzSM/ui/input/`

Run the following command:

```
tipcli.sh Import --username tbsmadmin --password tbsmadmin
```

For more details, see Exporting and importing

Exporting and importing customization artifacts

TBSM centralizes the management of many of its customization files, also called customization artifacts, into a central artifact datastore. The artifact datastore is maintained in the TBSM database. These customization artifacts can be accessed and configured using programmatic interfaces to the Data server, Dashboard server, and the Discovery Library Toolkit.

This section outlines the purpose and syntax for the commands used to export and import customization artifacts. For more information about using these commands, see *Exporting and Importing* at the following location:https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/Tivoli%20Business%20Service%20Manager1

Using your local file system and a set of command line utilities you can export and import artifacts out of and into the datastore. Using the artifact datastore for managing customization artifacts, you can meet your failover, maintenance, and reliability goals. The customization artifact datastore allows you to:

- Promote customization artifacts between Test, Quality Assurance or Certification, and Production environments using the command line tools that access the artifact datastore. These customization artifacts can be transferred from one system in a form that is easily consumable by artifact command line tools on another system.
- Synchronize the primary Data server and the Discovery Library Toolkit systems with failover Data server and Discovery Library Toolkit systems. This can be done when the systems share a single datastore.
- Share customization artifacts between a primary and failover Data server through the single artifact datastore. This enables you to take advantage of customization features such as editing Custom Canvases when the failover Data server is active.
- Transfer customization artifacts, such as Custom Canvases, between systems when they are exported from the datastore.
- Simplify the management of the default configurations and post-installation customization through the organization of the customization artifacts into categories along with a designation of whether the artifact is of a TBSM-provided origin or a customized user-defined origin.
- Back up customization artifacts that are maintained in the datastore in a manner that is consistent with your database backup plan.

Customization artifact command line utilities

TBSM provides a set of command line utilities that allow you to manually interact with the customization artifact datastore:

The command utilities **only** support exchanging data between servers that are running the same version of TBSM.

Note: The syntax for each of the commands is given in the topics in this section. To access the command help information, type the command followed by a `-?` and press `Return`.

These commands are located in the `$TBSM_HOME/XMLtoolkit/bin` directory.

UNIX On UNIX systems, these commands have a .sh extension. For example, **`getArtifact.sh`.**

Windows On Windows systems, these commands have a `.bat` extension. For example, **`getArtifact.bat`.**

tbsm_export and tbsm_import
These commands allow you to move artifacts between TBSM systems. Artifacts are exported from the source system into a directory structure which captures all attributes of the artifact. This directory structure can then be made available to a target system for use by the `tbsm_import` command.

getArtifact, putArtifact, removeArtifact and listArtifact
These commands work interactively with the artifact datastore when you develop customization enhancements. The commands are tailored to individual artifact operations against the runtime configuration of the artifact as defined by the category and origin attributes of the artifact.

dbfileutility
The `dbfileutility` command is the basis for each of the command line utilities. Using the options available with `dbfileutility`, you can access all facets of the artifact datastore. However, it is not generally necessary to use this utility as the required functions are available using the other command line utilities.

tbsm_export

This section describes the `tbsm_export` command.

Purpose

The `tbsm_export` command allows you to export customization artifacts from the TBSM database and write their contents to a specified directory. The `tbsm_export` and `tbsm_import` commands allow you to move customization artifacts from one TBSM system to another.

The `tbsm_export` command can be used to periodically back up customized artifacts.

Syntax

The syntax for this command is:

```
tbsm_export.sh|bat [-U dbUser [-P dbPassword] ] -d directory [ [-n artifactname]
[-c category]                                                                    [-s
subcategory] ]
                                      | -d directory -all
                                      | -d directory -commandfile filename
```

Note: The `tbsm_export` command parameters are positional. You must adhere to the command syntax and the order that they are provided. The command must be followed by, in order, the database access credentials (optional) and then the command options. The database access credentials and target directory are not required for `tbsm_export` commands in the command file.

Parameters

-U
The database user ID. If you do not specify a value for this parameter, you are prompted for the database user ID.

-P
The database used ID password. If you do not specify a value for this parameter, you are prompted for the database user ID password.

-d directory
The target file system directory for customization artifacts in an export or import directory hierarchy. The specified directory can be a fully qualified or relative directory path. This parameter can be replaced with `-directory directory`.

-n artifactname
The name of a customization artifact. This can be replaced with `-name artifactname`. The name you specify can include a percent-sign, % wildcard. This can be used to represent a number of characters, or none.

-c category
The category, or type, of a customization artifact. This can be replaced with `-category category`. The valid categories are:

Table 137. Valid categories for a customization artifact

Category	Description
chart	Charts
composites	Discovery Library Toolkit Common Data Model (CDM) attributes
customcanvas	Custom canvases
eventidentifiers	Discovery Library Toolkit event identifier rules
labeling	Discovery Library Toolkit labeling rules
maintenance	Maintenance schedules
menuactions	User defined actions and sub-menus
notifications	Discovery Library Toolkit notification rule
scrbase	Discovery Library Toolkit Service Component Repository (SCR) base configuration
sla	Service-level agreement (SLA) definitions
taddmfilters	Discovery Library Toolkit TADDM to SCR filter rules
tbsmattributes	Discovery Library Toolkit SCR to TBSM attribute filter rules
treetemplate	Tree templates
viewdefinition	View definitions
scrconfig	Discovery Library Toolkit base configuration XML files

The `tbsm_export` command additionally supports the following service model categories:

Table 138. Service model categories

Category	Description
datafetchers	Data fetchers
datasources	Data sources
templates	Service templates and their associated rules

-s subcategory
The subcategory, or sub-type, of a customization artifact. This can be replaced with `-subcategory subcategory`.

-all
This parameter exports all customization artifacts.

-commandfile filename
This parameter allows you to specify a command file containing one or more `tbsm_export` commands with the following syntax:

```
tbsm_export [ [-n artifactname] [-c category] [-s subcategory] ]
    |    -all
```

The commands within the command file must be specified with each on an individual line. Any line with a leading number sign # is interpreted as a inexecutable comment. The specified command file can be at a fully qualified or relative path file location.

Note: The `tbsm_export` command exports each customization artifact into an export directory hierarchy structure that defines their category, subcategory and origin. This structure is then used by a subsequent `tbsm_import` command. TBSM runtime customization artifacts can be copied from the TBSM database directly to a specified directory, without an export directory hierarchy, using the `getArtifact` command.

Note: TBSM base customization artifacts are not exported by the `tbsm_export` command.

Example

The following example exports all of the customization artifacts in the database to the specified directory:

Note: The **`.bat`** extension is not required when you enter the command on Windows systems. However, the **`.sh`** extension is required when you enter the command on UNIX systems.

```
tbsm_export -directory \temp\tbsmExportImportDir -all
```

Example

The following example exports all of the customization artifacts that are in the database and that begin with the character sequence `AdminView`.

```
tbsm_export -directory \temp\tbsmExportImportDir -name AdminView%
```

Example

The following example exports all of the customized Discovery Library Toolkit event identifier rules in the TBSM database to a specified location:

```
tbsm_export -directory \temp\tbsmExportImportDir -category eventidentifiers
```

Example

The following example uses a command file to export a collection of customization artifacts in the database to the specified directory:

```
tbsm_export -directory \temp\tbsmExportImportDir -commandfile
\tbsmuser\exportCommandFile.txt
```

An example of a command file is as follows:

```
# TBSM export commands
# Export Admin data server artifacts
tbsm_export -name AdminScorecard% -category treetemplate
tbsm_export -name AdminView% -category viewdefinition
# Export User data server artifacts
tbsm_export -name UserScorecard% -category treetemplate
tbsm_export -name UserView% -category viewdefinition
# Export all service templates
tbsm_export -category templates
```

tbsm_import

This section describes the `tbsm_import` command.

Purpose

The `tbsm_import` command allows you to import customization artifacts to the TBSM database and therefore make them available to the runtime configuration. The `tbsm_import` command imports customization artifacts that have been exported using the `tbsm_export` command. The `tbsm_export` and `tbsm_import` commands are used to move customization artifacts from one TBSM system to another.

Syntax

The syntax for this command is:

```
tbsm_import.sh|bat [-U dbUser [-P dbPassword] ] -directory directory
```

Note: The `tbsm_import` command parameters are positional. You must adhere to the command syntax and the order that they are provided. The command must be followed by, in order, the database access credentials (optional) and then the command options.

Parameters

-U
The database user ID. If you do not specify a value for this parameter, you are prompted for the database user ID.

-P
The database used ID password. If you do not specify a value for this parameter, you are prompted for the database user ID password.

-d directory
The target file system directory for customization artifacts in an export or import directory hierarchy. The specified directory can be a fully qualified or relative directory path. This parameter can be replaced with `-directory directory`.

Note: You must update the runtime configuration of the Data server or the Discovery Library Toolkit by performing the recommended activation procedure for the categories of the artifacts that are imported into the database. For more information, see the *Customization Artifact Runtime Activation* and *Customization artifact categories* sections of this document.

Note: `tbsm_import` command must be executed using an appropriate directory hierarchy created by the `tbsm_export` command. New or updated customization artifacts that are added to the TBSM database directly from a specified directory, without an export/import directory hierarchy, must be added using the `putArtifact` command.

Note: If the customization artifacts in the specified directory are replacing previous versions of the same files, the previous version is saved in the database as a backup. When a new backup version is saved, the previous backup, if one existed, is deleted from the database.

Example

The following example imports all of the customization artifacts in the database from the specified directory:

Note: The **`.bat`** extension is not required when you enter the command on Windows systems. However, the **`.sh`** extension is required when you enter the command on UNIX systems.

```
tbsm_import -directory \temp\tbsmExportImportDir
```

getArtifact

This section describes the `getArtifact` command.

Purpose

The `getArtifact` command allows you to retrieve a customization artifact, available to the TBSM runtime configuration, from the TBSM database and write the contents to a specified directory.

Syntax

The syntax for this command is:

```
getArtifact.sh|bat [-U dbUser [-P dbPassword] ] -d directory -n artifactname
 -c category [-s subcategory]
```

Note: The `getArtifact` command parameters are positional. The command must be followed by, in order, the database access credentials (optional) and then the command options.

Parameters

-U
The database user ID. If you do not specify a value for this parameter, you are prompted for the database user ID.

-P
The database used ID password. If you do not specify a value for this parameter, you are prompted for the database user ID password.

-d directory
The target file system directory for the customization artifact. The specified directory can be a fully qualified or relative directory path.

-n artifactname
The name of a customization artifact. This can be replaced with `-name artifactname`.

-c category
The category, or type, of a customization artifact. This can be replaced with `-category category`. The valid categories are:

Table 139. Valid categories for a customization artifact

Category	Description
`chart`	Charts
`composites`	Discovery Library Toolkit Common Data Model (CDM) attributes
`customcanvas`	Custom canvases
`eventidentifiers`	Discovery Library Toolkit event identifier rules
`labeling`	Discovery Library Toolkit labeling rules
`maintenance`	Maintenance schedules
`menuactions`	User defined actions and sub-menus
`notifications`	Discovery Library Toolkit notification rule
`scrbase`	Discovery Library Toolkit Service Component Repository (SCR) base configuration
`sla`	Service-level agreement (SLA) definitions
`taddmfilters`	Discovery Library Toolkit TADDM to SCR filter rules
`tbsmattributes`	Discovery Library Toolkit SCR to TBSM attribute filter rules
`treetemplate`	Tree templates
`viewdefinition`	View definitions
`scrconfig`	Discovery Library Toolkit base configuration XML files

-s subcategory
The subcategory, or sub-type, of a customization artifact. This can be replaced with `-subcategory subcategory`.

Example

The following example retrieves a view definition from the database and writes the contents to `\temp \My_ViewDefinition_Relationships.xml`:

Note: The **.bat** extension is not required when you enter the command on Windows systems. However, the **.sh** extension is required when you enter the command on UNIX systems.

```
getArtifact -directory \temp -name My_ViewDefinition_Relationships.xml
-category viewdefinition
```

putArtifact

This section describes the `putArtifact` command.

Purpose

The `putArtifact` command allows you to copy a customization artifact into the database from the specified location.

Syntax

The syntax for this command is:

```
putArtifact.sh|bat [-U dbUser [-P dbPassword] ] -n artifactname -c category
[-s subcategory]
```

Note: The `putArtifact` command parameters are positional. You must adhere to the command syntax and the order in which they are provided.

Parameters

-U
The database user ID. If you do not specify a value for this parameter, you are prompted for the database user ID.

-P
The database used ID password. If you do not specify a value for this parameter, you are prompted for the database user ID password.

-n artifactname
The name of a customization artifact on the file system. The specified name can be a fully qualified or relative directory path. This can be replaced with `-name artifactname`.

-c category
The category, or type, of a customization artifact. This can be replaced with `-category category`. The valid categories are:

Table 140. Valid categories for a customization artifact

Category	Description
`chart`	Charts
`composites`	Discovery Library Toolkit Common Data Model (CDM) attributes
`customcanvas`	Custom canvases
`eventidentifiers`	Discovery Library Toolkit event identifier rules

Table 140. Valid categories for a customization artifact (continued)	
Category	**Description**
`labeling`	Discovery Library Toolkit labeling rules
`maintenance`	Maintenance schedules
`menuactions`	User defined actions and sub-menus
`notifications`	Discovery Library Toolkit notification rule
`scrbase`	Discovery Library Toolkit Service Component Repository (SCR) base configuration
`sla`	Service-level agreement (SLA) definitions
`taddmfilters`	Discovery Library Toolkit TADDM to SCR filter rules
`tbsmattributes`	Discovery Library Toolkit SCR to TBSM attribute filter rules
`treetemplate`	Tree templates
`viewdefinition`	View definitions
`scrconfig`	Discovery Library Toolkit base configuration XML files

-s subcategory
The subcategory of a customization artifact. This can be replaced with `-subcategory subcategory`.

Note: If the specified customization artifact had a previous version in the database, the previous version is saved in the database as a backup. When a new backup version is saved, the previous backup, if one existed, is deleted from the database.

Note: You must update the runtime configuration of the Data server or the Discovery Library Toolkit by performing the recommended activation procedure for the categories of the artifacts that are imported into the database. For more information, see the *Customization Artifact Runtime Activation* and *Customization artifact categories* sections of this document.

Example

The following example copies a customized Discovery Library Toolkit event identifier rules file named `My_EventIdentifiers.xml` into the database:

Note: The **`.bat`** extension is not required when you enter the command on Windows systems. However, the **`.sh`** extension is required when you enter the command on UNIX systems.

```
putArtifact -name \temp\My_EventIdentifiers.xml -category
eventidentifiers
```

removeArtifact

This section describes the `removeArtifact` command.

Purpose

The `removeArtifact` command allows you to remove a customization artifact from the TBSM database.

Syntax

The syntax for this command is:

```
removeArtifact.sh|bat [-U dbUser [-P dbPassword] ] -n artifactname -c category
[-s subcategory]
```

Note: The `removeArtifact` command parameters are positional. You must adhere to the command syntax and the order that they are provided.

Parameters

-U
The database user ID. If you do not specify a value for this parameter, you are prompted for the database user ID.

-P
The database used ID password. If you do not specify a value for this parameter, you are prompted for the database user ID password.

-n artifactname
The name of a customization artifact. This can be replaced with `-name artifactname`.

-c category
The category, or type, of a customization artifact. This can be replaced with `-category category`. The valid categories are:

Table 141. Valid categories for a customization artifact

Category	Description
`chart`	Charts
`composites`	Discovery Library Toolkit Common Data Model (CDM) attributes
`customcanvas`	Custom canvases
`eventidentifiers`	Discovery Library Toolkit event identifier rules
`labeling`	Discovery Library Toolkit labeling rules
`maintenance`	Maintenance schedules
`menuactions`	User defined actions and sub-menus
`notifications`	Discovery Library Toolkit notification rule
`scrbase`	Discovery Library Toolkit Service Component Repository (SCR) base configuration
`sla`	Service-level agreement (SLA) definitions
`taddmfilters`	Discovery Library Toolkit TADDM to SCR filter rules
`tbsmattributes`	Discovery Library Toolkit SCR to TBSM attribute filter rules
`treetemplate`	Tree templates
`viewdefinition`	View definitions
`scrconfig`	Discovery Library Toolkit base configuration XML files

-s subcategory
The subcategory of a customization artifact. This can be replaced with `-subcategory subcategory`.

Note: If the specified customization artifact had a previous version in the database, the previous version is saved in the database as a backup. When a new backup version is saved, the previous backup, if one existed, is deleted from the database.

Note: You must update the runtime configuration of the Data server or the Discovery Library Toolkit by performing the recommended activation procedure for the categories of the artifacts that are imported into the database. For more information, see the *Customization Artifact Runtime Activation* and *Customization artifact categories* sections of this document.

Example

The following example removes a customized Discovery Library Toolkit event identifier rules file named `My_EventIdentifiers.xml` from the database:

Note: The **`.bat`** extension is not required when you enter the command on Windows systems. However, the **`.sh`** extension is required when you enter the command on UNIX systems.

```
removeArtifact -name My_EventIdentifiers.xml -category
eventidentifiers
```

listArtifact

This section describes the `listArtifact` command.

Purpose

The `listArtifact` command allows you to list the customization artifacts available to the runtime configuration in the TBSM database.

Syntax

The syntax for this command is:

```
listArtifact.sh|bat [-U dbUser [-P dbPassword] ][-n  artifactname] [-c category]
[-s subcategory] ]                          |    -all
```

Note: The `listArtifact` command parameters are positional. You must adhere to the command syntax and the order that they are provided.

Parameters

-U
The database user ID. If you do not specify a value for this parameter, you are prompted for the database user ID.

-P
The database used ID password. If you do not specify a value for this parameter, you are prompted for the database user ID password.

-n artifactname
The name of a customization artifact. This can be replaced with `-name artifactname`. The name you specify can include a percent-sign, % wildcard. This can be used to represent a number of characters, or none.

-c category
The category, or type, of a customization artifact. This can be replaced with `-category category`. The valid categories are:

Table 142. Valid categories for a customization artifact

Category	Description
`chart`	Charts

Table 142. Valid categories for a customization artifact (continued)	
Category	**Description**
`composites`	Discovery Library Toolkit Common Data Model (CDM) attributes
`customcanvas`	Custom canvases
`eventidentifiers`	Discovery Library Toolkit event identifier rules
`labeling`	Discovery Library Toolkit labeling rules
`maintenance`	Maintenance schedules
`menuactions`	User defined actions and sub-menus
`notifications`	Discovery Library Toolkit notification rule
`scrbase`	Discovery Library Toolkit Service Component Repository (SCR) base configuration
`sla`	Service-level agreement (SLA) definitions
`taddmfilters`	Discovery Library Toolkit TADDM to SCR filter rules
`tbsmattributes`	Discovery Library Toolkit SCR to TBSM attribute filter rules
`treetemplate`	Tree templates
`viewdefinition`	View definitions
`scrconfig`	Discovery Library Toolkit base configuration XML files

-s subcategory
The subcategory, or sub-type, of a customization artifact. This can be replaced with `-subcategory subcategory`.

- all
Lists all customization artifacts.

Example

The following example lists all of the customization artifacts available, in the database, to the TBSM runtime configuration:

Note: The **.bat** extension is not required when you enter the command on Windows systems. However, the **.sh** extension is required when you enter the command on UNIX systems.

```
listArtifact -all
```

Example

The following example lists all of the customization artifacts that are in the database and that begin with the character sequence `AdminView`.

```
listArtifact -name AdminView%
```

Example

The following example lists all of the Discovery Library Toolkit event identifier rules available to the TBSM database runtime configuration.

```
listArtifact -category eventidentifiers
```

dbfileutility

This section describes the dbfileutility command.

Purpose

The dbfileutility command allows you to manipulate and administrate customization artifacts stored in the database.

The main functions of the dbfileutility command can be achieved using the getArtifact, putArtifact, removeArtifact, listArtifact, tbsm_export, and tbsm_import. The dbfileutility command is used when it is necessary to manipulate customization artifacts based on the file origin, customized or base TBSM configuration, or if there is a need to recover a backup version.

Syntax

The syntax for this command is:

```
dbfileutility.sh|bat    [-U dbUser [-P dbPassword] ] command-action command-options
```

Note: The dbfileutility command parameters are positional. You must adhere to the command syntax and the order that they are provided.

Parameters

-U
The database user ID. If you do not specify a value for this parameter, you are prompted for the database user ID.

-P
The database used ID password. If you do not specify a value for this parameter, you are prompted for the database user ID password.

command-action
The following command actions, and their supported options, are available on the dbfileutility command:

Table 143. Valid command actions

Category	Description
`get -d directory -f filename -c category [-s subcategory] -o origin [-b]`	Retrieves the specified customization artifact from the database and writes the contents to the specified file system directory. The `-d directory` option can be a fully qualified or relative path directory location.
`put -f filename -c category [-s subcategory] -o origin`	Copies the specified customization artifact into the database from the specified file system location, making it available to the TBSM runtime configuration. If the specified customization artifact had a previous version in the database, the previous version is saved in the database as a backup. When a new backup version is saved, the previous backup, if one existed, is deleted from the database. The `-f filename` option can be a fully qualified or relative path file location.

Table 143. Valid command actions (continued)	
Category	**Description**
`remove -f filename -c category [-s subcategory] -o origin [-b]`	Removes the specified customization artifact from the available runtime configuration in the database. The specified file or artifact is saved in the database as a backup. When a backup version is saved, the previous backup, if one existed, is deleted from the database.
`list [ [-f filename] [-c category] [-s subcategory] [-o origin] ] | -all`	Lists the customization artifacts in the database. The `-f filename` option can include a percent-sign % as a wildcard that can represent any number of characters, or none.
`export -d directory [ [-f filename] [-c category] [-s subcategory] [-o origin] ] | -d directory -all`	Exports customization artifacts in the database to the specified file system directory. The `export` command action exports each file and artifact into an export/import directory structure that defines their category, or type, subcategory and origin, that is used by a subsequent import command action. The `-d directory`' option can be a fully qualified or relative path directory location. The `-f filename` option can include a percent-sign % as a wildcard that can represent any number of characters, or none.
`import -d directory`	Imports customization artifacts from the specified directory into the database, making them available to the runtime configuration. The `import` command action must be executed with a proper export/import directory created by a previous export command action. If the specified customization artifact had a previous version in the database, the previous version is saved in the database as a backup. When a new backup version is saved, the previous backup, if one existed, is deleted from the database. The `-d directory`' option can be a fully qualified or relative path directory location.

The following command options are available on one or more command actions, for the `dbfileutility` command:

-f filename
The name of a customization artifact. This parameter can be replaced with `-filename filename`.

-c category
The category, or type, of a customization artifact. This can be replaced with `-category category`. The valid categories are:

Table 144. Valid categories for a customization artifact	
Category	**Description**
`chart`	Charts
`composites`	Discovery Library Toolkit Common Data Model (CDM) attributes
`customcanvas`	Custom canvases
`eventidentifiers`	Discovery Library Toolkit event identifier rules
`labeling`	Discovery Library Toolkit labeling rules
`maintenance`	Maintenance schedules
`menuactions`	User defined actions and sub-menus
`notifications`	Discovery Library Toolkit notification rule
`scrbase`	Discovery Library Toolkit Service Component Repository (SCR) base configuration
`sla`	Service-level agreement (SLA) definitions
`taddmfilters`	Discovery Library Toolkit TADDM to SCR filter rules
`tbsmattributes`	Discovery Library Toolkit SCR to TBSM attribute filter rules
`treetemplate`	Tree templates
`viewdefinition`	View definitions
`scrconfig`	Discovery Library Toolkit base configuration XML files

-s subcategory
The subcategory, or sub-type, of a customization artifact. This can be replaced with `-subcategory subcategory`.

-o origin
The origin of a customization artifact, where the valid origins are:

- `CUSTOM`: Customized customization artifact
- `TBSM`: Base TBSM customization artifact

-b
The backup version flag. When properly specified in a command, this parameter allows you to manipulate the backup version of the customization artifact. This parameter can be replaced with `-backup`

-d directory
The target file system directory for customization artifacts. The specified directory can be a fully qualified or relative directory path. This parameter can be replaced with `-directory directory`.

-all
When properly specified on a command, this parameter allows you to manipulate all customization artifacts.

Note: You must update the runtime configuration of the Data server or the Discovery Library Toolkit by performing the recommended activation procedure for the categories of the artifacts that are imported into the database. For more information, see the *Customization Artifact Runtime Activation* and *Customization artifact categories* sections of this document.

Example

The following example lists all of the customization artifacts in the database:

Note: The **.bat** extension is not required when you enter the command on Windows systems. However, the **.sh** extension is required when you enter the command on UNIX systems.

```
dbfileutility list -all
```

Example

The following example gets a customized view definition from the database:

```
dbfileutility get -directory \temp -filename My_ViewDefinition_Relationships.xml
-category viewdefinition -origin CUSTOM
```

Example

The following example puts a customized Discovery Library Toolkit event identifier rules file into the database.

```
dbfileutility put -filename \temp\My_EventIdentifiers.xml -category
eventidentifiers -origin CUSTOM
```

Example

The following example gets the backup version of a customized view definition from the database:

```
dbfileutility get -directory \temp -filename My_ViewDefinition_Relationships.xml
-category viewdefinition -origin CUSTOM -backup
```

Customization artifact categories

Customization artifacts are organized in the datastore within categories and, where required, subcategories. The category is the primary qualifying attribute of an artifact and conveys the purpose of the artifact. Other than some minor validation by the Discovery Library Toolkit categories, the artifact datastore does not enforce any restrictions on the names and content of artifacts. It is the responsibility of the using components, in other words the Data server, Dashboard server, Discovery Library Toolkit and manual processes using the command line utilities, to properly form and name artifacts.

The procedures for updating the TBSM runtime configuration after manually adding, replacing, or removing customization artifacts, using the command line utilities, varies with each category and is described in more detail in the *Customization artifact runtime activation* section of this document. The categories currently supported in the datastore each represent a different type of customization artifact.

Data server artifact categories

The following are the Data server artifact categories:

chart
This artifact category contains the chart `rptdesign` files that are used to define charts created from the TBSM chart editor and Business Intelligence and Reporting Tools (BIRT) Designer application.

When manually adding, replacing or removing charts in the datastore, the Data server and Dashboard server must be restarted in order to update the runtime configuration.

maintenance
This artifact category contains only the `scheduleTime.xml` file. This file is used to define the scheduled maintenance periods for services.

When you are manually replacing the `scheduleTime.xml` file in the datastore, you must restart the Data server to update the runtime configuration.

menuactions
This artifact category contains the menu actions XML files. These files are used to define the right-click menu and submenu actions that provide access to other views within TBSM or to support launch-

in context to external applications. This category uses a subcategory that defines the right-click option as a primary menu action or a submenu action, as `action` and `submenuaction`, respectively. Each menu and submenu action is contained within its own XML file, with the outer most XML element being the `openURLAction` and `dynamicSubMenuAction` tags, respectively. Each file must be uniquely named within the category and subcategory, and must be named the same as the XML `name` tag within the file.

When manually adding, replacing, or removing menu action XML files from the datastore, issue the `rad_reinitcanvas` command or restart the Dashboard server to update the runtime configuration.

sla
This artifact category contains only the `cumulTimeSLA.xml` file. This file is used to define service-level agreements for services.

The `cumulTimeSLA.xml` file must not be manually replaced or removed from the datastore.

treetemplate
This artifact category contains the tree template XML files used to define the tree templates which provide column names and column content information for the **Service Tree** portlet. Each tree template is contained within its own XML file, with the outer most XML element being the `treeTemplate` tag. Each file must be named uniquely within the category and be named the same as the XML name tag within the file.

When manually adding, replacing, or removing tree template XML files from the datastore, issue the `rad_reinitcanvas` command or restart the Dashboard server to update the runtime configuration.

viewdefinition
This artifact category contains the view definition XML files in the datastore used to define alternate views within the `Service Viewer` portlet.

When manually adding, replacing, or removing view definition XML files from the datastore, issue the `rad_reinitcanvas` command or restart the Dashboard server to update the runtime configuration.

Discovery Library Toolkit artifact categories

The following are the Discovery Library Toolkit artifact categories:

composites
This artifact category contains the composite definition XML files that are used to provide enhanced modeling features for processing resource, attribute, and relationship data that enters TBSM through the Service Component Repository (SCR). The composite definition capabilities include the definitions that can cause multiple resource instances to be viewed as one as well as the ability to introduce new resource instances and relationships into the model.

The Discovery Library Toolkit provides default composite definitions, in the `scrconfig` category, in the artifact named `CDM_TO_TBSM4x_MAP.xml`. This artifact category contains your additions to the default definitions. Any number of composite definition XML files can be added to the datastore and consumed by the Discovery Library Toolkit.

When manually adding, replacing, or removing composite definition XML files from the datastore, issue the `utils toolkit_reinit_and_resume` command or restart the Discovery Library Toolkit to update the runtime configuration.

eventidentifiers
This artifact category contains the event identifier rule definition XML files used to provide the mechanism for enriching resource instance information in the SCR with alternate identification strings, most commonly understood as `BSM_Identity` aliases, that can be used to map events and data fetcher results to resource instances.

The Discovery Library Toolkit default event identifier rule definitions are in an artifact named `EventIdentifierRules.xml`. This artifact category contains your additions to the default definitions. Any number of event identifier rule definition XML files can be added to the datastore and consumed by the Discovery Library Toolkit. The event identifier rule definitions in the artifact named `EventIdentifierRules.xml` can be overridden by putting a new version of the artifact, with the same name, in the datastore using the `putArtifact` command.

When manually adding, replacing, or removing event identifier rule definition XML files from the datastore, issue the `utils toolkit_reinit_and_resume` command or restart the Discovery Library Toolkit to update the runtime configuration.

labeling

This artifact category contains the labeling rule definition XML files. These files are used to enable the assignment of display names for resources within the SCR that do not have a label when they are imported into the system. These rules also allow you to override display names if the discovery source display name is not sufficient.

The Discovery Library Toolkit default labeling rule definitions are in an artifact named `LabelingRules.xml`. This artifact category contains your additions to the default definitions. Any number of labeling rule definition XML files can be added to the datastore and consumed by the Toolkit. The labeling rule definitions in the artifact named LabelingRules.xml can be overidden by putting a new version of the artifact, with the same name, in the datastore with a `putArtifact` command.

When manually adding, replacing, or removing labeling rule definition XML files from the datastore, issue the `utils toolkit_reinit_and_resume` command or restart the Discovery Library Toolkit to update the runtime configuration.

notifications

This artifact category contains the notification rule definition XML files used to provide the mechanism for driving Impact policies based on the resource, attribute, and relationship data being imported into the SCR. TBSM does not include a default notification rule definition XML file. This artifact category should contain your definitions. Any number of notification rule definition XML files can be added to the datastore and consumed by the Discovery Library Toolkit.

When manually adding, replacing, or removing notification rule definition XML files from the datastore, issue the `utils toolkit_reinit_and_resume` command or restart the Discovery Library Toolkit to update the runtime configuration.

scrconfig

This artifact category contains Discovery Library Toolkit base configuration XML files whose required content spans multiple purposes. The names of the artifacts are:

- `CDM_TO_TBSM4x_MAP.xml`
- `CDM_TO_TBSM4x_MAP_Templates.xml`
- `CrossNamespaceNamingRules.xml`
- `NamingRules.xml`
- `OtherNamespaces.xml`

When manually adding, replacing, or removing Discovery Library Toolkit base configuration XML files from the datastore, issue the `utils toolkit_reinit_and_resume` command or restart the Discovery Library Toolkit to update the runtime configuration.

taddmfilters

This artifact category contains the filter file definition XML files that is used to control the amount of resource, relationship, and attribute information that the Discovery Library Toolkit loads from Tivoli Application Dependency Discovery Manager (TADDM). This artifact category contains your additions to the standard definitions. Any number of filter file definition XML files can be added to the datastore and consumed by the Discovery Library Toolkit.

When manually adding, replacing, or removing filter file definition XML files from the datastore, issue the `utils toolkit_reinit_and_resume` command or restart the Discovery Library Toolkit to update the runtime configuration.

tbsmattributes

This artifact category is the location in the datastore for the TBSM attribute definition XML files used to define the list of attributes held in the SCR that are passed through to the TBSM memory model. By default, the SCR passes a limited number of resource attributes into the TBSM memory. TBSM does not include a default TBSM attribute definition XML file. This artifact category is used to contain your definitions. Any number of TBSM attribute definition XML files can be added to the datastore and

consumed by the Discovery Library Toolkit. Prior to TBSM 6.1, this customization was accomplished by editing a SCR view named `view_componentAttributesLimited`.

To define additional attributes that pass from the SCR into the TBSM service memory model, create an XML file with the outer most XML element named `AttributeSieve` and add child elements to list the attributes. Attribute names must include the namespace designation of the attribute and can include wildcards. For example, the following content might be contained within a `tbsmattributes` category artifact:

```
<AttributeSieve>
<attr>cdm:ActivityName</attr>
<attr>tbsm:%</attr>
</AttributeSieve>
```

When manually adding, replacing, or removing TBSM attribute definition XML files from the datastore, issue the `utils toolkit_reinit_and_resume` command or restart the Discovery Library Toolkit to update the runtime configuration.

Origin attributes for customization artifact

Customization artifacts are defined in the datastore with an `origin` attribute. There are two origin values supported TBSM and CUSTOM. The TBSM origin is assigned to artifacts that are installed with TBSM and that determine the default behavior.

The CUSTOM origin is assigned to artifacts that contain customization that you have defined after installation and that either wholly or partially change the default behavior. The correct origin of an artifact must be set by the components that use it, in other words, the Data server, Dashboard server, Discovery Library Toolkit, and manual processes that use the command line utilities. When manually adding or replacing artifacts in the datastore, it is a good practice to use the `putArtifact` command because it ensures the proper setting of the CUSTOM origin.

If you want to edit an artifact that is currently in the datastore, retrieve the artifact from the datastore with the `getArtifact` command and return it to the datastore with the `putArtifact` command. If you want to add an artifact to the datastore, put the artifact in datastore with the `putArtifact` command.

Customization Artifact Runtime Activation

The point at which a customization artifact moves from the datastore to the runtime configuration of a process is defined by the process itself and occurs when the process loads artifacts from the datastore. When manually adding, replacing, or removing customization artifacts in the datastore using the command line utilities, it is important that you are aware of how to update the runtime configuration.

About this task

The *Customization artifact categories* section of this document outlines the procedure to update the runtime configuration for each artifact category.

Loading Data server customization artifacts

For Data server customization, depending on the category, artifacts are loaded from the database into the runtime configuration when one or more of the following procedures are performed:

- Restart the Data server
- Restart the Dashboard server.
- From the command prompt, issue the `rad_reinitcanvas` command:

Windows

```
cd %TBSM_HOME%\bin
rad_reinitcanvas.bat
```

UNIX

```
cd $TBSM_HOME/bin
rad_reinitcanvas
```

Note: You might have to log out and log in to TBSM. This is necessary only if the previous configuration was cached.

Loading Discovery Library Toolkit customization artifacts

For Discovery Library Toolkit customization, depending on the category, artifacts are loaded from the database into the runtime configuration when one or more of the following procedures are performed:

- Restart the Restart the Discovery Library Toolkit.
- From the command prompt, issue the `utils toolkit_reinit_and_resume` command:

Windows

```
%TBSM_HOME%\XMLtoolkit\bin\utils.bat -e toolkit_reinit_and_resume.xml
```

UNIX

```
$TBSM_HOME/XMLtoolkit/bin/utils.sh -e toolkit_reinit_and_resume.xml
```

Chapter 7. TBSM and IBM Dashboard Application Services Hub

You can visualize TBSM service data in the IBM Dashboard Application Services Hub console component of Jazz for Service Management.

You access the TBSM service model data in the IBM Dashboard Application Services Hub console by configuring a data provider connection to TBSM. You then configure widgets in the Dashboard Application Services Hub.

The Dashboard Application Services Hub is a component of Jazz for Service Management. You install the Jazz for Service Management as a separate application from TBSM as described in the Jazz for Service Management information Center here:

http://pic.dhe.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.psc.doc_1.1.0/psc_ic-homepage.html

Detailed reference and tutorial information for using the **TBSM Data Provider** from the **Dashboard Application Services Hub** console can be found on the **TBSM Developer Works** wiki at:

https://www.ibm.com/developerworks/community/wikis/home?lang=en#/wiki/Tivoli%20Business%20Service%20Manager1/page/Advanced%20Topics

Note: Deprecation for Jazz™ for Service Management Registry Services is due with version 1.1.3 and thus the Open Services Lifecycle Collaboration (OSLC) Hover Preview functionality, which uses this feature, is also deprecated, when used with versions or Jazz™ for Service Management equal to or greater than version 1.1.3.

Connecting to Your TBSM Service Model Data

Connect to the TBSM data provider.

About this task

Before you can visualize data, you must create a *connection* between your TBSM data provider and the Jazz for Service Management dashboard. The connection gives the dashboard access to the service model data contained in the TBSM data provider.

Security considerations

Your existing security implementation dictates how you configure remote access to your TBSM UI data provider and the service data that is available from the UI data provider.

Before you define a connection to the remote UI data provider, make sure that you understand how security is applied to services in the TBSM data model. For more information about how security is defined globally or at the service and template level, see the *Granting user and group permissions to templates and services* topic in the TBSM Service Configuration Guide.

TBSM Security for Remote Data Access

The TBSM UI data provider returns TBSM services in all its datasets. The services that are returned depend on the definition of the dataset, the parameters that are specified, and the security settings for the connecting user. For example, if a user has the global role **tbsmViewService,** then the user sees all services that are available in a specific dataset with specific parameters.

Note: Many of the datasets available from the TBSM UI data provider are based on templates and TBSM also supports security for templates. However, all users can see datasets that are listed for all templates, but these datasets are empty if the user does not have the required privileges to see the services that are associated with a specific template.

Selecting a user for the TBSM UI data provider connection

When making a connection to a remote TBSM UI data provider, you must first supply a user that can search the remote host for available UI data providers. Any user with access to the remote TBSM Dashboard server can access the TBSM UI data provider. The user you chose becomes the default user for later requests to the provider. Depending on the security configuration for the TBSM service model, this single default user may not allow access to all the required services.

For maximum flexibility, you should configure Single Sign On (SSO) in your environment. When the connection is made to the TBSM UI data provider you will be given the option to use the credentials of the user currently logged in to the Dashboard Application Service Hub (DASH). With SSO configured, you will be able to check this option and use the same security configuration for the users of both consoles.

For example, the default **tbsmadmin** user has the global roles that allow this user to view all services. Accordingly, if you use the **tbsmadmin** user in the UI data provider connection, any DASH user who accesses the UI data provider can see **all** TBSM services. Conversely, the default **tbsmadmin** user does not have permission to view services. If you use the **tbsmadmin** user in the UI data provider connection, the DASH user would **not see any** services.

However, if SSO is configured and you select to use the DASH user credentials, you will see the same services on the DASH console that you would for the same user in the TBSM console. That is, when the **tbsmop1** user is logged in to the DASH console, the configured pages would include only the services that are permitted for viewing by the **tbsmop1** user on the TBSM console. In this case, it does not matter what user was used to define the connection to the TBSM UI data provider.

Chapter 8. Performance tuning

This section describes where to find information about optimizing TBSM performance.

Before you begin

Tuning your operating system: TBSM is built on the WebSphere Application Server and you need to review information on how to tune your operating system.

For information on tuning your operating system, see:

https://www.ibm.com/support/knowledgecenter/SS7JFU_8.5.5/as_ditamaps/was855_welcome_express_dist_iseries.html

Data fetcher tuning

A data fetcher can be configured to cache the values it fetches. When a data fetcher caches these values, a configurable number of fetched rows are kept in memory. On the next fetch, if any of the rows are the same as those previously cached, the rows that have not changed are not processed against the service model. This can save significant processing time, for example, when the fetcher returns a large number of rows, or there is extensive processing against the service model for each fetched row. However, memory is used to cache the rows.

About this task

Use this procedure to turn on and off data fetcher caching and also to amend the number of fetched rows that are cached.

Note: If caching is turned on and you do not immediately see updates to scorecards and canvas views, it is possible that the visuals might have been added after a fetcher was run and had already cached values. The updated scorecards and canvas views are displayed when the values change. To view the updated values when the fetcher next runs, you must manually clear the cache. For information about clearing the cache, see the TBSM *Troubleshooting Guide*.

Procedure

1. To enable caching for a data fetcher, you must modify the properties file that defines the data fetcher. On the TBSM Data server, switch to the directory:

   On UNIX systems:

   `/opt/IBM/tivoli/impact/etc`

   On Windows systems:

   `C:\Program Files\IBM\tivoli\impact\etc`
2. The data fetcher properties file name is TBSM_`<fetchername>`.props, where `<fetchername>` is replaced with the name of your data fetcher. Add or modify properties as follows:
   a) Enable or disable caching by changing the value of the property `impact.<fetchername>.enablecaching=<value>` where `<value>` is `true` to enable caching or `false` to disable caching.
   b) By default 100 rows are cached. The cache size is defined by property `impact.<fetchername>.maxnumberofsavedrows=nnn`. Larger values for nnn mean greater memory usage, especially for large row sizes. To maximize the value of caching, this value must be set to the number of rows you normally expect to be returned by the fetcher, assuming that the memory requirements are not prohibitive.
3. After the properties are modified for a data fetcher, you must disable, then re-enable the data fetcher using the TBSM Console.

a) Open the **Data fetcher** tab of the Service Navigation portlet.

b) Right-click the data fetcher that was modified and click **Disable** and then right-click again and click **Enable**.

Disabling KPI driven numerical status updates

If numerical status updates driven by Key Performance Indicators (KPIs) are not required, you can disable them. These events are sent from the Data Server to OMNIbus. Disabling these updates can result in an improvement in event processing performance, as well as lower CPU utilization, on the Data Server and OMNIbus Server.

About this task

Note: If you disable the numerical status updates, no numeric status (purple) events are displayed in the **Rules** tab of the **Service Details** portlet.

Procedure

1. To turn off the updates, open the Impact interface and select the **Policies** tab.
2. Click **Policies** to launch the **Policies** portlet window.
3. To load the base policies, from the **Project** list, select **TBSM_Base**.
4. In the list of policies, double-click the `ServiceEventUpdater` policy to edit it.
5. Edit the policy from:

```
// Comment next 3 lines to avoid sending purple numeric rule status events
if (ObjectToCopy.Severity = 1) {
AddDataItem(Type, ObjectToCopy);
}

// Comment next 3 lines to avoid sending color status events of numeric rule
elseif (EventType = 4 AND ObjectToCopy.Severity <> 1) {
 AddDataItem(Type, ObjectToCopy);
}
```

   to

```
// Comment next 3 lines to avoid sending purple numeric rule status events
// if (ObjectToCopy.Severity = 1) {
//  AddDataItem(Type, ObjectToCopy);
// }

// Comment next 3 lines to avoid sending color status events of numeric rule
if (EventType = 4 AND ObjectToCopy.Severity <> 1) {
AddDataItem(Type, ObjectToCopy);
}
```

6. Click **Save** and close the Policy Editor.

Chapter 9. TBSM Metric Collection

Tivoli Business Service Manager (TBSM) is the first Dashboard Application Service Hub application to use the new Time Window Analyzer portlet. This portlet provides a framework for presenting data in a manner that makes it possible to infer meaningful information that can be acted upon with the goal of enhancing a customer's ability to better manage IT resources to the business needs. Specifically, the Time Window Analyzer portlet charts metric data, which is frequently referred to as key performance indicators (KPIs), overlaid with the awareness of relevant occurrences, or markers, that may shed light on how to interpret the measured values of the KPIs.

There are two main components involved in delivering the proper Time Window Analyzer experience: a visual presentation layer and a data collection, maintenance, and access layer. The focus of this section of the document is to discuss the key information related to the data collection, maintenance, and access of metric data.

Key concepts

The TBSM Metric Collection component implements a ‘recent history' repository for storing and maintaining metrics for analysis through the Time Window Analyzer views. Within TBSM, a *metric* is data pulled back from a data fetcher and placed into a service's rule attribute through its associated templates. However, any numerical rule attribute, including those attributes that are derived from other rule attributes can be saved as metrics and presented to the Time Window Analyzer User Interface for presentation. The TBSM Metric Collection runs as part of the TBSM Data server and is responsible for recording metric values into a relational database.

Metrics are maintained within the data store for a configurable period of time. For the life of that metric within the data store, it is maintained as originally captured until it eventually ages out of the data store. ‘Roll-up' techniques that are frequently used to lessen the number of captured data points as they age are not used by the Metric Collection component due to its nature as a near-term analysis tool. Warehousing the data is provided through alternate TBSM flows.

Only metrics (that is, TBSM numerical-rule attributes) configured to the Metric Collection component can be maintained in the data store. Since maintaining near-term history at frequent intervals requires additional system resources, it is recommended that you consider which metrics provide the most benefit to the analysis tasks enabled by their use with Time Window Analyzer rather than enabling metric collection for all rule attributes.

Metrics are stored in a relational database. At installation time, the schema for this database can be created in the same database as the TBSM model data, or a separate database can be created for this metric history schema.

The following section defines commonly used terms regarding the Metric Collection component.

Metric
: a quantification of a particular characteristic over a given period of time. Metrics are plotted by the Time Window Analyzer portlet

Metric data point or Metric instance
: a specific measurement, including the time it was measured, the metric that was being measured, the resource the metric is related to, and the measurement itself

Metric meta information or Metric metadata
: provides information necessary for the processing and viewing of metric data points. It includes the metric name, display name, description, and measurement frequency. Most importantly, it contains the settings that determine whether a metric is being maintained in the near-term history data store, whether state changes for the metric are captured as markers, or both.

TBSM metrics
: Within TBSM, a metric is associated with data pulled back from a data fetcher and placed into a service's rule attribute through its associated templates. However, any numerical-rule attribute,

including those attributes that are derived from other rule attributes can be saved as metrics and retrieved by the Time Window Analyzer User Interface for presentation. This approach allows values whose latest measurement, as viewed within a TBSM scorecard view, to be maintained historically for Time Window Analyzer portlet views.

TBSM Metric collection must be configured to recognize a particular rule attribute as a metric and to maintain its values in its near-term history data store.

In addition to the ability to capture any template rule attribute as a metric, TBSM also makes a service instance's overall state as a metric available. This metric is referred to as the **OverallAttribute** and has a value of `0` (*good state*), 3 (*warning state*), and 5 (*critical state*).

Administering Metric Collection

There are two aspects to administering the TBSM Metric Collection component. The first is the component itself and involves settings that control the way the Metric Collection process works independent of the particular metrics that are to be collected. The second aspect involves defining which metrics are to be collected and maintained by the TBSM Metric Collection component.

Controlling the metric collection process

The TBSM Metric Collection feature is started as a thread of the TBSM Data Server application. The following configuration changes can be made through the `TBSM_tbsm_metric_history_user.props` file.

- The feature can be turned on or off. If turned off, the metric collection is disabled independent of whether a particular metric is enabled for metric collection. If turned on, the metric collection feature actively monitors for metric value changes to record to the 'near-term' history data store. Metric meta definitions then determine whether a particular metric is being maintained historically.
- The length of time that a metric data point is kept in the 'near-term' history data store. Any data point that ages past the configured time period–specified in days–will be periodically deleted. This value also controls the oldest data point returned to the Time Window Analyzer portlet.
- In order to produce efficient writes to the data store, the metric collection holds metric data for a brief amount of time so that data can be flushed into the data store in 'large' sets. This interval is configurable. The metrics are not available for querying until they are written to the data store.
- Periodically, the metric collection runs a purge process to remove metric data points that have been aged out. This process is controlled through the property file.
- The Metric Collection feature can be configured to consolidate metric value changes within an interval into one recorded metric change in the database. The time interval is configurable and can be valuable to limit the number of database metric points that are written to the metric data store.

Any change to the property file requires a stop and start of the TBSM Data server in order for the changes to take effect.

The Metric Collection property file

This section highlights the main properties that can be changed to affect the behavior of the metric collection component.

The `TBSM_tbsm_metric_history_user.props` property file is located at `c:\Program Files\IBM \tivoli\impact\etc` or `/opt/IBM/tivoli/impact/etc`

The following table defines the key properties of `TBSM_metric_history_user.props`.

Table 145. Key properties of TBSM_metric_history_user.props

Name	Purpose	Default	Change if ...
impact.collectmetrichistory	Controls whether the Metric Collection feature is active on this TBSM Data server. If the value is `true` then the metric collection feature is actively looking for changes in metric values to record into the history data store. If `false`, the feature is disabled and no metric collection can take place. Querying of the metric database takes place independently of this setting.	true	the TBSM installation will not be collecting metrics, turn this property to `false`.
impact.purgemetrichistory intervalhours		24	If4 or 8, to databasehave the maintenanprocedure cerun proceduremultiple s musttimes per executeday more frequently due to performan ce issues with collecting and accessing metric history, use a smaller value, like

Table 145. Key properties of TBSM_metric_history_user.props (continued)

Name	Purpose	Default	Change if ...
impact.purgemetrichistory time		03:00 (24 hour clock)	Database pruning, and other maintenance procedures, can be costly to overall system performance. Therefore, it is best to align their schedule with off hours if at all possible. Change this value if 3:00 AM is not an acceptable time for database maintenance procedures.

Table 145. Key properties of TBSM_metric_history_user.props (continued)

Name	Purpose	Default	Change if ...
impact.storehistoryperioddays	Number of days to keep metric history data within the TBSM Metric History database	14 (days)	This value should be reduced if too many data points for 1 day is provides keeping data for 4 days to be useful to users of the TBSM Metric History database, such as those using the Time Window Analyzer feature of TBSM. This value should only be increased after assessing the possible performance issues that additional data may cause in TBSM.

Table 145. Key properties of TBSM_metric_history_user.props (continued)			
Name	**Purpose**	**Default**	**Change if ...**
impact.metrichistorycollectorsecs	Rather than writing metric data out to the TBSM Metric History database at the moment that the new metric value point is available, TBSM pools records into a cache that writes to the database on an interval consistent with this value. This technique will delay the availability of a particular metric up to this time interval before it is available to be returned in a query of the TBSM Metric History database. However, this technique provides a more efficient mechanism for persisting the data into the database.	31 (seconds)	

Table 145. Key properties of TBSM_metric_history_user.props (continued)

Name	Purpose	Default	Change if ...
impact.metrichistory collectioninterval	Consolidates data points that arrive within this interval into one data point. The time interval begins when a data point arrives. Any additional data points for the same metric within the number of seconds represented by this property will be consolidated into one data point with the last metric value being captured to the database.	30 (seconds)	Alter this value to increase, or decrease, the possible number of metrics that will be kept in the metric data store. With the default settings, a data fetcher changing a metric value every 15 seconds will only result in approximately one historical data point within the 30 seconds. If the value of this property is set to 15, then each data point would be saved into the data store.
impact.metricmetadefaultenabled	When defining a metric using RADshell commands, if the enabled flag is not initially provided, this property controls whether data collection for that metric will be enabled (collection will start) by default.	true	

Defining and enabling individual metric collection

Note:

With the exception of the `OverallAttribute` rule metric storage, which can only be set using the RAD shell commands provided in this topic, metric collection for individual rules can be achieved in the TBSM user interface. It is preferable that you work in the TBSM user interface rather than using the RAD shell commands provided. This information is provided for information and for advanced users.

For information about performing these tasks using the TBSM interface, see the *TBSM Service Configuration Guide*. For information about using the RAD shell commands, see the *Administering TBSM > Administering TBSM using the RAD shell tool* topic.

Through RAD shell commands, the following tasks can be accomplished:

- A metric is defined and identified to the Metric Collection component. Definitions include the unique metric name that associates the definition with a particular template name and rule-attribute name within the TBSM model, as well as a display name for the metric to be used within the Time Window Analyzer view.

  Metric names are defined as the TBSM template name plus the numerical-rule attribute name concatenated together in the form of *`templatename.rulename`*. The only exception to this rule is a metric named *OverallAttribute*, defined with a display name of *Status Changes* that identifies the overall state of an object.
- The near-term historical collection of a particular metric can be enabled or disabled.

  Since metric names are directly associated with the TBSM template name plus the rule attribute name, metric data collection is enabled or disabled at the rule level and affects all resources tagged with the template containing the rule. Any resource that has the identified template associated with that metric will have metric collection controlled by that definition.
- The collection of metric information for a particular TBSM service resource can be disabled or re-enabled.

  The ability to disable or re-enable the collection of a metric on a resource basis only allows the disabling of metric collection if the metric is enabled. Enabling a specific resource for metric collection on a metric that is disabled is not supported.
- A metric can be configured to generate a marker to record the time that a metric value causes the rule attribute to change its state.

  For example, consider a metric that is configured to change to a bad state when its value crosses `70`. As the metric is queried, a metric data point is captured each time the metric value changes (for example, `45`, `30`, `50`, `45`, `60`, and `65`). However, when the metric value crosses the `70` threshold (for example, `40`, `60`, `65`, `71`), the metric collection process will record the `71` value as a metric data point and then, acting as a marker provider, insert a marker into the Marker Repository to record the point at which the rule attribute changed state. This capability allows the analysis of metric behaviors potentially pointing to the need to lower or raise a threshold. For example, finding that a metric value tends to exceed `70`, then continue to `75` before promptly returning to a steady state value of `50`, may imply that a threshold value of `70` provides a warning too early. If that same metric goes from `75` to `80` and it almost always continues its upward trend, then a threshold of `78` may be a more appropriate warning.
- The Metric Collection component records only changes in metric values in its historical data store. However, metric display and access requirements require a more frequent confirmation of the metric value. The Metric Collection component can be configured to provide this confirmation of a value at a particular interval per metric. As a result, the following behavior is present when metric data is viewed within Time Window Analyzer:

  Consider a case where the metric being captured is the state of a resource. A particular resource entered the green/good state on November 24. That resource remained green until November 28 before entering a warning state. The TBSM Metric Collector records the metric as good on November 24 and warning on November 28. If the Time Window Analyzer requests to see this metric from the time period of November 24 through November 31, the marker will be returned so that the slope of the line graph accurately represents how long the metric stayed in the good state before turning into the warning

state. This is achieved by returning the last recorded data point on the configured frequency interval for the metric. In this example, the graphed slope will accurately show the metric staying at a particular value for an extended period of time before the change in value. Therefore, it will be clear that the resource was in a good state from November 24 through November 28, up to the point that it entered the warning state.

This frequency interval is useful when using the Time Window Analyzer portlet in ‘refresh' mode; a metric does not change frequently enough for the chart to be updated on a regular basis if only changes were to be graphed.

- List metric information that is defined to TBSM.
- Export metric information that is defined to TBSM, in RAD shell format.
- Start the Metric Collection purging process to disable metrics that have aged out.

The following list outlines the purposes and the corresponding RAD shell commands in order to implement the above tasks. Many of the commands have multiple signatures to simplify and vary their use according to their parameters. Command details are available in the RAD shell commands section.

Define metric meta information

- `addMetricMeta`
- `updateMetric`*`attributename`*

Enable or disable metric collection

- `enableMetricCollection`
- `disableMetricCollection`
- `enableServiceMetricCollection`
- `disableServiceMetricCollection`

List metric information defined to TBSM

- `listMetricMetaData`
- `listDisabledServicesMetricCollection`

Export metric information into RAD shell format

- `exportMetricMetaData`
- `exportDisabledServicesMetricCollection`

Start the Metric Collection purge process

- `purgeMetricDBOnDemand`

Default configuration

Some of the key default configuration settings for TBSM Metric Collection component are described below.

- At install time, an administrator decides whether the metric history is stored in its own database, or in the same database as the TBSM service model data.
- Whatever database is chosen, a schema called *TBSMHISTORY* is created that contains all the metric history tables.
- Metric data in the data store ages out after 14 days.
- The maintenance of the metric history data store is scheduled to run at 3:00 am local time of the TBSM Data server each day.
- TBSM **OverallAttribute** is defined as a metric and it is being maintained in the metric history database.
- Markers are being created whenever a service status changes, as indicated by the **OverallAttribute**.

These configurations are meant to provide a positive default experience when getting started with TBSM. You should exercise caution if many services are constantly changing state (**OverallAttribute** metric),

especially since these state changes cause markers to be generated. To disable this collection or enable the collection of other metrics, `rad_radshell` functions must be used.

Understanding the presentation of Metrics for viewing

This topic describes the presentation of metrics for viewing.

The metric datastore works within the TBSM Data server to collect data effectively and efficiently. To meet this objective, metric values are only captured into the metric datastore when its value changes. The Metric Collection feature interacts with TBSM rule attributes, which can derive their value from data fetchers or be aggregated from other rule attributes. As a result, the Metric Collection feature records metric values as it is maintained in the rule attribute itself.

The following repercussions, or limits, to this technique are typically seen when viewing metric data:

- If a data fetcher is turned off, the TBSM rule attribute maintains its last value and the Metric Collection feature reports the unchanging value indefinitely.
- If a metric is disabled for the Metric Collection feature for a period of time, viewing the metric reports the last recorded value as its *current* value until the metric data point is deleted from the database.

Important metric data store issues

This section discusses important issues concerning the metric data store.

Storage Capacity

Since its purpose is to maintain a historical record of metric values as TBSM sees them, the Metric Collection component has disk storage requirements that must be met in order for the system to remain stable. This section discusses storage capacity needs for the metric near-term history data. Since TBSM also maintains a marker repository, be sure to consider its storage capacity needs as well. Use the following formulas to estimate the disk space required for maintaining metric data.

Approximately one (1) megabyte (M) of disk space is required for every 5250 metric data points being maintained. A *data point* represents a unique combination of a resource identifier, a metric identifier, a measurement value, and the point in time when the measurement took place. So to estimate storage requirements, one must estimate the number of data points to be maintained.

- MDP = NR x NM x MPH x 24 x impact.storehistoryperioddays,

  where

  - *MDP* is the number of metric data points.
  - *NR* is the number of resources that metric data is being collected for.
  - *NM* is the average number of metrics being collected per resource.
  - *MPH* is the average data points maintained per hour per metric.
  - *24* is the number of hours in a day.
  - *impact.storehistoryperioddays* is the property that determines the length of time that a metric data point is to be retained (default is 14).
- Capacity = MDP/5250,

  where

  - *Capacity* is the size in megabytes that should be allocated for storing metric data.
  - *MDP* is the number of metric data points

For example, an installation with 1000 services each with five metrics that change twice an hour generates 240,000 data points a day and 3,360,000 over 14 days. Such a setup would require 640 MB of storage.

Configuring the number of connections to the metric data store

TWA user interface requests will eventually drive queries into the metric and marker data store. The connections to the database are allocated from a pool of connections that are shared among all users as well as some internal processes. If a connection to the database is not available when a Time Window Analyzer user interface request is made, the request will wait until a connection is available before continuing its process.

The exact number of connections needed to support Time Window Analyzer user requests is a complex algorithm that includes the number of users using Time Window Analyzer simultaneously, the number of metrics being graphed, the number of markers being returned, and the number of resources that have metric data associated with them. However, simply being aware that connection pooling could affect the response time for Time Window Analyzer views will allow an installation to consider increasing system resources to provide more available connections for Time Window Analyzer users.

Use properties to configure the number of connections to a database instance and datasource pool:

Number of connections allowed on a database or DB2 instance

DB2 allows you to set the number of connections to the database manager using the command:

```
UPDATE DATABASE MANAGER CONFIGURATION USING MAX_CONNECTIONS nnn
```

For a specific database, the command to limit the number of remote and local application connections is:

```
UPDATE DATABASE CONFIGURATION FOR databasename USING MAXAPPLS nnn.
```

These commands can be run from a DB2 command prompt using the db2 command. For more information about these commands and all other DB2 topics, see http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.doc/welcome.html.

Number of connections allowed in a datasource pool

This value will control the number of connections TBSM maintains in its connection pool for each of its defined datasources. By default, each datasource is allocated a connection pool of five connections.

You may need to increase this value if you find user response time lags when a number of users are viewing metric data simultaneously. You can do this by finding the datasource definition in the directory:

`/opt/IBM/tivoli/impact/etc`.

`C:\Program Files\IBM\tivoli\impact\etc`

The default datasource for the metric data is in the datasource file named `TBSM_TBSMMetricHistory.ds`. The number of connections in the connection pool for this datasource is defined by property `TBSMMetricHistory.DB2.MAXSQLCONNECTION`. If you modify this property you must restart the TBSM Data server.

Tuning

It is recommended that regular maintenance appropriate to the target database is scheduled. Consulting with the database administrator for the targeted data store is strongly recommended.

Creating a high availability metric data store and failover

The TBSM Metric Collection data store does *not* participate in TBSM's failover implementation. If a high availability solution is required for metric collection, target the TBSM Metric Collection component to a high availability database configuration.

All TBSM Metric Collection property files must be kept in sync between the primary and backup TBSM Data servers. The TBSM Metric Collection Component does not provide any automatic processes for syncing these files.

Best Practice Suggestions

When using RAD shell commands to define and enable metric data to TBSM, maintain a list of the commands for purposes of easily recreating the setup at another time.

Chapter 10. Tivoli Marker Repository Service

Tivoli Business Service Manager (TBSM) is the first Dashboard Application Service Hub application to use the new Time Window Analyzer portlet. This portlet provides a framework for presenting data in a manner that makes it possible to infer meaningful information that can be acted upon with the goal of enhancing a customer's ability to better manage IT resources to the business needs. Specifically, the Time Window Analyzer portlet charts metric data, which is frequently referred to as key performance indicators (KPIs), overlaid with the awareness of relevant occurrences or markers that may shed light on how to interpret the measured values of the KPIs.

There are two main components involved in delivering the proper Time Window Analyzer experience: a visual presentation layer and a data collection, maintenance, and access layer. The focus of this section of the document is to discuss the key information related to the data collection, maintenance, and access of marker data.

Key concepts

The Marker Service implements a repository for storing and maintaining marker data and an interface for marker clients, marker providers, and marker updaters. The application is a stand-alone Web service that is installed with the TBSM Data server and is accessible through the configured HTTP or HTTPS port for that server.

Markers are stored in a relational database. At install time, the schema for this database can be created in the same database as the TBSM model data, or a separate database can be created for this marker data.

Although not technically part of the Marker Service, the TBSM Marker Provider Service and a TBSM Marker Updater Service will be discussed in this section. TBSM Marker Provider Service implements the marker provider role to capture TBSM data into the marker repository. The TBSM Updater Service looks to add TBSM context to markers not provided by TBSM.

The following section defines commonly used terms regarding the Marker Repository Service.

Marker
: a point-in-time description of an occurrence. The occurrence is assumed to be important but not necessarily positive and is meant to mark a significant event. Its significance or value is determined solely by the identifying provider and ultimately by the user analyzing metric data in the light shed by the information in the marker.

Marker provider
: detects and forwards markers to the Marker Service. A *provider* is typically associated with a product or application that provides marker data. Marker providers must be registered to the Marker Service in order to introduce markers into the marker repository.

  Marker providers are identified by a class string and by an instance identifier. The provider class string is a simple mechanism for grouping marker providers by their type or purpose. A provider instance represents a specific instance of a provider class.

  TBSM plays the role of marker provider for capturing state change conditions for metrics which are enabled for marker creation.

  The Marker Service provides a command-line interface, a Netcool Impact Policy Library, and the Marker Service WSDL in order to simplify the development of marker providers.

Marker client
: accesses the Marker Service to query and gain access to the saved markers. The Time Window Analyzer portlet is a client of the Marker Service. The Marker Client Utility is a rich tool that can be used to implement applications for the *client*, *updater*, and *provider* marker roles.

Marker updaters
watches for the existence of new markers within the Marker Repository Service and is charged with enriching the context (categories) of the marker with information that was not available to the original marker provider.

TBSM plays the role of marker *updater* with the objective of assuring that markers that are not provided with a TBSM resource identifier are enriched with one if possible. It also optionally adds in the TBSM dependency hierarchy of the resource. For example, if the resource is in a particular business application, then the service instance name of the server and the business application is added to the marker.

Marker Attributes

A marker has the following attributes:

Summary
a brief string description of the marker. This field is used to capture an abbreviated version of the details of that marker. Up to 255 characters are allowed; however, the Time Window Analyzer portlet uses this field for view flyover text; therefore, limiting the summary to 50-75 characters is desirable.

The summary can include simple HTML tags to enhance formatting.

Details
a verbose string capturing detailed information about the marker. This field can be up to 500 characters.

The details can include simple HTML tags to enhance formatting.

Status Time
The time that the described marker took place or will take place.

Categories
a set of strings associated with a marker. Categories provide context to the marker that enhances a client's ability to correlate the significance of the event the marker is documenting with other markers and other relevant pieces of data, including resource identifiers. Categories categorize markers into one or more like groups enabling easy searching. At the minimum, a marker should include a category describing the type of marker and the resource identifier for the primary resource described in the marker.

Provider class
a string that identifies the instance of the provider class that inserted this marker.

Provider key
an identity that may help relate the marker back to the source object that was used to create it, if available.

Identifier
a unique value assigned to the marker by the Marker Service.

Marker categories

The ability to create associations between a marker and other markers and resources is of utmost importance to the Time Window Analyzer feature. Marker categories serve the purpose of categorizing, or tagging, a marker with semantics that can be easily searched and correlated. It is through categories that markers have relevance beyond the specific event it is detailing.

The following simple example captures many of the important concepts of marker categories:

A Linux server is upgraded to a new fix pack level. Once the upgrade is completed, IBM Tivoli Application Dependency Discovery Manager detects the upgrade and captures the information into its database. Since Tivoli Application Dependency Discovery Manager is configured to issue an event to OMNIbus when this server changes, it forwards a change event to OMNIbus as a means of notifying any interested parties of the configuration change.

Once the event detailing the configuration change to the Linux server arrives at OMNIbus, a marker provider policy executing in Impact reads the event from OMNIbus and starts its process. First the OMNIbus event is translated into a marker. Essentially, an appropriate marker time, summary, and detailed description are gleaned from the incoming event. In addition, the provider adds the following categories to the marker to enable context of the marker to extend beyond the information contained solely within the marker itself:

- A category tag is added to identify the marker as a change notification.
- A category tag is added to associate the marker with a particular resource.

After properly packaging up the marker, the provider requests the Marker Service to insert the marker into the repository.

Once the Marker Service receives the marker, it is available for client access. Without any categories, a marker can only be searched for by the time it occurred. However, due to the additional categories added to the marker, a client could search for all markers of a particular type (for example, change notifications), as well as all markers for a particular resource (for example, server1) or any combination of time, type, or resource identifier.

As demonstrated by this example, the implementation of categories enables a powerful–yet somewhat undefined–search and correlation mechanism. In order to facilitate category use, a category convention is used to provide context to the category value associated with the marker.

Category conventions

Though a powerful concept, unorganized implementations of categories could result in difficulties correlating markers with other information. At their essence, categories enable groups of markers to be associated with one another or some unrelated object. For example, a category can describe the type of marker it is associated with, the resource it is directly associated with, and the resources it may be indirectly associated with due to its direct association to another resource.

Consider a marker that marks the upgrade of server1 which runs in a cluster1 at the time the upgrade occurred. In that case, a marker could apply to server1 directly and cluster1 indirectly. Furthermore, any business service that relies on cluster1 may also be indirectly associated with that marker. To allow the category function to support the capturing of this information, as well as to enable users of marker category data to understand the intent of a particular marker category, conventions will be used to designate the context of a category. These conventions are not enforced but without them, fully using the power of categories will be difficult.

In general, category conventions follow this structure:

context:value

where context is a string that describes the information that is being provided by the *value* component of the category tag.

The degree of success of this technique is directly related to consistent use of category context values.

Category Types

Category types identify the type of event captured by the marker. Convention also allows the detailing of a subtype for a given marker. These types of categories are very important to the Time Window Analyzer portlet since these marker categories can be identified as ‘overlays' on the view.

The following category-context values should be used to identify markers of the described type. If a marker is being created to capture a marker of a different type, extend the list and enforce its use.

Table 146. Category Context Values

Type or Subtype	Description	Providers	Category Tag Value
OverallAttribute	The marker describes a service overall state change.	TBSM	type:OverallAttribute

Table 146. Category Context Values (continued)			
Type or Subtype	**Description**	**Providers**	**Category Tag Value**
STATEYtoR	The marker describes a state change from yellow to a red – warning to bad.	TBSM	subtype:STATEYtoR
STATEGtoR	The marker describes a state change from green to a red – good to bad.	TBSM	subtype:STATEGtoR
STATERtoY	The marker describes a state change from red to yellow – bad to warning.	TBSM	subtype:STATERtoY
STATERtoG	The marker describes a state change from red to green – bad to good.	TBSM	subtype:STATERtoG
STATEGtoY	The marker describes a state change from green to yellow – good to warning	TBSM	subtype:STATEGtoY
STATEYtoG	The marker describes a state change from yellow to a green – warning to good.	TBSM	subtype:STATEYtoG
CHANGEEVENT	The marker describes a configuration change such as the installation of a patch or a configuration file change.	TADDM change event Impact provider	type:CHANGEEVENT
PROBLEMTICKET	This marker describes a problem ticket	Future use	type:PROBLEMTICKET

Overlay Category Types: Category types are of extra significance because they are used by the Time Window Analyzer portlet to group markers for visualization purposes. Define the appropriate category values as an ‘overlay' to make them available to Time Window Analyzer. See the `CreateCategory` command-line option to the Marker Web Service for detail. Be sure to consistently use the naming convention for overlay categories in the CreateCategory command–using the *id* parameter–and when creating a marker with a particular category type.

The *type* and *subtype* categories also provide a linkage to visuals shown within the Time Window Analyzer portlet.

Category resource identifiers

These categories include a namespace that scopes the resource identifier into the naming domain where the identifier value has meaning. This convention must be used to allow clients of marker data–including the Time Window Analyzer portlet–to successfully associate a marker with a particular resource.

The following category context values will be used to identify resource identifiers and their respective domain. These values are available for any marker provider implementation to use.

Table 147. Category context values to identify resource identifiers			
Namespace	**Description**	**Providers**	**Category tag value**
tbsm	Resource identifier is a TBSM service instance name	TBSM	tbsm:ET_Trader
guid	Resource identifier is a Common Data Model Globally Unique Identifier (GUID)		guid:09085C7AC48F3D9E83CF6389797C0C02

Table 147. Category context values to identify resource identifiers (continued)

Namespace	Description	Providers	Category tag value
taddm	Resource identifier is a Tivoli Application Dependency Discovery Manager resource identifier	Impact policy that turns change events from Tivoli Application Dependency Discovery Manager into markers	taddm:4DBE78368BC131D68151B7A33C587B31
itmmsn	Resource identifier is an ITM-managed system name		Itmmsn:Primary:CVTWIN44:NT
alias:	Resource identifier is a TBSM identifier available for event mapping. For alias resource identifiers, the value consists of the field name and value separated by an equal (=) sign		alias:BSM_Identity=host.raleigh.ibm.com alias:Node=host
itnmip	Resource identifier as provided in source token value of an IBM Tivoli Network Manager IP Edition IDML book		Itnmip:2153&=ITNM3823

The TBSM Updater service uses namespace conventions to attempt to resolve different namespace resource identifiers to a TBSM service instance name. See "Administration of the TBSM Updater Service" on page 239 for more details.

Administering the Marker Service

There are two aspects to administering the TBSM Marker Service. The first involves administering the service itself. The second aspect involves defining the marker data it manages and managing the repository content.

Controlling the marker server process

The TBSM Marker Service is started as an application on the TBSM Data Server. Since the TBSM Data server executes in a WebSphere Server, stopping the Marker Service independently of the TBSM Data Server application requires the use of the WebSphere `wsadmin` commands.

The following configuration changes can be made through the `TBSM_markerserver.props` file.

- The Marker Service periodically ages out marker instances and related information from the marker data store. The time when this purge process takes place is configurable.
- The length of time that a marker will be kept in the data store. Any marker whose status time ages past the configured time period–specified in days–will be deleted during the marker purge process.

Any change to the property file requires a stop and start of the TBSM Data server in order for the changes to take effect.

The Marker Service property file(s)

This section highlights the main properties that can be changed to affect the behavior of the marker service.

The TBSM_markerserver.props property file is located at

```
/opt/IBM/tivoli/impact/etc
```

Table 3 defines its key properties.

Table 148. Key properties of TBSM_markerserver.props

Name	Purpose	Default	Change if ...
purgemarkerhistoryinterval hours	The frequency in hours to purge the marker repository database of old data.	24	Purging data on a more frequent time interval may improve the performance of this procedure.
purgemarkerhistorytime	Specifies a specific time each day to start the purge process. This value is only considered if the *purgemarkerhistoryintervalhours* is set to a value outside the range of 1 and 24. If this value is set and *purgemarkerhistoryintervalhours* is set to a value greater than 24, the time of a purge will be *purgemarkerhistoryintervalhours* ahead of the time that the last purge occurred forced to the specific time of day specified by this parameter. If this value is set and purgemarkerhistoryintervalhours is 0, the time of a purge will be at the time of day indicated by the *purgemarkerhistorytime* property.	(unset)	Controlling the actual time of day that a purge occurs is needed.
storehistoryperioddays	Number of days to keep marker data before purging takes place.	14 (days)	This value should be reduced if keeping 14 days of marker instances proves to be adversely affecting performance or if the storage capacity requirements are too high.

Administering markers in the Marker Repository

In addition to administering the Marker Service application through its property file, it is necessary to administer some aspects of the data the Marker Service maintains–provider definitions, category definitions, and the markers themselves.

Using the Marker Client Utility, the following tasks can be accomplished:

- Marker provider classes and instances are defined to the Marker Service.

  Without these definitions, markers cannot be introduced into the marker data store.
- Categories can be defined to the Marker Service as overlays for use by the Time Window Analyzer portlet.
- Marker provider classes and instances can be listed from the Marker Service data store.
- Marker provider classes, instances, and their related markers can be deleted from the Marker Service data store.
- Markers can be queried, saved to a file, inserted, and deleted.
- The Marker Service prune process can be invoked.
- Categories can be added or removed from a set of markers.

For information regarding Marker Client Utility commands, refer to "Using the Marker Client Utility" on page 250.

Default configuration

Some of the key default configuration settings for the TBSM Marker Service are outlined in this topic.

These include:

- The Marker Web service is installed with the TBSM Data server and is started and stopped with that server.
- The Marker Service uses the HTTP protocol for communications. It is listening on port 17310 by default.
- At install time, an administrator decides whether the marker data is stored in its own database, or in the same database as the TBSM service model data.
- Whatever database is chosen, a schema called *TWAMARKER* is created that contains all the marker data tables.
- Markers are aged out of the data store after 14 days.
- The marker data store pruning process is scheduled to run every 24 hours.
- Marker provider class and provider class instance definitions are added for the TBSM provider.
- Marker category overlay types are defined for the TBSM `OverallAttribute` state change marker.

The importance of Marker Providers

The importance of markers to the Time Window Analyzer solution cannot be over-stated. Without it, the Time Window Analyzer is not much more than a graphing utility. With proper use of markers, the Time Window Analyzer enables the correlation of disparate information which has not been previously available to Dashboard Application Service Hub users, resulting in improved service.

TBSM provides marker collection for changes in the overall status of a resource, as well as for status changes that result from a template rule attribute that exceeds its defined thresholds. In addition, an Impact policy is provided as a guideline for catching change events from Tivoli Application Dependency Discovery Manager and forwarding them as markers. However, much more value returns if you use the Marker Provider tools for other significant events.

Consider implementing Marker Providers for:

- Additional change markers from product sources other than Tivoli Application Dependency Discovery Manager
- Marketing promotion events (beginning, ongoing, and ending) that might help explain increased transaction rates
- Relevant business calendar events
- Market events directly related to services and products you produce
- State, national, and global events that may affect your business

Building Marker Providers

A key design point of the marker solution enables third parties to enrich the solution by providing valuable markers. In order to accomplish this goal, the Marker Service provides a command-line interface, a Netcool Impact Policy Library, and the Marker Service WSDL, in order to simplify the development of marker providers. Each technique provides similar capabilities as a *provider*; however, depending on the integration scenario, one technique may be a better fit than another.

In general, the Impact solution can be used when the data that is the source of the marker is accessible through an Impact data source interface, including OMNIbus events or relational database tables that can be easily queried through Impact.

The Marker Client Utility can be used in just about any scenario where the source of the marker data is readily available and a simple utility is all that is needed to forward the marker instance to the Marker Service. The utility is useful if markers need to be introduced into the system manually, especially since a .tar or .zip file is provided for moving the command-line utility to different systems.

The Marker Web service WSDL specifies the Marker Web service interface in detail. For advanced marker *providers* that cannot be implemented using Impact or a command-line utility, the WSDL can be used to interface directly to the Web service. Unlike the other provider technologies, this implementation may be sensitive to changes in the Marker Web service interface and should only be used if the other implementations do not sufficiently meet the needs of the provider being built. Using a WSDL to implement a direct Web service implementation to the Marker Service is beyond the scope of this document.

Registering a Marker Provider

A marker *provider* class and marker *provider* instance must be registered with the Marker Service in order for it to successfully introduce markers into the repository. These two steps can be done at any time in the life of a provider and can be accomplished through execution of the marker command-line utility on behalf of any other provider implementation.

To successfully register a marker provider, issue the following Marker Client Utility commands.

- RegisterMarkerProviderClass
- RegisterMarkerProvider

The marker provider must provide the registered marker provider class name and provider instance ID when inserting markers into the Marker Service.

Using the Marker Command Line Utility to build a provider

The Maker Command Line Utility is the most basic way of implementing a marker provider. It simply provides a command-line capability for interacting with the Marker Service. Since it simply provides an interface to the Marker Service, the use of the utility must determine how to interface the utility to a source of data that can be converted to markers. In the simplest case, the source of data could be a person.

Since the Marker Client Utility allows markers to have a status time in the past or future, the command line utility may be useful for system administrators to add markers that may shed some light on metric behavior whenever needed. When used in the *provider* role, the `InsertMarker` command is most relevant.

The Marker Client Utility can be copied to systems other than the one TBSM installs it on.

Using the Netcool Impact Marker Provider library to build a provider

TBSM provides a set of documentation, procedures, and policies that opens the power of Tivoli Netcool Impact platform to the possibilities of mining markers from just about any data source accessible though Impact. The Marker Service policies provide a foundation for interacting with the Marker Service primarily as a marker *provider*. Impact provides access capabilities and the policy language for interacting with a data source and programmatically manipulating the data to convert it into a marker.

The package includes a helper function policy library that interacts directly with the Marker Web service and additional policies for registering providers, inserting test markers, and capturing change events from Tivoli Application Dependency Discovery Manager and turning them into markers. In addition, instructions describe how to install and configure these tools for proper use in Impact.

TBSM as a Marker Provider

When configured to do so, TBSM provides markers each time a resource changes its overall state and when a particular metric rule on a template causes a state change due to exceeding thresholds defined for the metric rule.. Controlling this behavior is controlled by the RAD shell commands used to define metric metadata for recent historical data collection. See the RAD shell command `enableMetricMarkers`.

Consider using these TBSM status-change markers to help tune status-propagation rules and to analyze the effect of a particular metric on the overall state of a resource. However, please be aware of the volume of markers that may result.

Administration of the TBSM Updater Service

The previous section, "Building Marker Providers," discusses techniques to be used for the implementation of marker providers. In many cases, the marker provider presents a resource identifier in a namespace consistent with the source of the data that created the marker. Unless that resource identifier is correlated with a TBSM service instance name, the Time Window Analyzer will be unable to associate the marker with collected metric data; therefore, the markers will not be displayed.

To address this issue, TBSM implements a marker *updater* service, which watches for new markers introduced into the Marker Service data store and attempts to correlate the marker resource identifier to a TBSM service instance name. If successful, the TBSM updater service updates the marker with the TBSM service instance name and, optionally, tags the marker with the complete resource hierarchy path.

To achieve this action, the following steps are taken:

1. Within the TBSM Data server, the TBSM Marker Updater process polls the Marker Service for new markers.
2. If the marker already has a category value of `MARKERPROCESSED`, the TBSM Marker Updater does not process the marker.
3. If the marker does not have a TBSM resource identifier as denoted by the `tbsm:` prefix of a category value, then the TBSM Marker Updater service attempts to match the resource identifiers that are provided with the marker to a TBSM resource identifier.
   a. For each category resource identifier, as listed in the **"Marker Categories"** section, attempt to use that information to find the TBSM service instance name for that resource.
   b. If the TBSM service instance is found for the marker, update the marker with an additional category tag of the format `tbsm:`*`tbsmserviceinstancename`*.
4. If the marker has a category with a TBSM resource identifier as denoted by the `tbsm:` prefix of a category value and does not have the category value of `NO_TBSM_HIERARCHY`, the TBSM Marker Updater adds the dependency hierarchy resources as categories.
5. Add a category value of `MarkerProcessed` to the marker to indicate that the TBSM Marker Service has processed this marker.

Configuring the TBSM Marker Updater process is controlled by the property file named `TBSM_tbsm_marker_user.props`.

The TBSM Marker Updater property file

Table 1 table defines key properties of file `TBSM_tbsm_marker_user.props`, located at `C:\Program Files\IBM\tivoli\impact\etc` or `/opt/IBM/tivoli/impact/etc`.

Table 149. Key Properties of TBSM_tbsm_marker_user.props			
Name	**Purpose**	**Default**	**Change if**
impact.markerupdater.active	At TBSM startup, this property controls whether the TBSM Marker Updater service starts or not.	true	Change to false, if all Markers being collected already contain TBSM service identifiers and their affecting services (ancestors of the directly related service)
impact.markerupdater.pollintervalsecs	Controls the time interval between executions of the TBSM marker updater process.	87 (seconds)	The updater service lags behind in its processing of markers being added to the repository. The TBSM log file (`trace.log`) writes trace records that details the processing being done.
impact.markerupdater.categ ory.include	The list of marker category values that must be associated with a marker in order for the TBSM marker updater to process it.	%	The performance of the TBSM Marker Updater Service requires limiting what it looks for in the Marker Repository.
impact.markerupdater.categ ory.exclude	A list of marker category values that cannot be associated with a marker in order for the TBSM marker updater to process it. This property simply provides an alternative to the *impact.markerupdater.categor y.include* property using the negative condition rather than the positive.	<no exclusions>	The performance of the TBSM Marker Updater Service requires limiting what it looks for in the Marker Repository.

Netcool Impact Marker Provider Library

TBSM provides a set of documentation, procedures, and policies that opens the power of the Tivoli Netcool Impact platform to the possibilities of mining markers from just about any data source accessible though Impact. The package includes:

- `MarkerImpactWSLibrary.ipl` policy library that includes the policy function named `SendMarkerToRepository` which simplifies the sending of marker information to the Marker Web Service. There are two versions of this policy. The installed version is used with a TBSM Data server. Another version exists in the `$TBSM_HOME/contrib/markerimpactprovider` directory and is used with servers that are not TBSM or Impact.
- The `markerRegisterProviders.ipl`, which registers the provider that is used by the `markerChangeProvider` policy. This policy must be executed for both TBSM and non-TBSM Impact servers before using either the `markerChangeProvider.ipl` or `markerSampleSimpleProvider.ipl` policies.
- The `markerChangeProvider.ipl`, which is an implemented marker provider policy for Tivoli Application Dependency Discovery Manager change events. This policy can also be used as a sample for building any Impact-based policy provider.
- The `markerSampleSimpleProvider.ipl` which is a sample provider policy to the `MarkerImpactWSLibrary.ipl` policy library which exercises the `MarkerImpactWSLibrary.ipl` policy library and can be used to test for a proper configuration.

- Impact can be configured as a marker provider, both from the Impact embedded inside TBSM, and from a standalone Impact server. The following procedure is necessary only when configuring a standalone Impact server as a provider. To configure the TBSM server as a provider, the only step to be done is to create the impact internal data type and data item in the type as described below.
- Instructions found in the *TBSM as a Marker Provider* section of this document describe how to install and configure these tools for proper use in Impact. However, it is strongly recommended that you refer to Impact documentation concerning the implementation of a Web service DSA. Instructions include information about the following options:
  - Compiling the Marker Web Service WSDL and making the resulting jar files available for Impact policy use (only required when setting up the `MarkerImpactWSLibrary.ipl` on a non-TBSM Impact server).
  - Importing the `MarkerImpactWSLibrary.ipl` policy library
  - Creating an Impact internal data type for defining key runtime parameters for successful use of the policy library, including the host name URL where the Marker Web service is installed where the Marker Web service is installed (only required when setting up the `MarkerImpactWSLibrary.ipl` on a non-TBSM Impact server).
  - Importing and executing the `markerRegisterProviders.ipl` policy
  - Importing and testing the sample policy
  - Importing the `markerChangeProvider.ipl` policy

Compiling the Web service WSDL into Impact

This step creates Impact functions that can be used from policies to interact with the targeted Web service. In this case, compiling the Marker Service WSDL creates the functions that are used in the `MarkerImpactWSLibrary.ipl` library. Custom marker provider policies indirectly interact with the output of these steps through the `MarkerImpactWSLibrary.ipl` library functions.

Note: This procedure is only necessary if you are setting up the Impact Marker Provider Library on a non-TBSM Impact server. If you want to use the marker provider library on the TBSM data server (which is also an Impact server), this procedure is unnecessary.

Compiling the WSDL with the targeted installation of Impact, rather than installing a precompiled library, lessens the chance of level mismatches that could interfere with a successful implementation.

The Impact command `nci_compilewsdl` must be issued against the Marker Service WSDL. The first step is to make a copy of the Marker Service WSDL file by completing the following steps:

1. In a web browser type the following URL:

   ```
   http://hostname:port/markerWeb/services/QueryPort
   /wsdl
   ```

   where

   - *hostname* is the host name of the TBSM Data server running the Marker Service
   - *port* is the port number to connect to. By default the port is 17310
   - If the TBSM Data server is configured to use HTTPS, use the HTTPS protocol to get the WSDL. The default port for HTTPS communications with the TBSM Data server is 17311.
2. When prompted for the user ID and password, type the Dashboard Application Service Hub administrative user ID, by default `tbsmadmin`, and the corresponding password.
3. When the browser displays the WSDL XML file, save the XML file to an accessible location. The saved WSDL file can then be used as input to the `nci_compilewsdl` command.

After you have copied the Marker Service WSDL to the marker system, run the `nci_compilewsdl` command as follows:

```
nci_compilewsdl TBSMMarkerService  path/MarkerService.wsdl $TBSM_HOME/wslib
```

where

- *path/* is the file path to the WSDL file

The `nci_compilewsdl` command creates a jar file named `TBSMMarkerService.jar` and places it in the `$TBSM_HOME/wslib` directory to enable Impact to interact with the Marker Web service.

After the jar file is moved into the correct directory, stop and start the Netcool Impact server so that the jar is available to it at runtime.

As an alternative to using `nci_compilewsdl`, you can use the web service wizard in the Impact UI to compile the `wsdl` file, and avoid having to restart the Impact server.

Importing and configuring `MarkerImpactWSLibrary.ipl`

The `MarkerImpactWSLibrary.ipl` policy library provides a simple-to-use interface for sending markers to the Marker Repository. It envelopes the necessary procedures required to properly interact with the Marker Web service, allowing marker 'collectors' to focus on acquiring markers from different data sources rather than the idiosyncrasies of dealing with the Web service.

Note: This procedure apply where you want to install the `MarkerImpactWSLibrary.ipl` policy to a non-TBSM Impact server. The TBSM Data server already has a version of this policy installed by default and therefore this procedure is not necessary. If you want to install this policy onto a non-TBSM Impact server, the file is located in the `$TBSM_HOME/contrib/markerimpactprovider` directory.

To import the `MarkerImpactWSLibrary.ipl` policy into Impact, complete the following steps:

1. Log in to Impact. Optionally, create a project that will hold all the policies and other definitions needed for the implementation of a marker provider.
2. Open the **Policy** tab within the project. Select the **Import a Policy** option towards the bottom of the policy list.
3. Follow the directions to import the `MarkerImpactWSLibrary.ipl` policy. The file must be located on the same machine running the browser used to connect to the Impact user interface.

The primary function within the library is `SendMarkerToRepository`. Its call signature is described as follows, and it is referenced from another policy by using a *`policyname.function`* format MarkerImpactWSLibrary.SendMarkerToRepository.

```
Function SendMarkerToRepository(
        in_providerclass,
        in_providerinstance,
        in_dateString,
        in_dateUTC,
        in_summary,
        in_detail,
        in_providermarkerkey,
        in_markerResourceId,
        in_categorylist)
```

where

`in_providerclass`
: a string identifying the class of the provider. The provider class is simply a string that allows categories of provider instances. If provided as `NULL`, this value defaults to `ImpactProviderClass`.

`in_providerinstance`
: a string identifying the instance of the provider. The provider class is simply a string that allows the identification of a particular provider within its provider class. If inputted as `NULL`, this value will default to `ImpactProvider`.

`in_dateString` and `in_dateUTC`
: are used to capture the time that is associated with the marker. If both parameters are passed in as `NULL`, the marker is said to have occurred at the current time. If `in_dateString` is provided, it must be a string in this format of `yyyy-mm-dd HH:mm:ss timezone` . Parameter `in_dateUTC` allows the caller to provide the current time in UNIX Epoch time, which is the number of seconds since January 1, 1970. The `in_dateUTC` time is only considered if the `in_dateString` value is `NULL`.

in_summary
a brief string description of the marker. This field is used to provide a summary of the marker in as few lines as possible. Up to 255 characters are allowed; however, the Time Window Analyzer portlet uses this field for user interface flyover purposes. Therefore, limiting the summary to 50-75 characters is recommended.

in_detail
a detailed string description of the problem. This field can be up to **500** number of characters.

in_providermarkerkey
a string identifier for the source that resulted in the marker. Future use of this field would be to link the marker record back to the original source event.

in_markerResourceId
this string provides the primary resource entity associated with the marker. If multiple resources should be associated with this marker, additional identifiers can be passed in the `in_categorylist` parameter.

in_categorylist
an Impact DataItem/Orgnode type which contains a list of 'categories' for this marker. Markers can be found or searched for using a list of categories as their search criteria.

Categories are critical for the successful association of a particular marker with a set of resources, a type of data, or any other criteria.

Defining runtime settings for the `MarkerImpactWSLibrary.ipl` library

Note: This procedure apply where you want to install the `MarkerImpactWSLibrary.ipl` policy to a non-TBSM Impact server. The TBSM Data server already has a version of this policy installed by default and therefore this procedure is not necessary. If you want to install this policy onto a non-TBSM Impact server, the file is located in the `$TBSM_HOME/contrib/markerimpactprovider` directory.

The following runtime parameters are needed by the `MarkerImpactWSLibrary.ipl` policy library in order to successfully interact with the Marker Web service.

- Resulting package name from `nci_compile.wsdl` command.
  - For example, `TBSMMarkerService`
- The URL required to locate the Marker Web service up to–and including–the port number.
  - The Marker Web service is installed on the TBSM Data server and–by default–port `17310` is used for access
  - For example, http://*hostname.mycity.company*.com:17310
- Username and password. The default Dashboard Application Service Hub administrative user ID, `tbsmadmin` and corresponding password can be used.

These runtime parameters are to be defined as an Impact internal data type named `MarkerWebServiceInfo`.

Creating the `MarkerWebServiceInfo` internal data type

The following instructions describe the steps needed to properly define the `MarkerWebServiceInfo` internal data type.

Note: This step is required when configuring a TBSM server to function as a marker provider.

Note: This procedure apply where you want to install the `MarkerImpactWSLibrary.ipl` policy to a non-TBSM Impact server. The TBSM Data server already has a version of this policy installed by default and therefore this procedure is not necessary. If you want to install this policy onto a non-TBSM Impact server, the file is located in the `$TBSM_HOME/contrib/markerimpactprovider` directory.

These instructions are provided only as a guide. Impact documentation should be consulted for more thorough knowledge of internal data types and how they work within the Impact system.

At the end of this process the following definitions will be created for `MarkerWebServiceInfo` internal data type:

Table 150. `MarkerWebServiceInfo` definitions

	Field 1	**Field 2**	**Field 3**	**Field 4**
ID	packagename	URL	username	password
Field name	packagename	URL	username	password
Format	String	String	String	String
Display name	WSDL Package Name	Marker Web Service URL	Marker Web Service Authorized Username	Marker Web Service Authorized User Password
Description	The Impact package name that the Marker Web service was compiled into.	The Web Service URL including host name and port.	The name of a user authorized to access the Marker Web Service, for example, `tbsmadmin`.	The password for the user authorized to access the Marker Web Service

And a single data item for the `MarkerWebServiceInfo` data type will be created:

Key

AccessInfo

WSDL Package name
the package name that WSDL is compiled into which must be named `MarkerService`.

Marker Web Service URL
the URL where the Marker Web Service is installed (for example, http://*hostname*.*mycity*.*company*.com:17310)

Marker Web Service authorized username
The name of a user authorized to access the Marker Web Service, for example, the Dashboard Application Service Hub user ID, `tbsmadmin`

Marker Web Service authorized user password
The password for the user authorized to access the Marker Web Service

Step-by-step directions

Begin by logging in to the Impact Web interface.

1. In the navigation tree, click **System Configuration > Event Automation > Data Model** and select **Global** from the **Project** list.
2. In the **Data Sources and Types** list, click to expand the Internal data type.
3. Verify that a `MarkerWebServiceInfo` internal data type has not been defined. Right-click **Internal** and click **New Data Type**.
   a. Specify `MarkerWebServiceInfo` as the Data Type Name.
   b. Check the **Persistent** property
   c. Add a field:
      i) ID: `packagename`
      ii) Field name: `packagename`
      iii) Format: `String`
      iv) Display Name: `WSDL Package Name`
      v) Description: The Impact package name the Marker Web Service was compiled into.

d. Add another field:
   i) ID: URL
   ii) Field name: URL
   iii) Format: String
   iv) Display name: Marker Web service URL
   v) Description: The Web Service URL including hostname and port.
e. Add another field:
   i) ID: username
   ii) Field name: username
   iii) Format: String
   iv) Display name: Marker Web Service authorized username
   v) Description: The name of a user authorized to access the Marker Web Service, for example, tbsmadmin.
f. Add another field:
   i) ID: password
   ii) Field name: password
   iii) Format: String
   iv) Display name: Marker Web Service Authorized user password
   v) Description: The password for the user authorized to access the Marker Web Service.
g. Save the new data type.

4. Return to the list of Internal Data Types. The MarkerWebServiceInfo should now be listed.
5. Choose the **View Data Items** for MarkerWebServiceInfo icon.
6. Add the following data item:
   a. Key: AccessInfo
   b. WSDL Package Name: (the package name that the WSDL is compiled to) for example, TBSMMarkerService
   c. Marker Web Service URL: (the URL where the Marker Web Service is installed) http://*hostname.mycity.company*.com:17310
   d. Marker Web Service Username: (Marker Web Service authorized user ID), for example, the Dashboard Application Service Hub user ID, tbsmadmin
   e. Marker Web Service encrypted authorized user password.

      Note: To encrypt your user password, run the install_home/tbsm/bin/nci_crypt *password* command. Where *password* is the password you have chosen. This command generates an encrypted password string that you can enter for the authorized user password.
7. Save the item.
8. In the navigation tree, click **System Configuration > Event Automation > Policies** and select **TBSM_BASE** from the **Project** list.
9. Select MarkerImpactWSLibrary and click **Edit**.
10. In this policy, there are 4 instances of the line:

```
callProps.Password=WSPassword;
```

Locate each line and edit it to read:

```
callProps.Password=decrypt(WSPassword);
```

11. Click **Save**.
12. The process is now complete.

Configuring Additional Data Types For TBSM Failover Configuration

If the TBSM Data server that runs the Tivoli Marker Service is configured for failover, two additional data items will be created for the MarkerWebServiceInfo data type, which enables the marker web server policy library to flip between the primary TBSM Data server and the backup TBSM Data server if necessary.

The additional data items are:

Key
primary

WSDL package name
The package name that WSDL is compile into (`MarkerService`).

Marker Web service URL
The URL where the Marker Web Service is installed on the primary TBSM Data server

Marker Web Service authorized username
The name of a user authorized to access the Marker Web Service, for example, the Dashboard Application Service Hub user ID, `tbsmadmin`

Marker Web Service authorized user password
The password for the user authorized to access the Marker Web Service

and

Key
secondary

WSDL package name
the package name that WSDL is compiled into (`MarkerService`).

Marker Web service URL
the URL where the Marker Web Service is installed on the backup TBSM Data server

Marker Web Service authorized username
The name of a user authorized to access the Marker Web Service, for example, the Dashboard Application Service Hub user ID, `tbsmadmin`

Marker Web Service authorized user password
The password for the user authorized to access the Marker Web Service

Note:

- Define the primary and secondary data items in the same manner that was described to define the AccessInfo data item.
- Initially, the AccessInfo and primary data items should match exactly for all fields except the Key itself.
- If the marker Web service library detects a problem contacting the server configured in the AccessInfo data item, it will switch its URL value to the URL defined in the secondary URL data item. If the secondary URL data item has connection problems, it will flip to the primary URL definition.
- If, during the processing of events, a problem is detected sending a marker to the marker Web service, the operation that was in process will not complete. The markerChangeProvider Impact policy includes logic, containing characters that disable lines of code in the files, that proposes how to mark the successful creation of a marker from an event, which could then be used to recover from a failed event to create a marker. See the comments in the `markerChangeProvider.ipl` Netcool/Impact policy in the `$TBSM_HOME/contrib/markerimpactprovider` for guidance on how to show when a marker was successfully created from an event. You can use a similar method to show when an event failed to create a marker.

  Note: You must be familiar with Netcool/Impact policies before you attempt to use the method described in this file.

Importing and executing the markerRegisterProviders policy

Follow the same steps as described in the previous section to import markerRegisterProviders.ipl policy into Impact.

This policy creates the provider class and provider instance definitions in the target Marker Repository Service. If the defaults are not acceptable, or if you would like additional definitions, edit the file to do so. A marker overlay category is also created for change events.

- Once imported, edit the policy using the Impact Policy Editor and change as necessary to add in new provider definitions or category definitions. If you change the defaults, be sure to change the sample policies to use the correct provider names.
- Use the Impact user interface option to execute the policy.
- Once the policy completes see the Policy Log for the output of the policy.
- Confirm that the provider and category definitions were inserted into the Marker Repository by using the Marker Command Line Utility GetMarkerProviders and GetCategories options.

This step must be successfully completed before continuing the testing of the connection to the Marker Web Service.

Importing and testing with the sample provider policies

Follow the same steps as described when you are importing the markerSampleSimpleProvider.ipl sample provider policy into Impact. This policy is located in the $TBSM_HOME/contrib/markerimpactprovider directory.

- Once imported, edit the policy using the Impact Policy Editor.
- Search for the words *Customize Me* in the policy for the section of the policy where attributes of a marker are hard coded into the policy, if you would like to change the marker values hard coded into the policy.
- Use the Impact UI option to execute the policy.
- Once the policy completes, see the Policy Log for the output of the policy.
- Confirm that the marker was inserted into the Marker Repository by using the Marker Client Utility *GetMarkers* function to view the markers in the Marker Repository. The *DeleteMarkers* function can be used to remove the sample marker from the repository.

Importing the markerChangeProvider.ipl policy

This policy provides an implementation and an outline of a technique that can be used to capture events that detail configuration changes in the environment and make them available as markers.

The events that this policy works with are Tivoli Application Dependency Discovery Manager change events emitted to OMNIbus through the Tivoli Application Dependency Discovery Manager change event OPAL offering. The structure of the policy can be extended with If conditions to allow this same policy to work with other types of events.

Follow the same steps as described for other policies discussed in this section to import the markerChangeProvider.ipl sample provider policy into Impact.

Note: This procedure apply where you want to install the markerChangeProvider.ipl policy to a non-TBSM Impact server. The TBSM Data server already has a version of this policy installed by default and therefore this procedure is not necessary. If you want to install this policy onto a non-TBSM Impact server, the file is located in the $TBSM_HOME/contrib/markerimpactprovider directory.

- Once imported, edit the policy using the Impact Policy Editor.
- Familiarize yourself with the structure of the policy.
- Identify a means for getting the proper event structure to be provided on input to this policy.

1. Install and configure the Tivoli Application Dependency Discovery Manager Change Event OPAL offering.

2. Create some call with parameters
3. Create an Impact data source to mimic the event as if it came from Omnibus
   - AlertGroup value: `TADDM`
   - AlertKey value: `TADDM:123123123123123`
   - Summary value: `Something changed on VeryImportantHost`

- Search for the words *Customize Me* in the policy for the section of the policy where attributes of a marker are hard coded into the policy.
- If you would like to change the marker values hard coded into the policy, do so.
- Depending on the methodology chosen to provide input to the policy, generate the scenario that will cause the policy to be executed. For example, as is, the provider works properly with an OMNIbus Event Reader. Edit the OMNIbusEventReader Service and create an Event Mapping to filter events to the `markerChangeProvider` policy with an AlertGroup value of `TADDM`.
- Once the policy is executed and completes, see the Policy Log for the output of the policy.
- Confirm that the marker was inserted into the Marker Repository by using the Marker Client Utility *GetMarkers* function to view the markers in the Marker Repository. The *DeleteMarkers* function can be used to remove the sample marker from the repository.

Using an HTTPS connection to the Marker Web service

This section describes how to use an HTTPS connection to the Marker Web service from Impact.

To use an HTTPS connection to the Marker Web service, issue the following commands:

```
cd /opt/IBM/tivoli/impact/sdk/bin/
```

```
./keytool -exportcert -alias default -keystore /opt/IBM/tivoli/impact/wlp/usr/
servers/TBSM/resources/security/key.jks -file cer.cer
```

```
./keytool -import -trustcacerts -file ./cer.cer -keystore /opt/IBM/tivoli/
impact/wlp/usr/servers/TBSM/resources/security/trust.jks -alias "Data server
hostname"
```

```
./keytool -import -trustcacerts -file /tmp/cer.cer -keystore ./cacerts -alias
"Data server hostname"
```

```
./keytool -import -trustcacerts -file ./cer.cer -keystore /opt/IBM/tivoli/
impact/sdk/jre/lib/security/cacerts -alias "Data server hostname"
```

Marker Command Line Client Utility

The Marker Command Line Utility is meant to address administering aspects of the marker data itself, as well as meeting the most basic needs for building a marker provider or client. Uses of it can range from simply making the command line available to users that would like to view or introduce markers into the system periodically to a more sophisticated solution that incorporates the command-line utility into existing automated procedures that may involve information that would be desirable as markers.

The Marker command-line tool is installed on the TBSM Data server in the `TBSM_HOME/bin` directory and can be invoked through the `tbsmmarker.sh` command on a UNIX server or through `tbsmmarker.bat` on a Windows server.

Moving the utility to a remote machine

The Marker Command Line Utility is installed and ready to be used on the TBSM Data server in the `TBSM_HOME/bin` directory. However, in order to provide the flexibility that may be required for administering the Marker Service or for building marker providers, the utility can be moved to another system for easy access.

Complete the following steps to successfully move the command-line utility:

1. Identify the machine that the Marker Command Line Utility will run on and verify prerequisites. The utility requires:
   - Java Version 5.x or higher **to be accessible from the command line (for example, verify by running Java –version from the command line)**
2. On the target system create a directory (for example, `markercli`) that will hold the utility on this system.
3. Copy the appropriate packaged file for the target system from the TBSM Data server system to the target system into the target directory.

   On the TBSM Data server in the `TBSM_HOME/contrib/markercli` directory, the Marker Command Line Utility is packaged in a tar file and a .zip file.
4. Using tar, unzip, or another program capable of extracting files from these file formats, extract the utilities' files from the packaged file.
5. There is no further *installation* process. Installation is complete. Verify that everything is working properly by issuing some of the commands such as:

   tbsmmarker
   This command should return the signatures for the other commands that can be executed by the utility. If this command fails, there is a problem accessing the correct Java environment on the machine.

   tbsmmarker GetServiceInfo –Server *dataserverhostname* -username *tbsmadmin* -password *tbsmadmin_password*
   This command exercises the connection to the Marker Service host and to the Marker Service itself. If it succeeds, it will gather information about the TBSM environment the Marker Service is installed into.

HTTP or HTTPS

By default, the TBSM Data server and the Marker Service use HTTP for communications. However, since that protocol can be changed, it is necessary to import a certificate from the TBSM Data server onto the machine using the command-line utility to enable communications between Marker Command Line Utility and a Marker Service using HTTPS.

1. Acquire/export the certificate from the TBSM Data server by executing the following commands.

   ```
   cd /opt/IBM/tivoli/impact/sdk/bin/
   ```

   ```
   ./keytool -exportcert -alias default -keystore /opt/IBM/tivoli/
   impact/wlp/usr/servers/TBSM/resources/security/key.jks -file cer.cer
   ```

   ```
   ./keytool -import -trustcacerts -file ./cer.cer -keystore /opt/IBM/tivoli/
   impact/wlp/usr/servers/TBSM/resources/security/trust.jks -alias "Data server
   hostname"
   ```

   ```
   ./keytool -import -trustcacerts -file /tmp/cer.cer -keystore ./cacerts -
   alias "Data server hostname"
   ```

   ```
   ./keytool -import -trustcacerts -file ./cer.cer -keystore /opt/IBM/tivoli/
   impact/sdk/jre/lib/security/cacerts -alias "Data server hostname"
   ```
2. Copy the certificate extracted in the first step to the target client command-line utility system.
3. Import the certificate into the target client JRE certificate repository
   a. Start the **ikeyman** utility, which should be part of your JRE. On Windows, it might be called `C:\Program files\IBM\Java60\jre\bin\keyman.exe`.
   b. Open the `cacerts` database file in **ikeyman**. On Windows, it might be called `C:\Program files \IBM\Java60\jre\lib\security\cacerts`. If you use the Browse dialog to navigate to the file, you may have to change the "type of file" to "all files". **ikeyman** will likely prompt you for a password for `cacerts`. Enter the password you use on your client. The default is likely *changeit*.

c. Add the certificate you exported on the Data server and copied to your client. It asks you to provide an alias for the certificate. Use any alias you want, but consider using something meaningful such as `tbsm@myserver`, where *myserver* is the host name of the TBSM Data server.

d. Close **ikeyman** when you are adding the certificate.

4. The Marker Command Line Utility is now ready to communicate over a secure HTTPS connection. Be sure to invoke the marker commands with the `–protocol` option set to HTTPS.
5. Test connectivity by issuing the following command:

```
tbsmmarker GetServiceInfo -S dataserver -protocol https -port 17311
-username tbsmadmin -password tbsmadmin_password
```

Using the Marker Client Utility

This section describes basic information and functions of the Marker Client Utility tool.

Basic information

The command-line utility is front-ended by a batch or shell script named `tbsmmarker`. The basic format of all commands follows this structure.

```
tbsmmarker function <communication parameters> <function parameters><-parmsFile
parmsfilename>
```

where

tbsmarker
the front-end batch (.bat for Windows) or shell script (.sh)

function
instructs the client utility, which the marker Service operates to invoke.

communication parameters
provide Marker Service connection and communication information to the client. These parameters default to the local machine running an HTTP protocol on port `17310`.

-Server
The host name of the system where the Marker Server is installed. Typically, the host name of the TBSM Data server. If not specified, the current host is assumed.

-port
The port used to connect to the Marker Service. Defaults to port `17310`.

-protocol *HTTPS* | *HTTP*
The protocol used to connect to the Marker Service. Defaults to HTTP.

-username
Authentication is required for the script to run. The Dashboard Application Service Hub administrative username created when the TBSM Data server was installed, `tbsmadmin` by default, can be entered for this parameter.

-password
The password that authenticates the user specified.

Note: During the installation TBSM, the Dashboard Application Service Hub administrative user is given authorization to the services running on the TBSM 4.2.1 Data server which return data to the command line interface (CLI). For information about authorizing additional users, see “Authorizing additional users for TBSM Data server services” on page 25.

function parameters
detail the parameters necessary to execute the requested *function*

The following details information about parameters for each function.

parmsFile *parmsfilename*

a mechanism to store and read client command-line parameters from a file in XML format, rather than providing them on the command line each time the utility is invoked. Additional parameters that are required by the function can be specified on the command line with the `parmsFile` parameters. The total set of the file parameters and the specified command-line parameters are considered for processing.

In addition to the advantage of not having to specify common parameters on every function (such as the communications parameters), the file allows you to specify data such as double-byte characters that might not be easy to specify on the command line.

The `parmsFile` format is as follows:

```
<parameters>
<commandlineparameterkeyword>parametervalue</commandlineparameterkeyword>
...

<commandlineparameterkeyword>parameter value</commandlineparameterkeyword>
</parameters>
```

where *commandlineparameterkeyword* is any parameter for a function as defined in this section.

The root element of the XML file must be `<parameters>`, whose child element must have a node name that matches a parameter and the element text is used as the value for the parameter.

For example, rather than providing all *<communication parameters>* on the command line, a file with this content could be specified on the `tbsmmarker` command:

```
<parameters>

<Server>tbsmhostname</Server>

<port>17311</port>

<protocol>https</protocol>

</parameters>
```

The following are some rules for looking at the command-line options:

1. Enclose strings containing special characters (blanks, quotes) in quotes
2. Treat commas as special characters on windows. For instance, when you enter a list of categories separated by commas, be sure to enclose the entire list in quotes.
3. In the command-line descriptions that follow, any parameter enclosed in < > is optional.
4. When options are shown in the list, the first one is the default.
5. No parameter (whether it is specified on the command line or in a parameters file) may appear more than once.

Functions

GetServiceInfo

```
GetServiceInfo <-Server string> <-port integer> <-protocol http|https>
<-username username> <-password password> <-parmsFile filename>
```

Returns information about the Marker Service.

GetCategories

```
GetCategories <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password>
<-parmsFile filename>  <-name string> <-label string>
<-description string>
```

Returns a list of Categories that have been defined as metadata to the Marker Service. Typically, this list details Categories that are of the Overlay type:

-label
A SQL-like expression used to query categories by their display label. Defaults to % - match all labels.

-name
A SQL-like expression used to query categories by their name. Defaults to % - match all names.

-description
A SQL-like expression used to query categories by their description. Defaults to % - match all descriptions.

The –label, -name and –description parameters (if specified) are logically 'anded' together and form the query to return specific category meta information.

RegisterMarkerProviderClass

```
RegisterMarkerProviderClass <-Server string> <-port integer>
<-protocol http|https> <-username username> <-password password>
<-parmsFile filename> -id string -name string <-description string>
```

Registers provider class information with the Marker Service. Provider classes and instances must be registered before a provider can insert markers into the Marker Repository.

-id
A string that uniquely identifies this provider class. This parameter is required.

-name
A string that identifies the provider class. This parameter is required.

-description
A text description of the provider class.

GetMarkerProviderClasses

```
GetMarkerProviderClasses <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password> <-parmsFile
filename> <-IdPattern string> <-DescriptionPattern string>
```

Returns a list of Marker Provider Classes that have been defined to the Marker Service.

-IdPattern
A SQL-like expression used to query provider classes by their name. Defaults to % - matches all.

-DescriptionPattern
A SQL-like expression used to query provider classes by their description. Defaults to % - match all descriptions.

RegisterMarkerProvider

```
RegisterMarkerProvider <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password> <-parmsFile
filename>  -name string -classID string <-uri string> <-description
string> <-className string> <-categories string<,string,...>>
```

Registers a provider to the Marker Service. Provider classes and instances must be registered before a provider can insert markers into the Marker Repository.

–name
A unique string that identifies this provider instance.

–uri
A URL that can be used to launch back to this provider.

–description
A string description of the provider.

–classID
References the –id parameter of the RegisterMarkerProviderClass to identify the provider class of this provider.

–className
References the –name parameter of the RegisterMarkerProviderClass to identify the provider class of this provider.

–categories
A list of strings representing categories to be automatically added to any marker that this provider inserts into the Marker Repository. A list of categories can be provided by delimiting each category value by a comma.

GetMarkerProviders

```
GetMarkerProviders <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password> <-parmsFile
filename> <-IdPattern string> <-DescriptionPattern string>
<-CategoryIDs string<,string, …string>>
<-ProviderClassIDs string<,string, . . . string>>
```

Returns a list of Marker Provider instances that have been defined to the Marker Service.

–IdPattern
A SQL-like expression used to query providers by their name. Defaults to % - matches all.

ProviderClassIDs
Specifies an optional list of provider class ID or IDs of the wanted marker provider or providers.

–DescriptionPattern
A SQL-like expression used to query providers by their description. Defaults to % - match all descriptions.

CategoryIDs
Specifies an optional list of category ID(s) that the returned marker providers must support.

CreateCategory

```
CreateCategory <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password>
<-parmsFile filename> -id string -label string
<-description string>
```

Defines meta information for a type of category that is designated to be an overlay. Categories that are defined as overlays are used to organize the viewing of markers by type.

Note: The Time Window Analyzer user interface uses categories of this type as the list of overlays to view within the portlet.

-id
A string used to uniquely identify the category.

-label
A displayable string used to identify this category type in a user interface.

-description
A description of the category, and therefore, of the markers that are associated with this category.

InsertMarker

```
InsertMarker <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password>
<-parmsFile filename> -providerClass string -providerID
string -summary string -detail string -categories
string<,string,...> <-time yyyy-MM-dd'T'HH:mm:ss[z]>
<-resourceID string> <-key string>
```

–summary
a brief string description of the marker. This field is used to capture an abbreviated version of the details of that marker. Up to 255 characters are allowed; however, the Time Window Analyzer portlet uses this field for user interface flyover purposes, so limiting the summary to 50-75 characters is desirable.

The summary can include simple HTML tags to enhance formatting.

–detail
a verbose string capturing all important information about the marker.

The details can include simple HTML tags to enhance formatting.

–time *yyyy-MM-dd'T'HH:mm:ss[z]*
The time associated with the marker whose purpose it is to capture the time that the described marker took place or will take place. The time must be specified in the format of `yyyy-mm-ddT HH:mm:ss timezone`. If the timezone is not provided, the timezone of the system where the command was issued.

-providerID
References the `–id` parameter of the RegisterMarkerProvider to identify the provider of this marker.

-providerClass
References the `–name` parameter of the RegisterMarkerProviderClass to identify the provider class of this provider.

-categories (string, string .., string)
A set of strings called ‘categories' can be associated with a marker. Categories provide context to the marker that enhances a client's ability to correlate the significance of the event the marker details with other markers and with other relevant pieces of data including resources and attributes of a resource. Categories ‘categorize' markers into one or more like groups, enabling easy searching. At the minimum, a marker should include a category describing the type of marker and the resource identifier for the primary resource described in the marker. For example, all markers for a particular resource should have that resource name as a category.

-resourceID
A special type of category that identifies the resource for the marker. For example, if the marker is telling the status of a host called ‘host1', then the resourceId should identify host1. See the section on category conventions for the proper way of identifying a resource identifier.

GetMarkers

```
GetMarkers <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password>
<-parmsFile filename> <-allCategories string<,string,...>>
<-anyCategories string<,string,...>> <-providerClasses
string<,string,...>> <-providerIDs string<,string,...>>
<-oldest yyyy-MM-dd'T'HH:mm:ss[z]> <-newest
yyyy-MM-dd'T'HH:mm:ss[z]>
<-maxrows integer> <-outputFilename string>
```

Queries and returns markers that are in the Marker Repository.

–outputFilename
The name of an output file where the output of the GetMarkers query will be placed in XML format.

–maxrows *(integer)*
The maximum number of markers to be returned by this GetMarkers query. The default is `1000`.

-oldest *yyyy-MM-dd'T'HH:mm:ss[z]*
A time representing the starting time range for the query. If not specified, the ‘oldest' is set to 24 hours before the ‘newest' time. Time is expressed in a 24-hour clock, and the standard timezone abbreviations can be used: EST, PST, EDT, GMT. If the timezone is left off the string, the local timezone is used.

-newest *yyyy-MM-dd'T'HH:mm:ss[z]*
A time representing the ending time range for the query. If not specified, the current time is used. Time is expressed in a 24-hour clock and the standard timezone abbreviations can be used: EST, PST, EDT, GMT. If the timezone is left off the string, the local timezone is used.

–providerIDs (string, string .., string)
A list of provider IDs in string format, delimited by a comma. Each string value should reference the same value specified on the `–id` parameter of the RegisterMarkerProvider to enable filtering of markers by provider class.

–providerClasses (string, string .., string)
A list of provider classes in string format, delimited by a comma. Each string value should reference the same value specified on the –name parameter of the RegisterMarkerProviderClass to enable filtering of markers by provider instance.

–allCategories (string, string .., string)
A list of categories in string format, delimited by a comma. Each string value represents a category value where all the specified categories must be on a given marker returned by the query. An 'AND' filter is created by this parameter.

–anyCategories (string, string .., string)
A list of categories in string format, delimited by a comma. Each string value represents a category value where at least one of the specified categories must be on a given marker returned by the query. An 'OR' filter is created by this parameter.

DeleteMarkers

```
DeleteMarkers <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password>
<-parmsFile filename> -markerIDs long<,long,...>
```

markerIDs
The list of one or more markers (as identified by their markerid) to act upon. A list is specified by provided the markerids separated by a comma with no spaces included. Optionally, this parameter can specify a file name that contains the output of a GetMarkers command request created by using the –outputFileName parameter.

DeleteCategory

```
DeleteCategory <-Server string> <-port integer> <-protocol http|https>
<-username username> <-password password> <-parmsFile string> -id string
```

id
Specifies the category to delete.

PruneProviderData

```
PruneProviderData <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password>
<-parmsFile string> -dropDate yyyy-MM-dd'T'HH:mm:ss[z]
```

dropDate

Specifies the end date (inclusive) through which unused data will be deleted:

- provider instances that do not have any marker data and are older than the drop date
- provider classes that have no provider instances and are older than the drop date
- unused category/provider mappings. The category/marker mappings get pruned when markers get pruned.
- any auto-created categories that are not referenced by markers or metadata.

Time is expressed in a 24-hour clock and the standard timezone abbreviations can be used: EST, PST, EDT, GMT. If the timezone is left off the string, the local timezone is used.

PruneMarkers

```
PruneMarkers <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password>
<-parmsFile string> -dropDate yyyy-MM-dd'T'HH:mm:ss[z]
```

dropDate
Specifies the end date (inclusive) through which markers will be deleted. Time is expressed in a 24-hour clock and the standard timezone abbreviations can be used: EST, PST, EDT, GMT. If the timezone is left off the string, the local timezone is used.

AddCategoryToMarker

```
AddCategoryToMarker <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password>
<-parmsFile string> - -categoryID string<,string,...>
-markerID long<,long,...>
```

Adds categories to existing markers.

categoryID
Specifies one or more category IDs to add to the specified `marker(s)`

markerID
The list of one or more markers (as identified by their markerid) to act upon. A list is specified by provided the markerids separated by a comma with no spaces included. Optionally, this parameter can specify a file name that contains the output of a GetMarkers command request created by using the `–outputFileName` parameter.

RemoveCategoryFromMarker

```
RemoveCategoryFromMarker <-Server string> <-port integer> <-protocol
http|https> <-username username> <-password password> <-parmsFile
string> - -categoryID string<,string,...> -markerID long<,long,...>
```

Removes categories from existing markers.

categoryID
Specifies one or more category IDs to remove from the specified marker or markers.

markerID
The list of one or more markers (as identified by their markerid) to act upon. A list is specified by provided the markerids separated by a comma with no spaces included. Optionally, this parameter can specify a file name that contains the output of a GetMarkers command request created by using the `–outputFileName` parameter.

Important issues concerning the marker data store

This section discusses important issues concerning the target data store.

Storage Capacity

Since its purpose is to maintain a historical record of markers, the Marker Service has disk storage requirements that must be met in order for the system to remain stable. This section discusses storage capacity needs for the marker data store. Since this release also maintains a metric repository, be sure to consider its storage capacity needs as well as if the data stores are collocated. Complete the following process for estimating disk space, which is required for maintaining markers:

- Markers = *NM* x *storehistoryperioddays*

  where

  – *NM* is the average number of markers collected per day
  – *storehistoryperioddays* is the property that determines the number of days before a marker is purged from the data store (default: 14).
- *Capacity* = *Markers*/1250

  where

  – *Capacity* is the size in megabytes that should be allocated for storing markers
  – *Markers* is the number of markers maintained before being purged from the data store

The design point for the marker data store, and for markers in general, assumes that the volume of markers will be somewhat limited. The Marker Service is not meant as a data store for any event that occurs. Not only will uncontrolled markers possibly result in poor performance, but it can easily create an unusable and ineffective Time Window Analyzer solution.

Tuning

It is recommended that regular maintenance appropriate to the target database is scheduled. Consulting with the database administrator for the targeted data store is strongly recommended.

Creating a high availability marker data store

High availability of the TBSM Marker Service data store is accomplished by the same DB2 HADR configuration described in the TBSM failover section.

All TBSM Marker Service property files must be kept in sync between the primary and backup TBSM Data servers. The TBSM Marker Service Component does not provide any automatic processes for syncing the file.

Best Practice Suggestions

When using the command-line utility to define marker categories, marker provider class, and marker provider instances, maintain a list of the commands for purposes of easily recreating the setup at another time.

To take advantage of the Time Window Analyzer function, it will be necessary to take the time to identify and create marker providers for the information that will be valuable to the diagnostics process.

Be careful how many markers are created. There needs to be a balance between providing too many markers and providing information that is valuable to the Time Window Analyzer users.

Chapter 11. Reference

Reference information is organized to help you locate particular facts quickly.

TBSM ports

The OMNIbus port default value is 4100. Other port types and numbers used by the TBSM Data and Dashboard servers are listed in this topic.

Ports used by the Data server

Use the information about port names and file locations to change TBSM Data server port numbers.

Communication port

The Data server uses port number 17542 for RMI communication. The following TBSM files contain this port number:

```
/opt/IBM/tivoli/impact/etc/TBSM_server.props
$TBSM_HOME/bin/fo_config.ant
```

DB2 port

By default. the Data server uses port number 50000 for connecting to DB2 databases. The following TBSM files contain this port number:

```
C:\Program Files\IBM\tivoli\impact\etc\TBSM_TBSMDatabase.ds
C:\Program Files\IBM\tivoli\impact\etc\TBSM_TBSMMarker.ds
C:\Program Files\IBM\tivoli\impact\etc\TBSM_TBSMMetricHistory.ds
C:\Program Files\IBM\tivoli\impact\etc\TBSM_TBSMComponentRegistry.ds
C:\Program Files\IBM\tivoli\impact\etc\TBSM_EventrulesDB.ds
C:\Program Files\IBM\tivoli\impact\etc\TBSM_SCR.ds
```

WebSphere ports

Table 151 on page 259 lists the port names and numbers, and, where applicable, the files in which these port numbers occur.

Table 151. WebSphere ports used by the Data server

Port name	Port number	File location
CSIV2_SSL_MUTUALAUTH_ LISTENER_ADDRESS	17322	
DCS_UNICAST_ADDRESS	17318	
ORB_LISTENER_ADDRESS	17320	
SAS_SSL_SERVERAUTH_ LISTENER_ADDRESS	17321	
SOAP_CONNECTOR_ADDRESS	17313	
WC_defaulthost_secure	17311	

Table 151. WebSphere ports used by the Data server (continued)		
Port name	**Port number**	**File location**
WC_adminhost	17315	
WC_defaulthost **Note:** The Time Window Analyzer marker service uses this port.	17310	`/opt/IBM/tivoli/impact/wlp/usr/servers/TBSM/apps/ nameserver.war/WEB-INF/web.xml` *`$TBSM_HOME`*`/bin/fo_config.ant /opt/IBM/tivoli/impact/etc/TBSM.props /opt/IBM/tivoli/impact/etc/TBSM_server.props /opt/IBM/JazzSM/profile/RAD_registry /opt/IBM/tivoli/impact/etc/nameserver.props`
WC_adminhost_secure	17316	

ObjectServer ports

Table 152 on page 260 lists the port names, numbers, and the files in which these port numbers occur.

Table 152. ObjectServer ports used by the Data server		
Server	**Port number**	**File location**
ObjectServer	4100	`/opt/IBM/tivoli/impact/etc/TBSM_ObjectServer_DS.ds /opt/IBM/tivoli/impact/etc/TBSM_OutputObjectServer_DS.ds` *`$TBSM_HOME`*`/bin/fo_config.ant` *`$TBSM_HOME`*`/bin/ncosUtil.bat`
Backup ObjectServer for failover	4100	`/opt/IBM/tivoli/impact/etc/TBSM_ObjectServer_DS.ds /opt/IBM/tivoli/impact/etc/TBSM_OutputObjectServer_DS.ds`

Derby Ports

Table 153 on page 260 lists the port names, numbers, and the files in which these port numbers occur.

Table 153. Derby ports used by the Data server		
Port name	**Port number**	**File location**

Default connection port	1527	primary data server:
		`$TBSM_HOME/bin/fo_config.ant`
		`$TBSM_HOME/etc/TBSM_ImpactDB.ds`
		backup data server:
		`$TBSM_HOME/bin/fo_config.ant`
		`$TBSM_HOME/etc/TBSM_B_ImpactDB.ds`

Table 153. Derby ports used by the Data server (continued)		
Port name	**Port number**	**File location**

Replication port	4851	primary data server:
		`$TBSM_HOME/bin/fo_config.ant`
		`$TBSM_HOME/etc/TBSM_impactdatabase.props`
		backup data server:
		`$TBSM_HOME/bin/fo_config.ant`
		`$TBSM_HOME/etc/TBSM_B_impactdatabase.props`

Ports used by the Dashboard server

Use the information about port names and file locations to change TBSM Dashboard server port numbers.

Communication ports

Table 154 on page 261 lists the port names and numbers, and, where applicable, the files in which these port numbers occur.

Table 154. Communication ports used by the Dashboard server		
Server	**Port number**	**File location**
Dashboard server	17543	`/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_server.props.`
Data server; also backup server for failover	17542	`/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_server.props.`

HTTP ports

Table 155 on page 261 lists the port names, numbers, and the files in which these port numbers occur.

Table 155. HTTP ports used by the Dashboard server		
Server	**Port number**	**File location**
Data server	17310	`/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_sla.props` `/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/nameserver.props`
Backup Data server for failover	17310	`/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_sla.props` `/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/nameserver.props`

WebSphere ports

Table 156 on page 262 lists the port names and numbers, and, where applicable, the files in which these port numbers occur.

Table 156. WebSphere ports used by the Dashboard server

Port name	Port number	Location
BOOTSTRAP_ADDRESS	16312	/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_sla.props
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	16322	
DCS_UNICAST_ADDRESS	16318	
ORB_LISTENER_ADDRESS	16320	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	16321	
SOAP_CONNECTOR_ADDRESS	16313	
WC_defaulthost_secure	16311	
WC_adminhost	16315	
WC_defaulthost	16310	/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_sla.props
WC_adminhost_secure	16316	

IBM Tivoli Business Service Manager utilities

Use the TBSM utilities to perform such operations as sending test events, creating a maintenance schedule, and deleting a maintenance schedule.

rad_discover_schema command - Discover schema

Use the **rad_discover_schema** command to ensure that the TBSM Data server is using the most recent changes to the ObjectServer schema.

Syntax

```
rad_discover_schema DASH_username DASH_password typename
```

Parameters

DASH_user
Dashboard Application Service Hub user name.

DASH_password
The password for the specified user name.

typename
The name of the Impact type that you want to discover.

Example

This examples updates the Data server with any schema changes for the Object Server used by TBSM for receiving events:

```
[tbsm61@nc9037038100 bin]$ ./rad_discover_schema tbsmadmin tbsmadmin_user_password
ObjectServer
```

rad_maint_add command - Add schedule

Use the **rad_maint_add** command to create a maintenance schedule and add it to a service or group of related services.

Syntax

```
rad_maint_add -o service -b YYYMMDDHH -e YYYMMDDHH -s 0|1|2 -n schedule_name
```

Parameters

This **rad_maint_add** command has the following parameters:

-o *service*
Specifies the name of the service. If the string contains any special character, enclose the string in quotation marks.

-b *start_time*
Specifies the starting time of the schedule. The *start_time* variable is a string in the YYYMMDDHH format. If the string contains a special character, enclose the string in quotation marks. By default, the starting time is set to the current time.

-e *end_time*
Specifies the ending time of the schedule. The *end_time* variable is a string in the YYYMMDDHH format. If the string contains a special character, enclose the string in quotation marks. By default, the ending time is set to one of the following values:

- (If a starting time is specified) 10 years after the starting time
- (If a starting time is not specified) 10 years after the command is run

-s *schedule_scope*
Specifies the scope of the schedule:

0
The schedule is added to the specified service.

1
The schedule is added to the specified service and all child services.

2
The schedule is added to the child services only.

-n *schedule_name*
Specifies the name of the schedule. The *schedule_name* variable can be either the name of an existing schedule or a new name. By default, schedules are assigned names in the following format: *starting_time - ending_time*, where both variables are formatted as YYYY/MM/DD HH:MM.

The following example shows how to add a schedule to the Equities service. The schedule will end on December 31, 2011 at 23:59. As the starting time is not specified, the schedule starts when the **rad_maint_add** command is run.

```
rad_maint_add -o Equities -e 201112312359
```

rad_maint_remove command - Remove schedule

Use the **rad_maint_remove** command to remove a maintenance schedule from a service or group of related services.

Syntax

```
rad_maint_remove -o service -s 0|1|2
```

Parameters

This function has the following parameters:

-o ***service***
Specifies the service from which you want to remove the schedule. If the string contains any special character, enclose the string in quotation marks.

-s
Specifies the scope of the schedule removal:

0
The schedule is removed from the specified service.

1
The schedule is removed from the specified service and all child services.

2
The schedule is removed from the child services only.

Example

The following example shows how to remove a schedule from the Equities service and all of its child services:

```
rad_maint_remove -o Equities -s 1
```

rad_sendevent Send test events

Use the **rad_sendevent** command to send test events to the ObjectServer.

Usage

After you configure a service model, send test events to ensure that the services in the model respond correctly to ObjectServer events. The **rad_sendevent** utility is located in the$TBSM_HOME/bin directory and can be run from a UNIX command prompt. The command **rad_sendevent.bat** is also available on Windows systems.

After you issue a **rad_sendevent** command, the command will accept field name and value pairs. Each item must be separated by a carriage return; two sequential carriage returns marks the end of the test event. To end a **rad_sendevent** session, hold **Ctrl** and press **C**.

Note: After holding **Ctrl** and pressing **C** to end the **rad_sendevent** session, you might see a `java.io.IOException` error message. This is normal, and can be safely ignored.

You also can send test events using either the IBM Tivoli Netcool/OMNIbusSQL interactive interface utility or the TBSM console. The TBSM console send event option does not work for all configurations, such as services with multiple incoming-status rules.

Syntax

```
rad_sendevent object_server port user_id password
```

Parameters

object_server
The host name or IP address of the system where the ObjectServer component is installed.

port
The port number on which the ObjectServer is listening.

user_id
A user ID with authority to access the ObjectServer.

password
The password associated with the user ID. Use two double quotation marks to specify a null password.

Example: Sending test events

The following command starts the **rad_sendevent** utility:

```
rad_sendevent <objectserverhost> <objectserverPort> username password
```

Typing the following entries sends an event with values for `Summary`, `AlertGroup`, `Location`, `Severity`, and `Identifier` fields:

```
Summary
Ticket Bar Chart Dummy Event
AlertGroup
TicketsByLocation
Location
US
Severity
1
Identifier
USTickets1A
```

Hold **Ctrl** and press **C** to end the **rad_sendevent** session.

Note: The value used for <objectserverhost> cannot be `localhost`. It must be an IP address or a host name.

Each event must have a unique value in the `Identifier` field. If you send an event that changes the status of a service, you must note the value of the `Identifier` field for that event. If you later change the status of the same service, you must change the status of the exact event that caused the original status change. Otherwise, TBSM and the ObjectServer see two separate events rather than an update to an existing event. As a result, the status of the service does not change as expected.

Also, if you are sending the event with the same `Identifier` multiple times, change the severity to clear and delete the event before you send it again. For example, if you are testing how different values affect a TBSM service, clear the event, and delete it before you send the event again. For information about manipulating an event list, see the IBM TBSM *Service Configuration Guide*, IBM TBSM *Scenarios Guide* or the IBM Tivoli Netcool/OMNIbus documentation.

Example: Sending test events using a file

You also can send the contents of a file to the **rad_sendevent** command. Create a file using the same syntax as you would with the prompt. On a UNIX system, issue the following command:

```
cat file_name | $TBSM_HOME/bin/rad_sendevent object_server port user_id password
```

where *file_name* is the fully qualified name of the file that contains the test-event information

`rad_crypt` Encrypt password

Use the **`rad_crypt`** script to create the encrypted password for TBSM Data Server users.

Usage

The `rad_crypt` script is used to create the encrypted password for the TBSM Data Server users. This script is in the following location:

`$TBSM_HOME/bin`

Syntax

Windows

```
rad_cryp.bat password
```

UNIX

```
./rad_crypt password
```

Parameters

password
The password being encrypted.

`rad_compilewsdl` Create WSDL JAR file

Use the **`rad_compilewsdl`** script to create a JAR file that contains a programmatic representation of WSDL data.

Usage

The `rad_compilewsdl` script is used to create a JAR file that contains a programmatic representation of the WSDL data. This script executes the Impact `nci_compilewsdl` script internally. For more information about the the nci_compilewsdl script, see: https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.13/com.ibm.netcoolimpact.doc/dsa/imdsa_web_running_the_wsdl_compiler_script_unix_t.html.

The `rad_compilewsdl` script is in the following location:

`$TBSM_HOME/bin`

Syntax

Windows

```
rad_compilewsdl.bat packageName wsdlFileUrl destination
```

UNIX

```
./rad_compilewsdl packageName wsdlFileUrl destination
```

Parameters

packageName
The name of the package containing WSDL data.

wsdlFileUrl
The name of the JAR file (without the `.jar` suffix) to be created by the script.

destination
The directory to copy the generated JAR file to, the default directory is `$NCHOME/wslib`.

Important configuration files

This section describes key configuration files that can be used with TBSM. It does not include all configuration files. Each topic contains information about the file properties.

`enqueuecl.properties` file

The properties set in the `enqueuecl.properties` file determine levels of message and trace logging for the Discovery Library Toolkit.

Name	Default value	Description
Message Log		
MessageLoggingLevel	Error	Logging level used for the message log: • Info – provides the most information. • Warning – provides additional information, but less than Informational. • Error – provides the least amount of information. Exceptions are shown, along with vital informational messages.
MessageLogMaximumFiles	3	The maximum number of message log files. This property indicates how many message log files are created before the logging service starts reusing files.
MessageLogMaximumFileSize	10240	The maximum size of a message log file in KB. This property indicates how large message log files can get before a new message log file is created, or the oldest message log file is reused.
Trace log		
TraceLoggingState	Enabled	The initial trace logging state. Valid values are: • Enabled • Disabled
TraceLoggingLevel	Fine	The initial logging level. Valid values are: • Finest – provides the most information. Might be required for debugging problems. • Finer – moderate amount of information. • Fine – provides the minimal amount of trace information.
TraceLogMaximumFiles	3	The maximum number of trace log files. This property indicates how many trace log files are created before the logging service starts reusing files.
TraceLogMaximumFileSize	10240	The maximum size of a trace log file in KB. This property indicates how large trace log files can get before a new trace log file is created or the oldest trace log file is reused.

fo_config.props file

You create the fo_config.props file using the fo_config script. If you used the default values when you installed TBSM, you do not have to modify most of the parameters in this file.

Parameters in the fo_config.props file

The following table lists the parameters that are set in the fo_config.props file.

Parameter	Default value	Description
DATA_PrimaryDataServerHostname	None	Host name of the primary Data server
DATA_PrimaryDataServerHTTPPort	17310	HTTP port of the primary Data server. If you changed the default value during installation, use the value of **WC_defaulthost** in /opt/IBM/tivoli/impact/etc/TBSM_server.props.
DATA_PrimaryDB2Hostname	None	This is the hostname of the primary DB2 server (or hostname of lone DB2 server if DB2 is not in failover). This value can be found in TBSM_TBSMDatabase.ds in the installdirectory/tbsm/etc/ directory. Find the the property: TBSMDatabase.DB2.PRIMARYHOST
DATA_PrimaryDB2Port	50000	DB2 Port of primary database. This value can be found in TBSM_TBSMDatabase.ds in the installdirectory/tbsm/etc/ directory. Find the the property: TBSMDatabase.DB2.PRIMARYPORT
ConfigureNameServer	true	By default the fo_config script will configure the nameserver configuration with the host/port of the primary server and backup server (if failover is configured). If you have already customized the nameserver configuration to include an additional Impact cluster ASWELL as the TBSM failover pair, you should set this property to false. Otherwise, fo_config overwrites your customizations. Additionally, you can run the nci_configuration_utility script to reconfigure the nameservers after you run fo_config.
DATA_PrimaryConfigureFailback	false	Failback is the behavior whereby when the default backup server is running as primary and the default primary comes back up, the default primary takes over as the primary TBSM server and the default backup transitions into the backup role. To enable this behavior set the following property to true. Otherwise, the default primary will assume the backup role if it starts up when the default backup server is running as primary.

Parameter	Default value	Description
DATA_BackupMillisecondsTill StartAsStandalone	300000	If there is a backup server and it is started before the primary server is started, by default it will wait 300 seconds until assuming the role as the primary. This duration can be configured with this property.
DATA_PrimaryObjectServerHostnam e	None	Host name of the primary ObjectServer.
DATA_PrimaryObjectServerPort	4100	Port of the primary ObjectServer. If you changed the default value during installation, use the value of **ObjectServer_DS.ObjectServer.PRIMARYPORT** in `/opt/IBM/tivoli/impact/etc/TBSM_ObjectServer_DS.ds`.
DATA_PrimaryObjectServerName	NCOMS	Name of the primary ObjectServer.
DATA_BackupDataServerHostname	None	Host name of the backup Data server. This parameter is optional. If this parameter is not specified, the system is configured as a standalone Data server, whether it was previously configured as part of a failover pair.
DATA_BackupDataServerHTTPPort	17310	HTTP port of the backup Data server. For details about changing the Data server port, see https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.12/com.ibm.netcoolimpact.doc/admin/imag_changing_the_http_port.html
DATA_BackupDB2Hostname	none	Only configure this if DB2 is in replication (HADR is configured for DB2). **Note:** When failover is configured for TBSM data servers, it does not mean DB2 must also be in failover. Also, it is possible to have DB2 failover setup without TBSM failover. In this case, TBSM can be configured for DB2 replication by using `fo_config` template and not setting a backup data server host, but setting a backup DB2 host.
DATA_BackupDB2Port	50000	DB2 port of backup DB2 server. This is optional. Only configure this property if DB2 is in failover (HADR).
DATA_BackupObjectServerHostnam e	None	Host name of the backup ObjectServer. This parameter is optional. If this parameter is the not specified, the Data server is configured to communicate with a standalone ObjectServer.
DATA_BackupObjectServerPort	4100	Port of the backup ObjectServer. This value is typically the same value as the value of **DATA_PrimaryObjectServerPort**.

Parameter	Default value	Description
DATA_BackupObjectServerName	NCOMS_BKUP	Name of the backup ObjectServer. This value must be different from the value of **DATA_PrimaryObjectServerName**.
UseObjectServerFailback	false	If the ObjectServer is configured for failback then set this property to true. This enables TBSM (embedded Impact) to connect to the primary ObjectServer if the primary ObjectServer comes back up, instead of continuing to use the backup ObjectServer.
DATA_Bi-DirectionalGatewayName	NCO_GATE	Name of the bi-directional gateway
DATA_Bi-DirectionalGatewayPort	4300	Port of the bi-directional gateway
DASH_PrimaryDataServerHostname	None	Host name of the primary Data server.
DASH_PrimaryDataServerRMIPort	17542	RMI port of the primary Data server.
DASH_PrimaryDataServerHTTPPort	17310	HTTP port of the primary Data server. This value must be the same as the value of **DATA_PrimaryDataServerHTTPPort**.
DASH_BackupDataServerHostname	None	Host name of the backup Data server. This parameter is optional. If this parameter is not specified, the Dashboard server is configured to communicate with a standalone Data server.
DASH_BackupDataServerRMIPort	17542	RMI port of the backup Data server.
DASH_BackupDataServerHTTPPort	17310	HTTP port of the backup Data server. This value must be the same as the value for **DATA_BackupDataServerHTTPPort**.

Example of the `fo_config.props` file

```
#===============================================================================
#  TBSM 6.1 Failover Configuration Properties file
#    Template generated at 20110610074608
#===============================================================================

# Invoke the script with template target to generate a configuration template
# in the local directory:
#
#     Unix:    $TBSM_HOME/bin/fo_config template
#     Windows: %TBSM_HOME%\bin\fo_config template
#
# Invoke the script with dashboardserver target and specifying
#    the configuration file to configure Dashboard Server:
#
#     Unix:    $TBSM_HOME/bin/fo_config dashboardserver
 -Dfo_config.props=[config-file-path]
#     Windows: %TBSM_HOME%\bin\fo_config dashboardserver
 -Dfo_config.props=[config-file-path]
#
# Invoke the script with dataserver target and specifying
#    the configuration file to configure Data Server:

#
#     Unix:    $TBSM_HOME/bin/fo_config dataserver
 -Dfo_config.props=[config-file-path]
#     Windows: %TBSM_HOME%\bin\fo_config dataserver
 -Dfo_config.props=[config-file-path]
#
```

```
#-------------------------------------------------------------------------
# Parameters used in Data Server configuration
#-------------------------------------------------------------------------
# Primary Data Server Hostname
#   This is a required field. Script will not run if leave blank.
#   It is set in the following locations:
#   1. The key REPLICANT.0.HOST in the
#      /opt/IBM/tivoli/impact/wlp/usr/servers/TBSM/apps/nameserver.war
/WEB-INF/web.xml
#      file on both the primary and the backup data server.
#   2. The keys impact.nameserver.0.host in the
#      /opt/IBM/tivoli/impact/etc/nameserver.props
#      file on both the primary and the backup data server.
#
DATA_PrimaryDataServerHostname=tbsmprimary.mycompany.com
#-----------------------------------------------------

# Primary Data Server HTTP Port
#   The default value is 17310.
#   Current value can be found in the following locations:
#   1. The key REPLICANT.0.PORT in the
#      /opt/IBM/tivoli/impact/wlp/usr/servers/TBSM/apps/nameserver.war
/WEB-INF/web.xml
#      file on both the primary and the backup data server.
#   2. The key impact.registry.0.port in the
#      $IMPACT_HOME/etc/nameserver.props/
#      file on both the primary and the backup data server.
#
DATA_PrimaryDataServerHTTPPort=17310
#-----------------------------------------------------

# Primary DB2 Host
#   Current value can be found in the following locations:
#   1. The key TBSMDatabase.DB2.PRIMARYHOST in the
#      /opt/IBM/tivoli/impact/etc/TBSM_TBSMDatabase.DB2.ds
#      file on the primary data server.
#   2. The key TBSMDatabase.DB2.PRIMARYHOST in the
#      /opt/IBM/tivoli/impact/etc/TBSM_B_TBSMDatabase.DB2.ds
#      file on the backup data server.
#
DATA_PrimaryDB2Hostname=db2primary.mycompany.com
#-----------------------------------------------------

# Primary DB2 Port
#   The default value is 50000.
#   Current value can be found in the following locations:
#   1. The key TBSMDatabase.DB2.PRIMARYPORT in the
#      /opt/IBM/tivoli/impact/etc/TBSM_TBSMDatabase.ds
#      file on the primary data server.
#   2. The key TBSMDatabase.DB2.PRIMARYPORT in the
#      /opt/IBM/tivoli/impact/etc/TBSM_B_TBSMDatabase.DB2.ds
#      file on the backup data server.
#
DATA_PrimaryDB2Port=50000
#-----------------------------------------------------



# Primary ObjectServer Hostname
#   This is a required field. Script will not run if leave blank.
#   It is set in the following locations:
#      The key ObjectServer_DS.ObjectServer.PRIMARYHOST in the
#      /opt/IBM/tivoli/impact/etc/TBSM_ObjectServer_DS.ds
#      file on both the primary and the backup data server.
#
DATA_PrimaryObjectServerHostname=tbsmprimary.mycompany.com
#-----------------------------------------------------

# Primary ObjectServer Port
#   The default value is 4100.
#   Current value can be found in the key
ObjectServer_DS.ObjectServer.PRIMARYPORT in the
#      /opt/IBM/tivoli/impact/etc/TBSM_ObjectServer_DS.ds
#
DATA_PrimaryObjectServerPort=4100
#-----------------------------------------------------

# Primary ObjectServer Name
#   The default value is NCOMS.
#   Current value can be found in the following locations:
#   1. The OMNIbus server interface file
#   2. The key Gate.ObjectServerA.Server in the
```

```
#      $OMNIHOME/gates/NCO_GATE/NCO_GATE.props
#      file on the backup ObjectServer (which will exist after
#      the gateway is configured)
#
DATA_PrimaryObjectServerName=NCOMS
#----------------------------------------------------

# By default the fo_config script will configure the nameserver
# configuration with the host/port of the primary server
# and backup server (if failover is configured).
# If you have already customized the nameserver configuration to include
# an additional Impact cluster ASWELL as the TBSM failover pair
# you should set the following property to false
# so that fo_config will not overwrite your customizations.
# Additionally, you can run the nci_configuration_utility script
# to reconfigure the nameservers after you run fo_config.
ConfigureNameServer=true

# Failback is the behavior whereby when the default backup server is running
# as primary and the default primary comes back up, the default
# primary takes over as the primary TBSM server and the default
# backup transitions into the backup role. To enable this behavior
# set the following property to true. Otherwise, when the default primary
# will assume the backup role if it starts up when the default backup
# server is running as primary.
DATA_PrimaryConfigureFailback=false

# If there is a backup server and it is started before the primary
# server is started, by default it will wait 300 seconds until assuming
# the role as the primary. This duration can be configured with
# the following property.
DATA_BackupMillisecondsTillStartAsStandalone=300000

# Backup Data Server Hostname
#   This is an optional field.  Server will be configured as a standalone
 if leave blank.
#   It is set in the following locations:
#   1. The key REPLICANT.1.HOST in the
#      $IMPACT_HOME//wlp/usr/servers/TBSM/apps/nameserver.war/WEB-INF/web.xml
#      file on both the primary and the backup data server.
#   2. The key impact.registry.1.host in the
#      /opt/IBM/tivoli/impact/etc/nameserver.props
#      file on both the primary and the backup data server.
#
DATA_BackupDataServerHostname=tbsmbackup.mycompany.com
#----------------------------------------------------

# Backup Data Server HTTP Port
#   The default value is 17310.
#   Current value can be found in the following locations:
#   1. The key REPLICANT.1.PORT in the
#      /opt/IBM/tivoli/impact/wlp/usr/servers/TBSM/apps/nameserver.war/
WEB-INF/web.xml
#      file on both the primary and the backup data server.
#   2. The key impact.registry.1.port in the
#      /opt/IBM/tivoli/impact/etc/nameserver.props
#      file on both the primary and the backup data server.
#
DATA_BackupDataServerHTTPPort=17310
#----------------------------------------------------


# Backup DB2 Host
# Configure this property if you have DB2 replication.
#   Current value can be found in the following locations:
#   1. The key TBSMDatabase.DB2.BACKUPHOST in the
#      /opt/IBM/tivoli/impact/etc/TBSM_TBSMDatabase.DB2.ds
#      file on the primary data server.
#   2. The key TBSMDatabase.DB2.BACKUPHOST in the
#      /opt/IBM/tivoli/impact/etc/TBSM_B_TBSMDatabase.DB2.ds
#      file on the backup data server.
#
DATA_BackupDB2Hostname=db2backup.mycompany.com
#----------------------------------------------------

# Backup DB2 Port
# Configure this property if you have DB2 replication.
#   The default value is 50000.
#   Current value can be found in the following locations:
#   1. The key TBSMDatabase.DB2.BACKUPPORT in the
#      /opt/IBM/tivoli/impact/etc/TBSM_TBSMDatabase.ds
#      file on the primary data server.
```

```
#   2. The key TBSMDatabase.DB2.BACKUPPORT in the
#      /opt/IBM/tivoli/impact/etc/TBSM_B_TBSMDatabase.DB2.ds
#      file on the backup data server.
#
DATA_BackupDB2Port=50000
#---------------------------------------------------


# Backup ObjectServer Hostname
#   This is an optional field.
#   Server will be configured as a standalone ObjectServer
 if leave blank.
#   Current value can be found in the following locations,
 if defined:
#   1. The OMNIbus server interface file
#   2. The key Gate.ObjectServerB.Server in the
#      $OMNIHOME/gates/NCO_GATE/NCO_GATE.props
#      file on the backup ObjectServer.
#
DATA_BackupObjectServerHostname=tbsmbackup.mycompany.com
#---------------------------------------------------

# Backup ObjectServer Port
#   The default value is 4100.
#   Current value can be found in the key
 ObjectServer_DS.ObjectServer.BACKUPPORT in the
#      /opt/IBM/tivoli/impact/etc/TBSM_ObjectServer_DS.ds
#      file on both the primary and the backup data server
(TBSM_B_ObjectServer_DS.ds).
#
DATA_BackupObjectServerPort=4100
#---------------------------------------------------

# Backup ObjectServer Name
#   Optional filed.  The default value is NCOMS_BKUP.
#   Current value can be found in the following locations:
#   1. The OMNIbus server interface file
#   2. The key Gate.ObjectServerA.Server in the
#      $OMNIHOME/gates/NCO_GATE/NCO_GATE.props
#      file on the backup ObjectServer.
#
DATA_BackupObjectServerName=NCOMS_BKUP
#---------------------------------------------------

# If ObjectServer is configured for failback then
# set the following property to true.
UseObjectServerFailback=false

# Bi-directional Gateway Name
#   Optional filed.  The default value is NCO_GATE.
#   Current value can be found in the following locations:
#   1. The OMNIbus server interface file
#   2. The generated Gateway directory and file names.
#   3. The key name in the $OMNIHOME/gates/NCO_GATE/NCO_GATE.props
#      file on the backup ObjectServer.
#
DATA_Bi-DirectionalGatewayName=NCO_GATE
#---------------------------------------------------

# Bi-directional Gateway Port
#   Optional filed.  The default value is 4300.
#   Current value can be found in the OMNIbus server interface file
#
DATA_Bi-DirectionalGatewayPort=4300
#---------------------------------------------------

#-------------------------------------------------------------
# Parameters used in Dashboard Server failover configuration
# The value of these parameters should match those in the Data
 Server configuration
#-------------------------------------------------------------
# Primary Data Server Hostname
#   This is a required field. Script will not run on dashboard
 server if leave blank.
#   It is set in the following locations:
#   1. The key impact.rmiupdateserver.hostname in the
#      /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear
/sla.war/etc/RAD_server.props
#      file on the dashboard server.
#   2. The key impact.sla.serverhost in the
#      /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear
/sla.war/etc/RAD_sla.props
```

```
#      file on the dashboard server.
#
DASH_PrimaryDataServerHostname=tbsmprimary.mycompany.com
#-----------------------------------------------------

# Primary Data Server RMI Port
#   The default value is 17542.
#   Current value can be found in the key
 impact.rmiupdateserver.port in the
#      /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear
/sla.war/etc/RAD_server.props
#      file on the dashboard server.
#
DASH_PrimaryDataServerRMIPort=17542
#-----------------------------------------------------

# Primary Data Server HTTP Port
#   The default value is 17310.
#   Current value can be found in the key impact.sla.serverhttpport
 in the
#      /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear
/sla.war/etc/RAD_sla.props
#      file on the dashboard server.
#
DASH_PrimaryDataServerHTTPPort=17310
#-----------------------------------------------------

# Backup Data Server Hostname
#   This is an optional field.  Data server will be configured as
 a standalone if leave blank.
#   It is set in key impact.rmiupdateserver.hostname.backup in the
#      /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear
/sla.war/etc/RAD_server.props
#      file on the dashboard server.
#
DASH_BackupDataServerHostname=tbsmbackup.mycompany.com
#-----------------------------------------------------

# Backup Data Server RMI Port
#   The default value is 17542.
#   Current value can be found in the key
 impact.rmiupdateserver.port.backup in the
#      /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear
/sla.war/etc/RAD_server.props
#      file on the dashboard server.
#
DASH_BackupDataServerRMIPort=17542
#-----------------------------------------------------

# Backup Data Server HTTP Port
#   The default value is 17310.
#   Current value can be found in the key
 impact.sla.serverhttpport.backup in the
#      /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear
/sla.war/etc/RAD_sla.props
#      file on the dashboard server.
#
DASH_BackupDataServerHTTPPort=17310
#-----------------------------------------------------

#-------------------------------------------------------------
# The following parameters are not used at this release
#-------------------------------------------------------------
# Primary ObjectServer Hostname - Reserved, Not used
DASH_PrimaryObjectServerHostname=

# Primary ObjectServer Port - Reserved, Not used
DASH_PrimaryObjectServerPort=4100

# Primary ObjectServer Name - Reserved, Not used
DASH_PrimaryObjectServerName=NCOMS

# Backup ObjectServer Hostname - Reserved, Not used
DASH_BackupObjectServerHostname=

# Backup ObjectServer Port - Reserved, Not used
DASH_BackupObjectServerPort=4100

# Backup ObjectServer Name - Reserved, Not used
DASH_BackupObjectServerName=NCOMS_BKUP

# Bi-directional Gateway Name - Reserved, Not used
```

```
DASH_Bi-DirectionalGatewayName=NCO_GATE

# Bi-directional Gateway Port - Reserved, Not used
DASH_Bi-DirectionalGatewayPort=4300
```

NCO_GATE.map file

The NCO_GATE.map file is typically located in the *installdirectory*/netcool/omnibus/gates/NCO_GATE/ directory. This file is updated automatically when you use the fo_config script. If you manually edit this file, add the entries that are added by the fo_config script.

Contents of the NCO_GATE.map file

The entries that are added automatically by the fo_config script are shown in bold type. If you manually edit this file, you must add all the entries shown in bold type. You must also add a comma (,) at the end of the following line: 'ServerSerial' = '@ServerSerial' ON INSERT ONLY

```
##############################################################################################
# Netcool/OMNIbus Bi-directional ObjectServer Gateway 7.2
#
# Default map definition file.
#
# Ident: $Id: objserv_bi.map 1.3 2003/10/13 10:36:35 cappl Development $
#
# Revision History:
#    1.1:    Initial revision.
#    1.2:    User related system table maps added.
#    1.3:    Dynamic tool tables and destop table maps added.
#
# Notes:
#
# Fields that are marked as 'ON INSERT ONLY' will only be passed when an event
# is inserted for the first time. (ie. they will not be updated). The ordering
# of the fields is not important as the gateway will use named value insertion.
#
##############################################################################################
CREATE MAPPING StatusMap
(
    'Identifier'       =     '@Identifier'        ON INSERT ONLY,
    'Node'           =    '@Node'            ON INSERT ONLY,
    'NodeAlias'       =    '@NodeAlias'        ON INSERT ONLY
                              NOTNULL '@Node',
    'Manager'       =    '@Manager'        ON INSERT ONLY,
    'Agent'           =    '@Agent'        ON INSERT ONLY,
    'AlertGroup'       =    '@AlertGroup'        ON INSERT ONLY,
    'AlertKey'       =    '@AlertKey'        ON INSERT ONLY,
    'Severity'       =    '@Severity',
    'Summary'       =    '@Summary',
    'StateChange'       =    '@StateChange',
    'FirstOccurrence'  =    '@FirstOccurrence'    ON INSERT ONLY,
    'LastOccurrence'   =    '@LastOccurrence',
    'InternalLast'       =    '@InternalLast',
    'Poll'           =    '@Poll'            ON INSERT ONLY,
    'Type'           =    '@Type'            ON INSERT ONLY,
    'Tally'           =    '@Tally',
    'Class'           =    '@Class'        ON INSERT ONLY,
    'Grade'           =    '@Grade'        ON INSERT ONLY,
    'Location'       =    '@Location'        ON INSERT ONLY,
    'OwnerUID'       =    '@OwnerUID',
    'OwnerGID'       =    '@OwnerGID',
    'Acknowledged'       =    '@Acknowledged',
    'Flash'           =    '@Flash',
    'EventId'       =    '@EventId'        ON INSERT ONLY,
    'ExpireTime'       =    '@ExpireTime'        ON INSERT ONLY,
    'ProcessReq'       =    '@ProcessReq',
    'SuppressEscl'       =    '@SuppressEscl',
    'Customer'       =    '@Customer'         ON INSERT ONLY,
    'Service'       =    '@Service'        ON INSERT ONLY,
    'PhysicalSlot'       =    '@PhysicalSlot'        ON INSERT ONLY,
    'PhysicalPort'       =    '@PhysicalPort'        ON INSERT ONLY,
    'PhysicalCard'       =    '@PhysicalCard'        ON INSERT ONLY,
    'TaskList'       =    '@TaskList',
    'NmosSerial'       =    '@NmosSerial'        ON INSERT ONLY,
    'NmosObjInst'       =    '@NmosObjInst'        ON INSERT ONLY,
    'NmosCauseType'       =    '@NmosCauseType',
```

```
    'LocalNodeAlias'   =    '@LocalNodeAlias'    ON INSERT ONLY,
    'LocalPriObj'       =    '@LocalPriObj'        ON INSERT ONLY,
    'LocalSecObj'       =    '@LocalSecObj'        ON INSERT ONLY,
    'LocalRootObj'       =    '@LocalRootObj'        ON INSERT ONLY,
    'RemoteNodeAlias'  =    '@RemoteNodeAlias'    ON INSERT ONLY,
    'RemotePriObj'       =    '@RemotePriObj'        ON INSERT ONLY,
    'RemoteSecObj'       =    '@RemoteSecObj'        ON INSERT ONLY,
    'RemoteRootObj'       =    '@RemoteRootObj'    ON INSERT ONLY,
    'X733EventType'    =    '@X733EventType'    ON INSERT ONLY,
    'X733ProbableCause'=    '@X733ProbableCause'    ON INSERT ONLY,
    'X733SpecificProb' =    '@X733SpecificProb'    ON INSERT ONLY,
    'X733CorrNotif'       =    '@X733CorrNotif'    ON INSERT ONLY,
    'URL'           =    '@URL'            ON INSERT ONLY,
    'ExtendedAttr'       =    '@ExtendedAttr'        ON INSERT ONLY,
    'ServerName'       =    '@ServerName'        ON INSERT ONLY,
    'ServerSerial'     =    '@ServerSerial'     ON INSERT ONLY,
  'BSM_Identity'          =    '@BSM_Identity',
  'BSM_ClassName'         =    '@BSM_ClassName',
  'BSM_SubIdentity'       =    '@BSM_SubIdentity',
  'RAD_ServiceName'       =    '@RAD_ServiceName',
  'RAD_SLAName'           =    '@RAD_SLAName',
  'RAD_ServiceID'         =    '@RAD_ServiceID',
  'RAD_ServiceTypeName'   =    '@RAD_ServiceTypeName',
  'RAD_ServiceTypeID'     =    '@RAD_ServiceTypeID',
  'RAD_FilterIDList'      =    '@RAD_FilterIDList',
  'RAD_FunctionName'      =    '@RAD_FunctionName',
  'RAD_UserFunctionName'  =    '@RAD_UserFunctionName',
  'RAD_IsIntermediateFunction'      =    '@RAD_IsIntermediateFunction',
  'RAD_FunctionType'      =    '@RAD_FunctionType',
  'RAD_TimeWindow'        =    '@RAD_TimeWindow',
  'RAD_TimeWindowStart'   =    '@RAD_TimeWindowStart',
  'RAD_TimeWindowEnd'     =    '@RAD_TimeWindowEnd',
  'RAD_TimeWindowLength'  =    '@RAD_TimeWindowLength',
  'RAD_CurrentRawMetric'  =    '@RAD_CurrentRawMetric',
  'RAD_CurrentRawMetricNumeric'     =    '@RAD_CurrentRawMetricNumeric',
  'RAD_CurrentRelativeMetric'       =    '@RAD_CurrentRelativeMetric',
  'RAD_CurrentFunctionState'        =    '@RAD_CurrentFunctionState',
  'RAD_ThresholdNum'                =    '@RAD_ThresholdNum',
  'RAD_ViolationThresholdRelativeM' =    '@RAD_ViolationThresholdRelativeM',
  'RAD_ViolationThresholdRawMetric' =    '@RAD_ViolationThresholdRawMetric',
  'RAD_ViolationRemainingRawMetric' =    '@RAD_ViolationRemainingRawMetric',
  'RAD_NextThresholdRelativeMetric' =    '@RAD_NextThresholdRelativeMetric',
  'RAD_NextThresholdRawMetric'      =    '@RAD_NextThresholdRawMetric',
  'RAD_NextThresholdRemainingRawM'  =    '@RAD_NextThresholdRemainingRawM',
  'RAD_TotalRawMetric'    =    '@RAD_TotalRawMetric',
  'RAD_RawInputValue'     =    '@RAD_RawInputValue',
  'RAD_RawInputLastValue' =    '@RAD_RawInputLastValue',
  'RAD_LastCount'         =    '@RAD_LastCount',
  'RAD_WebtopTool1'       =    '@RAD_WebtopTool1',
  'RAD_WebtopTool2'       =    '@RAD_WebtopTool2',
  'ITMStatus'             =    '@ITMStatus',
  'ITMDisplayItem'        =    '@ITMDisplayItem',
  'ITMEventData'          =    '@ITMEventData',
  'ITMTime'               =    '@ITMTime',
  'ITMHostname'           =    '@ITMHostname',
  'ITMIntType'            =    '@ITMIntType',
  'ITMResetFlag'          =    '@ITMResetFlag',
  'TECHostname'           =    '@TECHostname',
  'TECFQHostname'         =    '@TECFQHostname',
  'TECDate'               =    '@TECDate',
  'TECRepeatCount'        =    '@TECRepeatCount',
  'TECStatus'             =    '@TECStatus',
  'TECServerHandle'       =    '@TECServerHandle',
  'TECEventHandle'        =    '@TECEventHandle',
  'TECDateReception'      =    '@TECDateReception',
  'ticket_eligible'       =    '@ticket_eligible'

);

CREATE MAPPING JournalMap
(
    'KeyField'    =    TO_STRING(STATUS.SERIAL) + ":" +
                TO_STRING('@UID') + ":" +
                TO_STRING('@Chrono')    ON INSERT ONLY,
    'Serial'    =     STATUS.SERIAL,
    'Chrono'    =     '@Chrono',
    'UID'        =     TO_INTEGER('@UID'),
    'Text1'        =     '@Text1',
    'Text2'        =     '@Text2',
    'Text3'        =    '@Text3',
    'Text4'        =    '@Text4',
    'Text5'        =    '@Text5',
```

```
    'Text6'         =    '@Text6',
    'Text7'         =    '@Text7',
    'Text8'         =    '@Text8',
    'Text9'         =    '@Text9',
    'Text10'    =    '@Text10',
    'Text11'    =    '@Text11',
    'Text12'    =    '@Text12',
    'Text13'    =    '@Text13',
    'Text14'    =    '@Text14',
    'Text15'    =    '@Text15',
    'Text16'    =    '@Text16'
);

CREATE MAPPING DetailsMap
(
    'KeyField'    =    '@Identifier' + '####' +
                TO_STRING('@Sequence')    ON INSERT ONLY,
    'Identifier'    =     '@Identifier',
    'AttrVal'    =     '@AttrVal',
    'Sequence'    =     '@Sequence',
    'Name'        =     '@Name',
    'Detail'    =     '@Detail'
);
```

NCO_GATE.props file

If you manually configure the backup ObjectServer, create the NCOMS_BKUP.props file. If you use the fo_config script, this file is automatically created. This file is on the backup ObjectServer in the *installdirectory*/netcool/omnibus/etc/ directory.

Contents of the NCO_GATE.props file

The contents of the file is similar to the following example.

```
MessageLog                  : '$OMNIHOME/log/NCO_GATE.log'
Name                        : 'NCO_GATE'
PropsFile                   : '$OMNIHOME/etc/NCO_GATE.props'
Gate.MapFile                : '$OMNIHOME/gates/NCO_GATE/NCO_GATE.map'

Gate.ObjectServerA.Server   : 'NCOMS'
Gate.ObjectServerA.Username : 'root'
Gate.ObjectServerA.Password : ''
Gate.ObjectServerA.TblReplicateDefFile: '$OMNIHOME/gates/NCO_GATE/

objserv_bi.objectservera.tblrep.def'

Gate.ObjectServerB.Server   : 'NCOMS_BKUP'
Gate.ObjectServerB.Username : 'root'
Gate.ObjectServerB.Password : ''
Gate.ObjectServerB.TblReplicateDefFile: '$OMNIHOME/gates/NCO_GATE/

objserv_bi.objectserverb.tblrep.def'
```

NCOMS_BKUP.props file

If you manually configure the backup ObjectServer, change the value of BackupObjectServer in the NCOMS_BKUP.props file to TRUE.

Contents of the NCOMS_BKUP.props file

The contents of the file must be like the following example. In the actual file, there are typically some comments before the active values shown in this example. The line that you modify in your configuration is shown in bold type.

```
AlertSecurityModel: 0
AllowConnections: TRUE
AllowISQL: TRUE
AllowISQLWrite: TRUE
AllowTimedRefresh: FALSE
Auto.Debug: FALSE
Auto.Enabled: TRUE
```

```
Auto.StatsInterval: 60
BackupObjectServer: TRUE
Connections: 30
DTMaxTopRows: 100
DeleteLogging: FALSE
DeleteLogLevel: 0
DeleteLogSize: 1024
GWDeduplication: 0
Granularity: 60
Iduc.ListeningPort: 0
Ipc.SSLCertificate: ''
Ipc.SSLEnable: FALSE
Ipc.SSLPrivateKeyPassword: ''
MaxLogFileSize: 1024
Memstore.DataDirectory: '$OMNIHOME/db'
MessageLevel: 'warn'
PA.Name: 'NCO_PA'
PA.Password: ''
Profile: FALSE
ProfileStatsInterval: 60
RestrictPasswords: FALSE
RestrictProxySQL: FALSE
RestrictionUpdateCheck: TRUE
Sec.AuditLevel: 'warn'
UniqueLog: FALSE

# End of File
```

OMNIbus communications files

When you run the **nco_xigen** utility while configuring the ObjectServer communications on the primary and backup ObjectServer, the files contained in this topic are updated.

OMNIbus communications files for UNIX systems

The `interfaces.linux2x86` file is updated when you run the **nco_xigen** utility on a Linux® system. On other UNIX platforms, the format of the file is the same, but the name is based on the platform name. The `omni.dat` file is also updated.

`interfaces.`*`platform`* file for the primary OMNIbus server

The contents of the file must be like the following example.

Note: The fields `NCOMS`, `NCOMS_BKUP`, `NCO_GATE`, `tbsmprimary`, and `tbsmbackup` are default values. Your fields might be different.

```
# NCOMS => tbsmprimary 4100
NCOMS
    master tcp sun-ether tbsmprimary 4100
    query tcp sun-ether tbsmprimary 4100
    master tcp sun-ether tbsmbackup 4100
    query tcp sun-ether tbsmbackup 4100
# NCOMS_BKUP => tbsmbackup 4100
NCOMS_BKUP
    master tcp sun-ether tbsmbackup 4100
    query tcp sun-ether tbsmbackup 4100
# NCO_GATE => tbsmbackup 4300
NCO_GATE
    master tcp sun-ether tbsmbackup 4300
    query tcp sun-ether tbsmbackup 4300
#
# EOF
```

`interfaces.`*`platform`* file for the backup OMNIbus server

The contents of the file must be like the following example:

Note: The fields NCOMS, NCOMS_BKUP, NCO_GATE, tbsmprimary, and tbsmbackup are default values. Your fields might be different.

```
# NCOMS => tbsmprimary 4100
NCOMS
    master tcp sun-ether tbsmprimary 4100
    query tcp sun-ether tbsmprimary 4100

# NCOMS_BKUP => tbsmbackup 4100
NCOMS_BKUP
    master tcp sun-ether tbsmbackup 4100
    query tcp sun-ether tbsmbackup 4100

# NCO_GATE => tbsmbackup 4300
NCO_GATE
    master tcp sun-ether tbsmbackup 4300
    query tcp sun-ether tbsmbackup 4300

#
# EOF
```

omni.dat file for the primary ObjectServer

The contents of the file must be like the following example:

Note: The fields NCOMS, NCOMS_BKUP, NCO_GATE, tbsmprimary, and tbsmbackup are default values. Your fields might be different.

```
[NCOMS]
{
        Primary:        tbsmprimary 4100
        Backup:         tbsmbackup 4100
}
[NCOMS_BKUP]
{
        Primary:        tbsmbackup 4100
}
[NCO_GATE]
{
        Primary:        tbsmbackup 4300
}
```

omni.dat file for the backup ObjectServer

The contents of the file must be like the following example:

Note: The fields NCOMS, NCOMS_BKUP, NCO_GATE, tbsmprimary, and tbsmbackup are default values. Your fields might be different.

```
[NCOMS]
{
    Primary:    tbsmprimary 4100
}

[NCOMS_BKUP]
{
    Primary:    tbsmbackup 4100
}

[NCO_GATE]
{
    Primary:    tbsmbackup 4300
}
```

OMNIbus communications files for Windows systems

The `sql.ini` file is updated on both the primary and backup ObjectServer. This file is located in the *installdirectory*`\netcool\ini\` directory.

`sql.ini` file for the primary ObjectServer

The file must be like the following example:

Note: The fields `NCOMS`, `NCOMS_BKUP`, `NCO_GATE`, `tbsmprimary`, and `tbsmbackup` are default values. Your fields might be different.

```
[NCO_GATE]
query=NLWNSCK,tbsmbackup,4300
[NCOMS]
$BASE$00=NLWNSCK,tbsmprimary,4100
$BASE$01=NLWNSCK,tbsmbackup,4100
query=$BASE$00;$BASE$01;
master=NLWNSCK,tbsmprimary,4100
[NCOMS_BKUP]
query=NLWNSCK,tbsmbackup,4100
```

`sql.ini` file for the backup ObjectServer

The file must be like the following example:

Note: The fields `NCOMS`, `NCOMS_BKUP`, `NCO_GATE`, `tbsmprimary`, and `tbsmbackup` are default values. Your fields might be different.

```
[NCO_GATE]
query=NLWNSCK,tbsmbackup,4300
master=NLWNSCK,tbsmbackup,4300
[NCOMS]
query=NLWNSCK,tbsmprimary,4100
[NCOMS_BKUP]
query=NLWNSCK,tbsmbackup,4100
master=NLWNSCK,tbsmbackup,4100
```

`TBSM_consistency.props` file

The properties set in the `TBSM_consistency.props` file control the behavior of the consistency checker. The file is in the `/opt/IBM/tivoli/impact/etc` directory.

The consistency checker monitors the internal state model for inconsistencies based on actual event flow and model state. In cases where inconsistencies are found, the consistency checker modifies events and table entries to correct the inconsistencies. If you change any of the consistency checker properties, you must restart the Data server.

The following table lists the properties and default settings.

Property name	Description	Default
impact.consistency.interval	Specifies interval between runs of the consistency checker function.	11 minutes
impact.consistency.waitintervalsecs	Specifies the interval between runs of the consistency checker processing in seconds. If > 0 this value is used for the interval. Otherwise the minutes value *impact.consistency.interval* is used.	0 seconds
impact.consistency.threshold	Specifies the age at which events or status are considered inconsistent.	5 minutes
impact.consistency.autostart	If set to `false`, disables consistency checker.	true

Property name	Description	Default
impact.consistency.verbose	If set to `true`, causes detailed debugging information about the consistency checker to be written to the `messages.log` file in the `$IMPACT_HOME/wlp/usr/servers/TBSM/logs` directory.	false
impact.consistency.logtofile	If `true`, causes the `TBSM_consistency.log` file to be written. If `false`, this file is not written.	false
impact.consistency.maxforbatch	Specifies the maximum number of status events checked for a batch query, used when checking events existence when service dependency information is subject to repairs by consistency checking.	100

Model consistency function

In addition, you can enable the model consistency function with a separate setting in the `TBSM_sla.props` file that is located in the `/opt/IBM/tivoli/impact/etc` directory. By default, the **`impact.consistency.modelconsistencychecker.enabled`** property is set to `false`. If this property is set to `true`, the model checker is enabled.

The model consistency function ensures that TBSM reacts correctly to model changes that affect the service status calculations.

`xmltoolkitsvc.properties` file

The properties that are set in the `xmltoolkitsvc.properties` file influence the behavior of the Discovery Library Toolkit.

Name	Default value	Description	Updated by installer
General properties			
DL_PollBooksIntervalSeconds	15 seconds	The number of seconds that the service sleeps before checking for new data. Minimum: 5 seconds. If you use the `cmdbdiscovery` command to initiate bulk and delta imports, set this value to `-1`. The negative number turns off polling. The system waits until a request is made from the **`cmdbdiscovery`** command.	No
DL_PollTaddmIntervalSeconds	3600 seconds	The number of seconds that the service sleeps before checking Tivoli Application Dependency Discovery Manager for new data.	No
DL_PollApiIntervalSeconds	120 seconds	The number of seconds that the service sleeps before checking API queue.	No
DL_FileSystem	N/A	The file system that the service looks into for books to read.	Yes

Name	Default value	Description	Updated by installer
DL_ExportFileSystem	N/A	The file system that the `genidml` script will write the generated IDML book into.	Yes
DL_Interest	N/A	Limits the books to be read to the authors listed.	No
DL_Ignore	N/A	Removes the specified authors (sources) from the books read.	No
DL_TmpFile_CleanupInterval	7200	Number of minutes for which temporary files are kept. After this interval, the files are deleted.	No
DL_ImportSource	N/A	Defines the data import sources for the discovery library toolkit. valid values are: • `books` - if IDML books are to be read - • `taddm` - if resources are to be imported from Tivoli Application Dependency Discovery Manager • `all` - if IDML books and Tivoli Application Dependency Discovery Manager are sources.	Yes
DL_Books_Immediate_Poll	true	When the toolkit service is started, it immediately reads all available books. After this request, it waits `DL_PollBooksIntervalSeconds` until the next request. By setting this property to false, the service instead waits `DL_PollBooksIntervalSeconds` before making its first request for data.	No
DL_TADDM_Immediate_Poll	true	Set to `true` if the service must connect directly with Tivoli Application Dependency Discovery Manager to retrieve data. If set to `false`, data is read from Discovery Library books found in the `DL_FileSystem` property. If this property has been retired, use `DL_IMPORT_SOURCE` to specify the source of data.	No
DL_TADDM_DropL2IntfFDB	true	Automatically filter L2 Interface objects that represent temporary objects	No
DL_TADDM_DropMACDevices	true	Automatically filter all Computer systems with type `MACDevice` or `IPDevice`.	No
DL_TADDM_DropFDBObjects	true	Automatically filter temporary objects stored in the TADDM database	No

Name	Default value	Description	Updated by installer
DL_LifeCycleState_Filter	false	Allows additional filtering of TADDM business applications. This allows you to work with a subset of the business applications defined in TADDM that are being imported into TBSM. For more information, see the `xmltoolkitsvc.properties` file.	
TADDM connection properties			
DL_TADDM_HostName	N/A	Tivoli Application Dependency Discovery Manager host name or IP address	Yes
DL_TADDM_Port	9433	Tivoli Application Dependency Discovery Manager RMI port	Yes
DL_TADDM_GUI_Port	9430	Tivoli Application Dependency Discovery Manager HTTP port. This port is used for launch in context when launching the Tivoli Application Dependency Discovery Manager UI.	Yes
DL_TADDM_Retry_Limit	86400	The number of seconds that the system tries to connect to Tivoli Application Dependency Discovery Manager. When the wait limit is exceeded, then the toolkit service is terminated.	No
DL_TADDM_SSL	false	Specifies whether SSL must be used on the connection with Tivoli Application Dependency Discovery Manager.	Yes
DL_TADDM_SSL_Port	9433	If SSL is used on the connection with TADDM, this property specifies the port that must be used on the connection. The port is specified on the TADDM server in `collation.properties`.	Yes
TADDM database connection properties			
DL_TADDM_DBManager.ObjectURL	N/A	Database URL for the TADDM database.	Yes
DL_TADDM_DBManager.Drive	N/A	Database driver.	Yes
DL_TADDM_DBManager.Type	N/A	Identifies the type of database.	Yes
DL_TADDM_DBManager.Schema	N/A	Specifies the database schema.	Yes
DL_TADDM_DBManager.FetchSize	1000	The result set fetch size used on requests from the TADDM database.	No
TADDM Import Retrieval Process			

Name	Default value	Description	Updated by installer
DL_TADDM_JDBC_Import	true	Indicates whether or not CI instance and relationship data is retrieved directly from the TADDM database.	No
DL_TADDM_JDBC_DeltaImport	true	Control only delta imports and allow JDBC to be used for bulk imports but turn off JDBC for deltas.	No
DL_TADDM_AnalyzeChangeHistoryTable	true	When DL_TBSM_JDBC_Import is set to true, the delta processing requires the use of the table tbsm_change_history_table. When data is moved into this table, the table statistics are updated immediately after the move. If the database user does not have authority to update table statistics, then this property should be set to false to avoid exceptions in the log.	No
DL_FilterAttributesOnRequest	true	Reduces the relationships retrieved from TADDM by eliminating relationships associated with classes not imported.	No
DL_TADDM_Mode	storageserver	The TADDM server can operate in one of three modes as defined in the TADDM collation.properties file: • storageserver (com.collation.taddm.mode) - • domain (com.collation.cmdbmode) • enterprise (com.collation.cmdbmode) This property is only relevant where TBSM connects to an enterprise server. If TBSM connects to a domain or Storage server, tthe default value is sufficient.	No
DL_TADDM_JDBC_Import	true	Indicates whether or not CI instance and relationship data is retrieved directly from the TADDM database.	No
DL_TADDM_JDBC_DeltaImport	true	Control only delta imports and allow JDBC to be used for bulk imports but turn off JDBC for deltas.	No
TBSM connection properties			
DL_TBSM_Hostname	N/A	Host name running the TBSM server. Defaults to the localhost.	Yes

Name	Default value	Description	Updated by installer
DL_TBSM_HTTP_Port	17311 for TBSM 6.1 17310 for TBSM 4.2.x	TBSM HTTP port used by the TBSM data server.	Yes
DL_TBSM_Invalidate	true	Instructs the toolkit service to automatically invalidate server resources when discovery library resources have been added or modified.	No
DL_TBSM_Server_Classpath	N/A	Classpath additions for TBSM server-specific jar files.	Yes
DL_Verify_Book_Copied	true	Before reading an IdML book, the toolkit checks to ensure that the books is not currently being written to the file system. If the book is being written, the book is skipped and read on the next polling interval. This operation incurs a delay of up to 10 seconds per book, if you can ensure that the toolkit does not try to read a book while it is being copied, set this property to `false` in order to eliminate the delay.	No
DL_TBSM_SSL	false	Indicates whether SSL must be used on the connection with the TBSM server. If `true` is specified, be sure to update the **`DL_TBSM_HTTP_Port`** to the TBSM data server SSL port.	Yes
Database connection properties			
DL_DBManager.ObjectURL	N/A	JDBC connection URL. Specifics are dependent on the database. See the property file for additional information.	Yes
DL_DBManager.Driver	N/A	JDBC driver class name. Specifics are dependent on the database. See the property file for additional information.	Yes
DL_DBManager.Driver.classpath	N/A	Classpath for JDBC drivers.	Yes
DL_DBManager.MaxConnect	3	Limits the number of simultaneous connections to the database.	No
DL_DBManager.Type	DB2	Identifies the type of database. Recognized values: DB2.	Yes
DL_DBBatchSize	500	Controls the number of SQL statements batched before writing to the database.	No
DL_DBBatchFileLocation	`../xml/scr/batch`	File system used for writing SQL batch instructions associated with the SQL copy statement.	No

Name	Default value	Description	Updated by installer
DL_DBBatchFileLocationDatabase	`../xml/scr/batch`	If the above file system is shared, and the share location is different, then this property specifies the share name.	No
DL_XFORM_IO_BUFFER_Size	262144	Input buffer size associated with the reading of transform files.	No
DL_DBManager.conf.bulk.work_mem	32768	Sets the work_mem value for Postgres connections used by the toolkit. This overrides the value set in `postgresql.conf`. **Note:** Only used when running the toolkit on a TBSM 4.2.x system.	No
DL_RelationshipBatch	400	Number of GUID's collected before relationship information for these GUID's is requested. This is used for JDBC requests.	No
DL_RelationshipBatch4Api	50	Number of GUID's collected before relationship information for these GUID's is requested. This property is used when relationships are retrieved using the TADDM API. Maximum 50.	No
DL_GuidAliasBatch	200	Number of GUID's collected before GUID alias information for these GUID's is requested.	No
Transform properties			
DL_Xform_Use_BCP	true	Indicates whether the use of SQL copy statements is allowed. Using copy statements increases write performance to the database.	No
DL_RMI_PORT	12315	Discovery Library Toolkit RMI Port.	Yes
Database maintenance			
DL_DBVacuum_Start	01:00	Specifies the time when the DL tables must be vacuumed. The value specified must use a 24-hour clock (that is, 13:00 is 1 PM). Specify hours and minutes. Invalid values result in the default value being used.	No
DL_DBVacuum_Iterations	1	Specifies how many times the tables are vacuumed within a 24-hour period.	
DL_DBVacuum_Full	true	Issue a **`VACUUM FULL ANALYZE`** if true is specified.	
Java virtual machine properties			
ms	64000000	Initial heap size.	No
mx	100000000	Max heap size.	No

Name	Default value	Description	Updated by installer
Failover properties			
DL_Preferred_Primary	true	In a failover environment, this property indicates the instance that operates as the primary Discovery Library Toolkit instance. If the instance for which this property is set to true is the primary instance.	Yes
DL_Failover_Wait_Interval	420	When acting in secondary mode, determines how long the toolkit will wait for the acting primary to update its "heartbeat" before assuming the role of primary. The specified value is in seconds.	No
DL_Auto_Failback	false	Indicates whether the secondary toolkit should automatically fail back to the preferred primary toolkit instance when the secondary detects that the preferred primary is available again.	No
DL_Alternate_TBSM_Hostname	N/A	Whether the alternate is the primary or backup data server depends on DL_Preferred_Primary. If this is true, then the alternate is the backup data server. If DL_Preferred_Primary is false then the alternate is the primary data server.	Yes
DL_Alternate_TBSM_HTTP_Port	17310	The port that the alternate TBSM Data server is listening on.	Yes
DL_Alternate_TBSM_SSL	false	Should SSL be used on the connection to alternate TBSM server. Specify either true or false.	Yes
API Processing properties			
DL_Enable_API_Thread	true	Controls whether or not the thread used for processing data via the API is started.	No
DL_API_Record_Read_Limit	1000	Specifies the maximum number of records to be read/processed by the API thread at one time.	No
DL_API_Transaction_Age	24	Specifies how long (in hours) to keep transaction related data for completed/processed transactions in the API tables.	No

Name	Default value	Description	Updated by installer
DL_API_Cleanup_Process_Max_Wait	60	Specifies how long (in minutes) that a busy API processing thread will continue to run before pausing to invoke cleanup processing. Cleanup processing will delete aged API transaction records from the API tables for completed transactions.	No

Dashboard Application Service Hub overview

Web-based products built on the Dashboard Application Service Hub framework share a common user interface where you can launch applications and share information.

Dashboard Application Service Hub helps the interaction and secure passing of data between Tivoli products through a common portal. You can launch from one application to another and within the same dashboard view research different aspects of your managed enterprise.

Dashboard Application Service Hub is installed automatically with the first Tivoli product using the Dashboard Application Service Hub framework. Subsequent products may install updated versions of Dashboard Application Service Hub.

Dashboard Application Service Hub provides the following features:

- A Web based user interface for individual products and for integrating multiple products.
- A single, task-based navigation panel for multiple products. Users select actions based around the task that they want to complete, not by the product that supports that task.
- Single sign-on (SSO), consolidated user management, and a single point of access for different Tivoli applications.
- Aggregated views that span server instances, such as the Tivoli Netcool/OMNIbus ObjectServer.
- Inter-view messaging between products to support contextual linkage between applications.
- The ability to create customized pages and administer access to content by user, role, or group.

Accepting the security certificate

When logging in, you might see a security alert with a message that says there is a problem with the security certificate. This indicates that the browser application is verifying the security certificate of the application server.

Self-signed or CA-signed certificate

The application server uses a self-signed security certificate. You might see a Security Alert when you first connect to the portal that alerts you to a problem with the security certificate. You might be warned of a possible invalid certificate and be recommended to not log in.

Although this warning appears, the certificate is valid and you can accept it. Or, if you prefer, you can install your own CA-signed certificate. For information on creating your own CA-signed certificate, go to: http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/tsec_sslcreateCArequest.html

For more information about certificates, go to the IBM WebSphere Application Server Community Edition Documentation Project at http://publib.boulder.ibm.com/wasce/V2.1.1/en/overview.html, and search for *Managing trust* and *Managing SSL certificates*.

Logging in

Log in to the portal whenever you want to start a work session.

Before you begin

The Dashboard Application Service Hub Server must be running before you can connect to it from your browser.

About this task

Complete these steps to log in:

Procedure

1. In a Web browser, enter the URL of the Dashboard Application Service Hub Server: `http://`*`host.domain`*`:16310/ibm/console` or `https://`*`host.domain`*`:16311/ibm/console` if it is configured for secure access.
   - *`host.domain`* is the fully qualified host name or IP address of the Dashboard Application Service Hub Server (such as *MyServer.MySubdomain.MyDomain.com* or 9.51.111.121, or `localhost` if you are running the Dashboard Application Service Hub Server locally).
   - `16310` is the default nonsecure port number for the portal and `16311` is the default secure port number. If your environment was configured with a port number other than the default, enter that number instead. If you are not sure of the port number, read the application server profile to get the correct number.
   - `ibm/console` is the default path to the Dashboard Application Service Hub Server, however this path is configurable and might differ from the default in your environment.
2. In the login page, enter your user ID and password and click **Log in**.

   This is the user ID and password that are stored with the Dashboard Application Service Hub Server.

   Attention: After authentication, the web container used by the Dashboard Application Service Hub Server redirects to the last URL requested. This is usually `https://<host>:<port>/ibm/console`, but if you manually change the page URL, after being initially directed to the login page, or if you make a separate request to the server in a discrete browser window before logging in, you may be redirected unexpectedly.

   Note: If you have more than one instance of the Dashboard Application Service Hub Server installed on your computer, you should not run more than one instance in a browser session, that is, do not log in to different instances on separate browser tabs.

Results

After your user credentials have been verified, the Welcome page is displayed. If you entered the localhost or port number incorrectly, the URL will not resolve. View the application server profile to check the settings for localhost, port, and user ID.

What to do next

Select any of the items in the navigation tree to begin working with the console.

While you are logged into the Dashboard Application Service Hub Server, avoid clicking the browser **Back** button because you will be logged out automatically. Click **Forward** and you will see that your are logged out and must resubmit your credentials to log in again.

Note: If you want to use single sign-on (SSO) then you must use the fully qualified domain name of the Dashboard Application Service Hub host.

Port assignments

The application server requires a set of sequentially numbered ports.

The sequence of ports is supplied during installation in the response file. The installer checks that the number of required ports (starting with the initial port value) are available before assigning them. If one of the ports in the sequence is already in use, the installer automatically terminates the installation process and you must specify a different range of ports in the response file.

Viewing the application server profile

Open the application server profile to review the port number assignments and other information.

About this task

The profile of the application server is available as a text file on the computer where it is installed.

Procedure

1. Locate the `/opt/IBM/JazzSM/profile/logs` directory.
2. Open `AboutThisProfile.txt` in a text editor.

Example

This is the profile for an installation on in a Windows environment as it appears in `/opt/IBM/JazzSM/profile/logs\AboutThisProfile.txt`:

```
Application server environment to create: Application server
Location: C:\Program Files\IBM\JazzSM\profile
Disk space required: 200 MB
Profile name: DASHProfile
Make this profile the default: True
Node name: TIPNode Host name: tivoliadmin.usca.ibm.com
Enable administrative security (recommended): True
Administrative consoleport: 16315
Administrative console secure port: 16316
HTTP transport port: 16310
HTTPS transport port: 16311
Bootstrap port: 16312
SOAP connector port: 16313
Run application server as a service: False
Create a Web server definition: False
```

What to do next

If you want to see the complete list of defined ports on the application server, you can open `/opt/IBM/JazzSM/var/JazzSMProfile_portDef.properties` in a text editor:

```
#Create the required WAS port properties for TIP
#Mon Oct 06 09:26:30 PDT 2008
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=16323
WC_adminhost=16315
DCS_UNICAST_ADDRESS=16318
BOOTSTRAP_ADDRESS=16312
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=16321
SOAP_CONNECTOR_ADDRESS=16313
ORB_LISTENER_ADDRESS=16320
WC_defaulthost_secure=16311
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=16322
WC_defaulthost=16310
WC_adminhost_secure=16316
```

Configuring

Once you have installed Dashboard Application Service Hub, you can configure it to operate in a variety of ways, for example, you can enable load balancing and employ a central user repository.

Adding an external LDAP repository

After installation, you can add an IBM Tivoli Directory Server or Active Directory Microsoft Active Directory Server as an LDAP repository for *Dashboard Application Service Hub*.

About this task

To add a new LDAP repository:

Procedure

1. Log in to the *Dashboard Application Service Hub*.
2. In the navigation pane, click **Settings** > **Websphere Admin Console** and click **Launch Websphere Admin Console**.
3. In the WebSphere Application Server administrative console, select **Security** > **Global security**.
4. From the **Available realm definitions** list, select **Federated repositories** and click **Configure**.
5. In the Related Items area, click the **Manage repositories** link and then click **Add** to add a new LDAP repository.
6. In the **Repository identifier** field, provide a unique identifier for the repository.

   The identifier uniquely identifies the repository within the cell, for example, LDAP1.
7. From the **Directory type** list, select the type of LDAP server.

   The type of LDAP server determines the default filters that are used by WebSphere Application Server.

   Note: IBM Tivoli Directory Server users can choose either IBM Tivoli Directory Server or SecureWay as the directory type. For better performance, use the IBM Tivoli Directory Server directory type.
8. In the **Primary host name** field, enter the fully qualified host name of the primary LDAP server.

   The primary host name and the distinguished name must contain no spaces. You can enter either the IP address or the domain name system (DNS) name.
9. In the **Port** field, enter the server port of the LDAP directory.

   The host name and the port number represent the realm for this LDAP server in a mixed version nodes cell. If servers in different cells are communicating with each other using Lightweight Third Party Authentication (LTPA) tokens, these realms must match exactly in all the cells.

   Note:

   The default port value is 389, which is not a Secure Sockets Layer (SSL) connection port. Use port 636 for a Secure Sockets Layer (SSL) connection. For some LDAP servers, you can specify a different port. If you do not know the port to use, contact your LDAP server administrator.
10. Optional: In the **Bind distinguished name** and **Bind password** fields, enter the bind distinguished name (DN) (for example, `cn=root`) and password.

    Note: The bind DN is required for write operations or to obtain user and group information if anonymous binds are not possible on the LDAP server. In most cases, a bind DN and bind password are needed, except when an anonymous bind can satisfy all of the required functions. Therefore, if the LDAP server is set up to use anonymous binds, leave these fields blank.
11. Optional: In the **Login properties** field, enter the property names used to log into the WebSphere Application Server.

    This field takes multiple login properties, delimited by a semicolon (`;`). For example, `cn`.
12. Optional: From the **Certificate mapping** list, select your preferred certificate map mode. You can use the X.590 certificates for user authentication when LDAP is selected as the repository.

Note: The **Certificate mapping** field is used to indicate whether to map the X.509 certificates into an LDAP directory user by `EXACT_DN` or `CERTIFICATE_FILTER`. If you select `EXACT_DN`, the DN in the certificate must match the user entry in the LDAP server, including case and spaces.

13. Click **OK**.
14. In the Messages area at the top of the **Global security** page, click the **Save** link and log out of the WebSphere Application Server console.

What to do next

Configure the Dashboard Application Service Hub Server to communicate with an external LDAP repository.

Configuring an external LDAP repository

You can configure the Dashboard Application Service Hub Server to communicate with an external LDAP repository.

About this task

In a load balanced environment, all Dashboard Application Service Hub Server instances must be configured separately for the LDAP server. To configure an application server to communicate with an external LDAP repository:

Procedure

1. Log in to *Dashboard Application Service Hub*.
2. In the navigation pane, click **Settings** > **Websphere Administrative Console** and click **Launch Websphere Administrative Console**.
3. In the WebSphere Application Server administrative console, select **Security** > **Global security**.
4. From the **Available realm definitions** list, select **Federated repositories** and click **Configure**.
5. To add an entry to the base realm:
   a) Click **Add Base entry to Realm**.
   b) Enter the distinguished name (DN) of a base entry that uniquely identifies this set of entries in the realm.

      This base entry must uniquely identify the external repository in the realm.

      Note: If multiple repositories are included in the realm, use the DN field to define an additional distinguished name that uniquely identifies this set of entries within the realm. For example, repositories LDAP1 and LDAP2 might both use `o=ibm,c=us` as the base entry in the repository. So `o=ibm,c=us` is used for LDAP1 and `o=ibm2,c=us` for LDAP2. The specified DN in this field maps to the LDAP DN of the base entry within the repository (such as `o=ibm,c=us` b). The base entry indicates the starting point for searches in this LDAP directory server (such as `o=ibm,c=us` c).
   c) Click **OK**.
   d) In the Messages area at the top of the **Global security** page, click the **Save** link and log out of the WebSphere Application Server console.
6. In the WebSphere Application Server administrative console, select **Security** > **Global security**.
7. From the **Available realm definitions** list, select **Federated repositories** and click **Set as current** to mark the federated repository as the current realm.
8. Stop and restart the Dashboard Application Service Hub Server:
   a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
      - Windows `stopServer.bat server1`
      - Linux UNIX `stopServer.sh server1`

Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

- Windows `startServer.bat server1`
- Linux UNIX `startServer.sh server1`

9. Verify that the federated repository is correctly configured:
   a) In the portal navigation pane, click **Users and Groups** > **Manage Users**.
   b) Select **User ID** from the **Search by** list.
   c) Click **Search** to search for users in the federated repository.
   d) Confirm that the list includes users from both the LDAP repository and the local file registry.

   On the Dashboard Application Service Hub Server, LDAP users are queried only by the `userid` attribute. When users are imported into LDAP using an LDAP Data Interchange Format (LDIF) file, an auxiliary class of type `eperson` and an `uid` attribute is added to the LDAP user ID. Note that this is to be done only if you want to search the LDAP repository using VMM from the server.

What to do next

To be able to create or manage users in the portal that are defined in your LDAP repository, in the WebSphere Application Server administrative console, you must specify the supported entity types.

Managing LDAP users in the console

To create or manage users in the portal that are defined in your LDAP repository, in the WebSphere Application Server administrative console specify the supported entity types.

About this task

To create or manage LDAP users in the portal:

Procedure

1. Log in to the *Dashboard Application Service Hub*.
2. In the navigation pane, click **Settings** > **Websphere Admin Console** and click **Launch Websphere Admin Console**.
3. In the WebSphere Application Server administrative console, select **Security** > **Global security**.
4. From the **Available realm definitions** list, select **Federated repositories** and click **Configure**.
5. In the Additional Properties area, click **Supported entity types**, to view a list of predefined entity types.
6. Click the name of a predefined entity type to change its configuration.
7. In the **Base entry for the default parent** field, provide the distinguished name of a base entry in the repository.

   This entry determines the default location in the repository where entities of this type are placed on write operations by user and group management.
8. In the **Relative Distinguished Name properties** field, provide the relative distinguished name (RDN) properties for the specified entity type.

   Possible values are `cn` for **Group**, `uid` or `cn` for **PersonAccount**, and `o`, `ou`, `dc`, and `cn` for **OrgContainer**.

   Delimit multiple properties for the **OrgContainer** entity with a semicolon (`;`).
9. Click **OK** to return to the **Supported entity types** page.
10. In the Messages area at the top of the **Global security** page, click the **Save** link and log out of the WebSphere Application Server console.

11. For the changes to take effect, stop, and restart the Dashboard Application Service Hub Server.

    In a load balanced environment, you must stop and restart each Dashboard Application Service Hub Server instance.
12. Stop and restart the Dashboard Application Service Hub Server:
    a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
    - Windows `stopServer.bat server1`
    - Linux UNIX `stopServer.sh server1`

      Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

    b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
    - Windows `startServer.bat server1`
    - Linux UNIX `startServer.sh server1`

Results

You can now manage your LDAP repository users in the portal through the **Settings** > **Manage Users** menu items.

Note: When you add a new user, you should check that the user ID you specify does not already exist in any of the user repositories to avoid difficulties when the new user attempts to log in.

Restriction: You cannot currently update user IDs through the **Settings** > **Manage Users** portlet that have been created in Microsoft Active Directory repositories.

Configuring an SSL connection to an LDAP server

If your implementation of *Dashboard Application Service Hub* uses an external LDAP-based user repository, such as Microsoft Active Directory, you can configure it to communicate over a secure SSL channel.

Before you begin

This task assumes that you have already an existing connection to an LDAP server set up.

Your LDAP server (for example, an IBM Tivoli Directory Server Version 6 or an Microsoft Active Directory server), must be configured to accept SSL connections and be running on secured port number (636). Refer to your LDAP server documentation if you need to create a signer certificate, which as part of this task, must be imported from your LDAP server into the trust store of the Dashboard Application Service Hub Server.

About this task

Follow these instructions to configure the Dashboard Application Service Hub Server to communicate over a secure (SSL) channel with an external LDAP repository. All application server instances must be configured for the LDAP server.

Procedure

1. Log in to the portal.
2. Follow these steps to import your LDAP server's signer certificate into the application server trust store.
   a) In the navigation pane, click **Settings** > **Websphere Admin Console** and click **Launch Websphere Admin Console**.

b) In the WebSphere Application Server administrative console navigation pane, click **Security** > **SSL certificate and key management**.

c) In the Related Items area, click the **Key stores and certificates** link and in the table click the **NodeDefaultTrustStore** link.

d) In the Additional Properties area, click the **Signer certificates** link and click the**Retrieve from port** button.

e) In the relevant fields, provide hostname, port (normally 636 for SSL connections), SSL configuration details, as well as the alias of the certificate for your LDAP server and click the **Retrieve signer information** button and then click **OK**.

3. Follow these steps to enable SSL communications to your LDAP server:

a) In the navigation pane, click **Security** > **Secure administration, applications, and infrastructure**.

b) Select **Federated repositories** from the **Available realm definitions** drop down list and click **Configure**.

c) Select your LDAP server from the **Repository** drop down list.

d) Enable the **Require SSL communications** check box and the select the **Centrally managed** option.

e) Click **OK**.

4. For the changes to take effect, save, stop, and restart all Dashboard Application Service Hub Server instances.

What to do next

If you intend to enable single sign-on (SSO) so that users can log in once and then traverse to other applications without having to re-authenticate, configure SSO.

Configuring an SSL connection to the ObjectServer

For environments that include a Tivoli Netcool/OMNIbus ObjectServer user registry, you need to set up encrypted communications on the Dashboard Application Service Hub Server.

About this task

Follow these steps to establish a secure channel for communications between the Dashboard Application Service Hub Server and the ObjectServer.

Procedure

1. Retrieve the ObjectServer certificate information, as follows:

a) In the navigation pane, click **Settings** > **Websphere Admin Console** and click **Launch Websphere Admin Console**.

b) In the WebSphere Application Server administrative console navigation pane, click **Security** > **SSL certificate and key management**.

c) On the SSL certificate and key management page, click **Key stores and certificates** and on the page that is displayed, click **NodeDefaultTrustStore**.

d) On the NodeDefaultTrustStore page, click **Signer certificates** and on the page that is displayed, click **Retrieve from port**.

e) In the relevant fields, enter **Host**, **Port**, and **Alias** values for the ObjectServer and click **Retrieve signer information**.

The signer information is retrieved and stored. For your reference, when the signer information has been retrieved, the following details are displayed:

Serial number
Specifies the certificate serial number that is generated by the issuer of the certificate.

Issued to
Specifies the distinguished name of the entity to which the certificate was issued.

Issued by
Specifies the distinguished name of the entity that issued the certificate. This name is the same as the issued-to distinguished name when the signer certificate is self-signed.

Fingerprint (SHA digest)
Specifies the Secure Hash Algorithm (SHA hash) of the certificate, which can be used to verify the certificate's hash at another location, such as the client side of a connection.

Validity period
Specifies the expiration date of the retrieved signer certificate for validation purposes.

2. Open `/opt/IBM/JazzSM/profile/etc/com.sybase.jdbc3.SybDriver.props` in a text editor and change these parameters:
   a) Enable SSL for ObjectServer primary host: `USESSLPRIMARY=TRUE`
   b) Enable SSL for ObjectServer backup host: `USESSLBACKUP=TRUE`
3. Stop and restart the Dashboard Application Service Hub Server:
   a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
   - Windows `stopServer.bat server1`
   - Linux UNIX `stopServer.sh server1`

     Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

   b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
   - Windows `startServer.bat server1`
   - Linux UNIX `startServer.sh server1`

Configuring VMM for the ObjectServer

When your Tivoli Netcool/OMNIbus ObjectServer is in a federated repository, use the script provided with Dashboard Application Service Hub to configure the Virtual Member Manager adapter for the ObjectServer.

Before you begin

Have the following ObjectServer information at hand: administrator name and password, IP address, and port number. If you have a second ObjectServer for failover support, you need the IP address and port number. The ObjectServer must be running at the time of installing *Dashboard Application Service Hub*, as the installation process attempts to connect to the ObjectServer.

About this task

The script assumes that the DASH installation directory is the parent directory and that the cell name is JazzSMNode01Cell. Run the VMM configuration script on every computer where the application server is installed.

Procedure

1. Change to the *install_dir*`\bin` directory.
   The directory contains a script to run:
   - Windows `confvmm4ncos.bat`
   - Linux `confvmm4ncos.sh`

- UNIX `confvmm4ncos.sh`

2. Enter the following command at the command line: `confvmm4ncos user password address port [address2 port2]`where
   a) `user` is the ID of a user with administrative privileges for this ObjectServer
   b) `password` is the password for the user ID
   c) `address` is the IP address of the ObjectServer
   d) `port` is the port number used by the ObjectServer
   e) Optional: `address2` and `port2`, if there is a failover server, is the IP address and port number of the failover ObjectServer

Results
The VMM adapter is configured for the ObjectServer. Thereafter, whenever the user registry needs to be accessed, the VMM adapter is called for this information.

Single sign-on

The single sign-on (SSO) capability in Tivoli products means that you can log on to one Tivoli application and then launch to other Tivoli Web-based or Web-enabled applications without having to re-enter your user credentials.

The repository for the user IDs can be the Tivoli Netcool/OMNIbus ObjectServer or a Lightweight Directory Access Protocol (LDAP) registry. A user logs on to one of the participating applications, at which time their credentials are authenticated at a central repository. With the credentials authenticated to a central location, the user can then launch from one application to another to view related data or perform actions. Single sign-on can be achieved between applications deployed to Dashboard Application Service Hub servers on multiple machines.

Single sign-on capabilities require that the participating products use Lightweight Third Party Authentication (LTPA) as the authentication mechanism. When SSO is enabled, a cookie is created containing the LTPA token and inserted into the HTTP response. When the user accesses other Web resources (portlets) in any other application server process in the same Domain Name Service (DNS) domain, the cookie is sent with the request. The LTPA token is then extracted from the cookie and validated. If the request is between different cells of application servers, you must share the LTPA keys and the user registry between the cells for SSO to work. The realm names on each system in the SSO domain are case sensitive and must match exactly. See Managing LTPA keys from multiple WebSphere Application Server cells on the WebSphere Application Server Information Center.

Configuring single sign-on
Use these instructions to establish single sign-on support and configure a federated repository.

Before you begin
Configuring SSO is a prerequisite to integrating products that are deployed on multiple servers. All Dashboard Application Service Hub Server instances must point to the central user registry (such as a Lightweight Directory Access Protocol server).

Attention: ITM single sign on (SSO) support is only available with ITM Version 6.2 Fix Pack 1 or higher.

About this task
To configure the WebSphere federated repositories functionality for LDAP:

Procedure
1. Log in to the *Dashboard Application Service Hub*.

2. In the navigation pane, click **Settings** > **Websphere Administrative Console** and click **Launch Websphere administrative console**.
3. In the WebSphere Application Server administrative console navigation pane, click **Security** > **Global security**.
4. In the **Authentication** area, expand **Web security** and click **Single sign-on**.
5. Click the **Enabled** option if SSO is disabled.
6. Click **Requires SSL** if all of the requests are expected to use HTTPS.
7. Enter the fully-qualified domain names in the Domain name field where SSO is effective.

   If the domain name is not fully qualified, the Dashboard Application Service Hub Server does not set a domain name value for the **LtpaToken** cookie and SSO is valid only for the server that created the cookie. For SSO to work across Tivoli applications, their application servers must be installed in same domain (use the same domain name).
8. Optional: Enable the **Interoperability Mode** option if you want to support SSO connections in WebSphere Application Server version 5.1.1 or later to interoperate with previous versions of the application server.
9. Optional: Enable the **Web inbound security attribute propagation** option if you want information added during the login at a specific Tivoli Enterprise Portal Server to propagate to other application server instances.
10. After clicking **OK** to save your changes, stop and restart all the Dashboard Application Service Hub Server instances.

What to do next

Note: When you launch *Dashboard Application Service Hub*, you must use a URL in the format protocol:// host.domain:port /*. If you do not use a fully-qualified domain name, *Dashboard Application Service Hub* cannot use SSO between Tivoli products.

Load balancing

You can setup a load balancing cluster of portal nodes with identical configurations to evenly distribute user sessions.

Load balancing is ideal for *Dashboard Application Service Hub* installations with a large user population. When a node within a cluster fails, new user sessions are directed to other active nodes.

You can create a load balanced cluster from an existing stand-alone application server instance, but must export its data before you configure it for load balancing. The exported data is subsequently imported to one of the nodes in the cluster so that it is replicated across the other nodes in the cluster.

Work load is distributed by session, not by request. If a node in the cluster fails, users who are in session with that node must log back in to access the *Dashboard Application Service Hub*. Any unsaved work is not recovered.

Synchronized data

After load balancing is set up, changes in the console that are stored in global repositories are synchronized to all of the nodes in the cluster using a common database. The following actions cause changes to the global repositories used by the console. Most of these changes are caused by actions in the **Settings** folder in the console navigation.

- Creating, restoring, editing, or deleting a page.
- Creating, restoring, editing, or deleting a view.
- Creating, editing, or deleting a preference profile or deploying preference profiles from the command line.
- Copying a portlet entity or deleting a portlet copy.
- Changing access to a portlet entity, page, external URL, or view.

- Creating, editing, or deleting a role.
- Changes to portlet preferences or defaults.
- Changes from the **Settings** applications, including assigning users and groups to roles.

Note: Global repositories should never be updated manually.

During normal operation within a cluster, updates that require synchronization are first committed to the database. At the same time, the node that submits the update for the global repositories notifies all other nodes in the cluster about the change. As the nodes are notified, they get the updates from the database and commit the change to the local configuration.

If data fails to be committed on any given node, a warning message is logged into the log file. The node is prevented from making its own updates to the database. Restarting the Dashboard Application Service Hub Server instance on the node rectifies most synchronization issues, if not, the node should be removed from the cluster for corrective action. See "Monitoring a load balancing cluster" on page 314 for more information.

Note: If the database server restarts, all connections from it to the cluster are lost. It may take up to five minutes for connections to be restored, so that users can again perform update operations, for example, modifying or creating views or pages.

Manual synchronization and maintenance mode

Updates to deploy, redeploy, or remove console modules are not automatically synchronized within the cluster. These changes must be performed manually at each node. For deploy and redeploy operations, the console module package must be identical at each node.

When one of the deployment commands is started on the first node, the system enters *maintenance mode* and changes to the global repositories are locked. After you finish the deployment changes on each of the nodes, the system returns to an unlocked state. There is not any restriction to the order that modules are deployed, removed, or redeployed on each of the nodes.

While in maintenance mode, any attempts to make changes in the portal that affect the global repositories are prevented and an error message is returned. The only changes to global repositories that are allowed are changes to a user's personal portlet preferences. Any changes outside the control of the portal, for example, a form submission in a portlet to a remote application, are processed normally.

The following operations are also not synchronized within the cluster and must be performed manually at each node. These updates do not place the cluster in maintenance mode.

- Deploying, redeploying, and removing wires and transformations
- Customization changes to the console user interface (for example, custom images or style sheets) using `consoleProperties.xml`.

To reduce the chance that users could establish sessions with nodes that have different wire and transformation definitions or user interface customizations, schedule these changes to coincide with console module deployments.

Requirements

The following requirements must be met before load balancing can be enabled:

- If you are creating a cluster from a stand-alone instance of Dashboard Application Service Hub, you must export its data before you configure it for load balancing. Once you have configured the cluster, you can import the data to one of the nodes for it to be replicated across the other nodes.
- Lightweight Directory Access Protocol (LDAP) or OMNIbus ObjectServer must be installed and configured as the user repository for each node in the cluster. See Configuring LDAP user registries for instructions on how to enable LDAP for each node.
- A front-end network dispatcher (for example, IBM HTTP Server) must be setup to handle and distribute all incoming session requests. See Setting up intermediary services for more information about this task.

- DB2 Version 9.7 must be installed within the network to synchronize the global repositories for the console cluster.
- Each node in the cluster must be enabled to use the same LDAP using the same user and group configuration.
- All console nodes in load balancing cluster must be installed in the same cell name. After console installation on each node, use the **-cellName** parameter on the **manageprofiles** command.
- All console nodes in load balancing cluster must have synchronized clocks.
- The Websphere application server and Dashboard Application Service Hub Server versions must have the same release level, including any fix packs. Fixes and upgrades for the runtime must be applied manually at each node.
- Before joining nodes to a cluster, in each case make sure the node uses the same file-based repository user ID, which has been assigned the role of *iscadmins*.

Exporting data from a stand-alone server to prepare for load balancing

You can export date from an existing stand-alone application server instance to create a data file that can be imported to a load balanced cluster.

About this task

When you are creating a new load balanced cluster from a stand-alone instance, you must first export all data from the stand-alone instance and subsequently import the previously exported data once the cluster is set up.

Note: If you are joining the server to an existing cluster, the other nodes in the cluster should not contain custom data, that is, each node in the cluster should be clean installations. When you import data from the stand-alone server it is replicated across all other nodes.

Procedure

1. At the command line, change to the following directory:
   `/opt/IBM/JazzSM/profile/bin/`
2. Run the following command to export the stand-alone server's data:
   - Linux UNIX `restcli.sh export -username tbsmadmin -password tbsmadmin_password -destination data_file`
   - Windows `restcli.bat export -username tbsmadmin -password tbsmadmin_password -destination data_file`

   Where:

   tbsmadmin
   Specifies the administrator user ID.

   tbsmadmin_password
   Specifies the password associated with the administrator user ID.

   data_file
   Specifies the path and file name for the exported data, for example, `c:/tmp/data.zip`.
3. Create a new load balanced cluster using the stand-alone server, or join it to an existing cluster.
4. Import the previously exported data to any node in the cluster.
   a) At the command line, if necessary, change to the following directory:

   `/opt/IBM/JazzSM/profile/bin/`

   b) On one of the nodes in the cluster, run the following command to import the stand-alone server's data:

   `restcli.sh import -username tbsmadmin -password tbsmadmin_password -source data_file`

Where:

tbsmadmin
Specifies the administrator user ID.

tbsmadmin_password
Specifies the password associated with the administrator user ID.

data_file
Specifies the path and file name for the data to be imported, for example, `c:/tmp/data.zip`.

Results

Create a new load balanced cluster using the stand-alone application server, or join it to an existing cluster. Once the cluster is configured, you can import the data file to one of the nodes in the cluster.

What to do next

Setting up a load balancing cluster

You can configure a Dashboard Application Service Hub Server instance to use a database as a file repository instead of a local directory.

Before you begin

If you are creating a cluster from an existing Dashboard Application Service Hub Server instance that contains custom data, ensure that you have exported its data before you begin to configure it for load balancing. Once it is configured, you can import the data to one of the nodes in the new cluster.

Dashboard Application Service Hub is installed on a machine using the cell name designated for all console nodes within the cluster. You have installed and setup a network dispatcher (for example, IBM HTTP Server), DB2, and an LDAP as explained in "Requirements" on page 299.

Procedure

1. On the machine where DB2 is installed, create a DB2 database (see Creating databases).
2. Check that you have the JDBC driver for DB2 on the computer where Dashboard Application Service Hub is installed. The JDBC driver should be available at: `/opt/IBM/WebSphere/AppServer/universalDriver/lib`.
3. Configure load balancing, see https://www.ibm.com/support/knowledgecenter/SSSHTQ_8.1.0/com.ibm.netcool_OMNIbus.doc_8.1.0/webtop/wip/task/web_con_lb_configure.html.
4. In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
   - Windows `stopServer.bat server1`
   - Linux UNIX `stopServer.sh server1`

     Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.
5. Make sure your database is empty and the server is not started.

   Problems may occur if you try to setup load balancing on a non-empty database or active server.
6. From a command prompt, change to the `/opt/IBM/WebSphere/AppServer/bin` directory and issue this command:
   - Windows `..\ws_ant.bat -f install.ant configHA -Dusername=`*`DB2_username`* `-Dpassword=`*`DB2_password`*
   - Linux UNIX `../ws_ant.sh -f install.ant configHA -Dusername=`*`DB2_username`* `-Dpassword=`*`DB2_password`*

7. In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
   - Windows `startServer.bat server1`
   - Linux UNIX `startServer.sh server1`

Results
The load balancing cluster is created and the console node is joined to the cluster as the first node.

What to do next
Add (or join) additional nodes to the cluster.

Joining a node to a load balancing cluster
You can configure a Dashboard Application Service Hub Server to join an existing load balancing cluster.

Before you begin
1. If you are joining a stand-alone Dashboard Application Service Hub Server instance to a cluster, ensure that you first export all of its data. Once you have joined it to the cluster, you can then import the previously exported data. Other nodes in the cluster should not contain any custom data and should effectively be new installed instances.
2. Make sure you have successfully enabled load balancing following the steps in "Setting up a load balancing cluster" on page 301.
3. Dashboard Application Service Hub should be installed to the node using the same cell name that is designated for the cluster.
4. All console modules deployed to the cluster must be already deployed to the node that you intend to join.
5. You should deploy any wires or transformations used by the nodes in the cluster.
6. If the cluster is using any customization changes in `consoleProperties.xml` you must copy these changes and this file to the same location on the node that you intend to join.
7. The node must be configured to the same LDAP with the same user and group definitions as all other nodes in the cluster.

About this task
The following parameters are used on the `join` option when a node is added:

- **`-Dusername`** - specify the DB2 administrator's username
- **`-Dpassword`** - specify the DB2 administrator's password

Procedure
1. Check that you have the JDBC driver for DB2 on the computer where Dashboard Application Service Hub is installed. The JDBC driver should be available at: `/opt/IBM/WebSphere/AppServer/universalDriver/lib`.
2. Configure load balancing, see https://www.ibm.com/support/knowledgecenter/SSSHTQ_8.1.0/com.ibm.netcool_OMNIbus.doc_8.1.0/webtop/wip/task/web_con_lb_configure.html.
3. In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
   - Windows `stopServer.bat server1`
   - Linux UNIX `stopServer.sh server1`

Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

4. Make sure the Dashboard Application Service Hub Server is not started.
5. At a command prompt, change to the `/opt/IBM/WebSphere/AppServer/bin` directory and issue this command
   - Windows `..\ws_ant.bat -f install.ant configHA -Dusername=DB2_username -Dpassword=DB2_password`
   - Linux UNIX `../ws_ant.sh -f install.ant configHA -Dusername=DB2_username -Dpassword=DB2_password`
6. In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
   - Windows `startServer.bat server1`
   - Linux UNIX `startServer.sh server1`

Results

The console node is joined to the cluster.

What to do next

Add another node to the cluster, or if you have completed adding nodes, enable server to server trust for each node to every other node in the cluster.

Depending on the network dispatcher (for example, IBM HTTP Server) that you use, you might have further updates to get session requests routed to the new node. Refer to the documentation applicable to your network dispatcher for more information.

Enabling server-to-server trust

Use this procedure to enable load balanced nodes to connect to each other and send notifications.

About this task

These steps are required to enable load balancing between the participating nodes. Complete these steps on each node.

Procedure

1. In a text editor, open the `ssl.client.props` file from the `/opt/IBM//WebSphere/AppServer/profileTemplates/management/documents/properties/` directory.
2. Uncomment the section that starts with **`com.ibm.ssl.alias=AnotherSSLSettings`** so that it looks like this:

```
com.ibm.ssl.alias=AnotherSSLSettings
com.ibm.ssl.protocol=SSL_TLS
com.ibm.ssl.securityLevel=HIGH
com.ibm.ssl.trustManager=IbmX509
com.ibm.ssl.keyManager=IbmX509
com.ibm.ssl.contextProvider=IBMJSSE2
com.ibm.ssl.enableSignerExchangePrompt=true
#com.ibm.ssl.keyStoreClientAlias=default
#com.ibm.ssl.customTrustManagers=
#com.ibm.ssl.customKeyManager=
#com.ibm.ssl.dynamicSelectionInfo=
#com.ibm.ssl.enabledCipherSuites=
```

3. Uncomment the section that starts with **`com.ibm.ssl.trustStoreName=AnotherTrustStore`** so that it looks like this:

```
# TrustStore information
com.ibm.ssl.trustStoreName=AnotherTrustStore
com.ibm.ssl.trustStore=${user.root}/config/cells/JazzSMNode01Cell/nodes/
            JazzSMNode01/trust.p12
com.ibm.ssl.trustStorePassword={xor}CDo9Hgw=
com.ibm.ssl.trustStoreType=PKCS12
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=true
com.ibm.ssl.trustStoreReadOnly=false
```

4. Update the location of the trust store that the signer should be added to in the `com.ibm.ssl.trustStore` property of `AnotherTrustStore` by replacing the default value **`com.ibm.ssl.trustStore=${user.root}/etc/trust.p12`** with the correct path for your trust store. Example:

```
com.ibm.ssl.trustStore=${user.root}/config/cells/JazzSMNode01Cell/nodes/
                      JazzSMNode01/trust.p12
```

   After the update, the section must look like this:

```
com.ibm.ssl.trustStoreName=AnotherTrustStore
com.ibm.ssl.trustStore=${user.root}/config/cells/TIPCell/nodes/TIPNode/trust.p12
com.ibm.ssl.trustStorePassword={xor}CDo9Hgw=
com.ibm.ssl.trustStoreType=PKCS12
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=true
```

5. Save your changes to `ssl.client.props`.
6. Stop and restart the Dashboard Application Service Hub Server:
   a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
      - Windows `stopServer.bat server1`
      - Linux UNIX `stopServer.sh server1`

        Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

   b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
      - Windows `startServer.bat server1`
      - Linux UNIX `startServer.sh server1`
7. Complete all of the steps so far on each node before you continue with the rest of the steps.
8. Run the following command on each node for each *`myremotehost`* (that is, for every node that you want to enable trust with) in the cluster:

   Windows `C:\Program Files\IBM\JazzSM\profile\bin\retrieveSigners.bat NodeDefaultTrustStore AnotherTrustStore -host` *`myremotehost`* `-port` *`remote_SOAP_port`*

   Linux UNIX `/opt/IBM/JazzSM/profile/bin/retrieveSigners.sh NodeDefaultTrustStore AnotherTrustStore -host myremotehost -port` *`remote_SOAP_port`*

   where `myremotehost` is the name of the computer to enable trust with; `remote_SOAP_port` is the SOAP connector port number (16313 is the default). If you have installed with non-default ports, check `/opt/IBM/var/JazzSMProfile_portDef.properties` for the value of `SOAP_CONNECTOR_ADDRESS` and use that.
9. Stop and restart the Dashboard Application Service Hub Server:
   a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

- Windows `stopServer.bat server1`
- Linux UNIX `stopServer.sh server1`

  Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

- Windows `startServer.bat server1`
- Linux UNIX `startServer.sh server1`

Example

In this example, the load balancing cluster is comprised of two Microsoft Windows nodes named *myserver1* and *myserver2*. The command entered on *myserver1*:

```
retrieveSigners.bat NodeDefaultTrustStore AnotherTrustStore -host myserver2
 -port 16313
```

The command entered on *myserver2*:

```
retrieveSigners.bat NodeDefaultTrustStore AnotherTrustStore -host myserver1
 -port 16313
```

Verifying a load balancing implementation

Use the information in this topic to verify that your Dashboard Application Service Hub load balancing setup is working correctly once you have added all nodes to the cluster and enabled server-to-server trust.

About this task

This task allows you to confirm the following functions are working correctly:

- The database used for your load balancing cluster is properly created and initialized.
- Every node in the cluster uses the database as its repository instead of its own local file system.
- Server-to-server trust is properly enabled between nodes in the cluster.

To verify your load balancing configuration:

Procedure

1. Ensure that each Dashboard Application Service Hub Server instance on every node in the cluster is running.
2. In a browser, log into one node, create a new View and save your changes.
3. Log into the remaining nodes and verify that the newly created view is available in each one.

Preparing the HTTP server for load balancing

Install the IBM HTTP Server and configure the Web server plug-in for passing requests to the Dashboard Application Service Hub Server that are part of the load balancing configuration.

Before you begin

The IBM HTTP Server uses a Web server plug-in to forward HTTP requests to the Dashboard Application Service Hub Server. You can configure the HTTP server and the Web server plug-in to act as the load balancing server, that is, pass requests (HTTP or HTTPS) to one of any number of nodes. The load balancing methods supported by the plug-in are *round robin* and *random*:

- With a round robin configuration, when a browser connects to the HTTP server, it is directed to one of the configured nodes. When another browser connects, it is directed to a different node.
- With the random setting, each browser is connected randomly to a node. Once a connection is established between a browser and a particular node, that connection remains until the user logs out or the browser is closed.

The HTTP server is necessary for directing traffic from browsers to the applications that run in the *Dashboard Application Service Hub* environment. The server is installed between the portal and the Dashboard Application Service Hub Server, and is outside the firewall.

The Web server plug-in uses the `plugin-cfg.xml` configuration file to determine whether a request is for the application server.

About this task

Complete this procedure to configure the Web server plug-in for load balancing for each node.

Procedure

1. If you do not already have the IBM HTTP Server installed, install it before proceeding. It should be installed where it can be accessed from the Internet or Intranet (or both). Select the link at the end of this topic for the installation procedure.
2. Install IBM HTTP Server ensuring that you include the `IBM HTTP Server Plug-in for IBM WebSphere Application Server` option.

   For more information, see http://publib.boulder.ibm.com/infocenter/wasinfo/fep/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_installihs.html.
3. Create a new CMS-type key database.

   For more information see http://publib.boulder.ibm.com/infocenter/wasinfo/fep/index.jsp?topic=/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_createkeydb.html.
4. Create a self-signed certificate to allow SSL connections between nodes.

   For more information, see http://publib.boulder.ibm.com/infocenter/wasinfo/fep/index.jsp?topic=/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_certselfsigned.html.
5. To enable SSL communications for the IBM HTTP Server, in a text editor, open *HTTP_server_install_dir*`/conf/httpd.conf`. Locate the line `# End of example SSL configuration` and add the following lines, ensuring that the `KeyFile` line references the key database file created in step “3” on page 306 and save your changes.

   ```
   LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
   <IfModule mod_ibm_ssl.c>
     Listen 443
     <VirtualHost *:443>
       SSLEnable
     </VirtualHost>
   </IfModule>
   SSLDisable
   KeyFile "C:/Program Files/IBM/HTTPServer/bin/test.kdb"
   ```

   For more information, refer to the first example at http://publib.boulder.ibm.com/infocenter/wasinfo/fep/index.jsp?topic=/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_setupssl.html.
6. Restart the IBM HTTP Server.

   For more information, see http://publib.boulder.ibm.com/infocenter/wasinfo/fep/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_startihs.html.
7. On the IBM HTTP Server computer, to verify that SSL is enabled ensure that you can access `https://localhost`.
8. Stop and restart the Dashboard Application Service Hub Server:
   a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
      - Windows `stopServer.bat server1`

- Linux UNIX `stopServer.sh server1`

  Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

- Windows `startServer.bat server1`
- Linux UNIX `startServer.sh server1`

9. Start the HTTP server:
   a) Change to the directory where it is installed.
   b) Run this command: `bin/apachectl start`
   Note you must restart the server after changing the `plugin-cfg.xml` file.

What to do next

Enter the URL for the HTTP Server in a browser `http://`*`HTTP_server_host`*`/`*`HTTP_server_port`* and it will be forwarded to one of the nodes.

Note: The default load balancing method is random, whereby each browser is connected randomly to a node.

Setting clone IDs for nodes

Assign a clone ID for all nodes in the cluster.

About this task

Complete this procedure to set clone IDs for all nodes in the cluster. You must carry out these steps on each node.

Procedure

1. In a text editor, open the `server.xml` file from the `./JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/otherfiles/` directory
2. In `server.xml`, locate the entry `<components xmi:type="applicationserver.webcontainer:WebContainer`.
3. Within the `components` element, add the following entry:

```
<properties xmi:id="WebContainer_1183077764084" name="HttpSessionCloneId"
 value="12345" required="false"/>
```

Where:

`value` is the clone ID for the node, for example, `value="12345"`. The clone ID must be unique to each node. An example of an updated `components` element is provided here:

```
<components xmi:type="applicationserver.webcontainer:WebContainer"
xmi:id="WebContainer_1183077764084" enableServletCaching="false"
disablePooling="false">
      <stateManagement xmi:id="StateManageable_1183077764087"
initialState="START"/>
      <services xmi:type="applicationserver.webcontainer:SessionManager"
xmi:id="SessionManager_1183077764084" enable="true" enableUrlRewriting="false"
enableCookies="true" enableSSLTracking="false"
enableProtocolSwitchRewriting="false"
sessionPersistenceMode="NONE" enableSecurityIntegration="false"
allowSerializedSessionAccess="false" maxWaitTime="5"
accessSessionOnTimeout="true">
        <defaultCookieSettings xmi:id="Cookie_1183077764084" domain=""
maximumAge="-1" secure="false"/>
        <sessionDatabasePersistence
xmi:id="SessionDatabasePersistence_1183077764084"
datasourceJNDIName="jdbc/Sessions" userId="db2admin" password="{xor}Oz1tPjsyNjE="
```

```
db2RowSize="ROW_SIZE_4KB" tableSpaceName=""/>
        <tuningParams xmi:id="TuningParams_1183077764084"
usingMultiRowSchema="false" maxInMemorySessionCount="1000"
allowOverflow="true" scheduleInvalidation="false"
writeFrequency="TIME_BASED_WRITE" writeInterval="10"
writeContents="ONLY_UPDATED_ATTRIBUTES" invalidationTimeout="30">
          <invalidationSchedule xmi:id="InvalidationSchedule_1183077764084"
firstHour="14" secondHour="2"/>
        </tuningParams>
      </services>
      <properties xmi:id="WebContainer_1183077764084" name="HttpSessionCloneId"
value="12345" required="false"/>
    </components>
```

4. Save the changes you made to `server.xml`.

Generating the plugin-cfg.xml file

Run `GenPluginCfg.bat` to generate the `plugin-cfg.xml` file and save it in `/opt/IBM/JazzSM/profile/config/cells/JazzSMNode01Cell`.

About this task

Complete this procedure to generate the `plug-cfg.xml` file. You must carry out these steps on each node.

Procedure

1. On a node, change to `/opt/IBM/JazzSM/profile/bin` and run the following command:
   - Windows `GenPluginCfg.bat`
   - Linux UNIX `GenPluginCfg.sh`

   This command generates a file called `plugin-cfg.xml` and saves it to the `/opt/IBM/JazzSM/profile/config/cells/JazzSMNode01Cell` directory.
2. On the IBM HTTP Server, in the following directory, replace the existing `plugin-cfg.xml` with the version generated in step "1" on page 308:

   HTTP_web_server_install_dir`/plugins/config/webserver1`

   The following steps establish the new `/ibm/*` URI (Uniform Resource Identifier), which is where the plug-in will redirect requests:

   a) On the IBM HTTP Server, change to the directory where the Web server definition file is (such as `cd plugins/config/webserver1`).

   b) Open the `plugin-cfg.xml` file in a text editor, and in reference to the sample content extract provided below, edit the file to provide details of your IBM HTTP Server and all Dashboard Application Service Hub Server instances.

      HTTP SERVER PATH is the path to where the HTTP server is installed.
      HTTP SERVER PORT is the port for the HTTP server.
      SERVER1 is the fully qualified name of the computer where the application server is installed and started.
      SERVER2 is the fully qualified name of the computer where another application server is installed and started.
      CLONE_ID is the is the unique clone ID assigned to a particular node (server) in the cluster.

   c) In the `ServerCluster` section, the values for the `keyring` and `stashfile` properties should be ***HTTP SERVER PATH*** `/plug-ins/etc/plug-in-key.kdb` and ***HTTP SERVER PATH*** `/plug-ins/etc/plug-in-key.sth` respectively.

   d) Continue to add `Server` entries for any other nodes, following the same pattern. Add a new entry under `PrimaryServers` for each additional server.

   e) Add `CloneID` and `LoadBalanceWeight` attributes for every `Server` entry.

Important: For more information on web server plug-in workload management policies and to help you determine the appropriate values for the elements `LoadBalance` and `LoadBalanceWeight`, refer to the following articles:

- http://www.redbooks.ibm.com/abstracts/TIPS0235.html
- http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg21219567

Attention: The HTTP and HTTPS port values for all nodes should be the same.

```
<Config ASDisableNagle="false" IISDisableNagle="false"
IgnoreDNSFailures="false" RefreshInterval="60"
ResponseChunkSize="64" AcceptAllContent="false"
IISPluginPriority="High" FIPSEnable="false"
AppServerPortPreference="HostHeader" VHostMatchingCompat="false"
ChunkedResponse="false">
    <Log LogLevel="Trace" Name="HTTP SERVER PATH/Plugins/logs/webserver1/
http_plugin.log"/>
    <Property Name="ESIEnable" Value="true" />
    <Property Name="ESIMaxCacheSize" Value="1024" />
    <Property Name="ESIInvalidationMonitor" Value="false" />
    <Property Name="ESIEnableToPassCookies" Value="false" />
    <Property Name="PluginInstallRoot" Value="HTTP SERVER PATH/Plugins" />
    <VirtualHostGroup Name="default_host">
        <VirtualHost Name="*:16310" />
        <VirtualHost Name="*:80" />
        <VirtualHost Name="*:16311" />
        <VirtualHost Name="*:5060" />
        <VirtualHost Name="*:5061" />
        <VirtualHost Name="*:443" />
   <VirtualHost Name="*:HTTP SERVER PORT"/>
    </VirtualHostGroup>
 <ServerCluster CloneSeparatorChange="false" GetDWLMTable="false"
IgnoreAffinityRequests="true" LoadBalance="Round Robin"
Name="server1_Cluster" PostBufferSize="64" PostSizeLimit="-1"
RemoveSpecialHeaders="true" RetryInterval="60">
    <Server Name="TIPNode1_server1"
ConnectTimeout="0" CloneID="CLONE_ID" ExtendedHandshake="false"
ServerIOTimeout="0" LoadBalanceWeight="100" MaxConnections="-1"
WaitForContinue="false">
        <Transport Hostname="SERVER1" Port="16310"
Protocol="http"/>
     <Transport Hostname="SERVER1" Port="16311"
Protocol="https">
            <Property name="keyring" value="HTTP SERVER PATH\Plugins\config
\webserver1\plugin-key.kdb"/>
            <Property name="stashfile" value="HTTP SERVER PATH\Plugins\config
\webserver1\plugin-key.sth"/>
        </Transport>
        </Server>
<Server Name="TIPNode1_server2"
ConnectTimeout="0" CloneID="CLONE_ID" ExtendedHandshake="false"
ServerIOTimeout="0" LoadBalanceWeight="100" MaxConnections="-1"
WaitForContinue="false">
          <Transport Hostname="SERVER2" Port="16310"
Protocol="http"/>
       <Transport Hostname="SERVER2" Port="16311"
Protocol="https">
             <Property name="keyring" value="HTTP SERVER PATH\Plugins\config
\webserver1\plugin-key.kdb"/>
             <Property name="stashfile" value="HTTP SERVER PATH\Plugins\config
\webserver1\plugin-key.sth"/>
          </Transport>
          </Server>
        <PrimaryServers>
      <Server Name="TIPNode1_server1" />
           <Server Name="TIPNode1_server2" />
        </PrimaryServers>
    </ServerCluster>
    <UriGroup Name="server1_Cluster_URIs">
        <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/ivt/*" />
        <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/IBM_WS_SYS_RESPONSESERVLET/*" />
        <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/IBM_WS_SYS_RESPONSESERVLET/*.jsp" />
        <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/IBM_WS_SYS_RESPONSESERVLET/*.jsv" />
        <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
```

```
Name="/IBM_WS_SYS_RESPONSESERVLET/*.jsw" />
        <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/IBM_WS_SYS_RESPONSESERVLET/j_security_check" />
        <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/IBM_WS_SYS_RESPONSESERVLET/ibm_security_logout" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ibm/console/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ibm/help/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ibm/action/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ISCWire/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/isc/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ISCHA/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/tip_ISCAdminPortlet/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ISCAdminPortlets/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/mum/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ibm/TIPChangePasswd/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ibm/TIPExportImport/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ibm/tivoli/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/proxy/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/TIPWebWidget/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ibm/dbfile/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/ibm/TIPChartPortlet/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/TIPUtilPortlets/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/WIMPortlet/*" />
        <Uri AffinityCookie="JSESSIONID_ibm_console_16310"
AffinityURLIdentifier="jsessionid" Name="/SysMgmtCommonTaskGroups/*" />
    </UriGroup>
    <Route ServerCluster="server1_Cluster" UriGroup="server1_Cluster_URIs"
VirtualHostGroup="default_host" />
    <RequestMetrics armEnabled="false" newBehavior="false" rmEnabled="false"
traceLevel="HOPS">
        <filters enable="false" type="URI">
            <filterValues enable="false" value="/snoop" />
            <filterValues enable="false" value="/hitcount" />
        </filters>
            <filters enable="false" type="SOURCE_IP">
            <filterValues enable="false" value="255.255.255.255" />
            <filterValues enable="false" value="254.254.254.254" />
        </filters>
        <filters enable="false" type="JMS">
            <filterValues enable="false" value="destination=aaa" />
        </filters>
        <filters enable="false" type="WEB_SERVICES">
            <filterValues enable="false" value="wsdlPort=aaa:op=bbb:nameSpace=ccc" />
        </filters>
    </RequestMetrics>
</Config>
```

Configuring SSL from each node to the IBM HTTP Server

For load balancing implementations, you must configure SSL between the IBM HTTP Server plug-in and each node in the cluster.

Before you begin

This task assumes that you have already installed and configured the IBM HTTP Server for load balancing.

About this task

For each node in the cluster, follow these instructions to configure the node to communicate over a secure (SSL) channel with the IBM HTTP Server.

Procedure

1. Log in to the *Dashboard Application Service Hub*.
2. In the navigation pane, click **Settings** > **Websphere Administrative Console** and click **Launch Websphere administrative console**.
3. Follow these steps to extract signer certificate from the trust store:
   a) In the WebSphere Application Server administrative console navigation pane, click **Security** > **SSL certificate and key management**.
   b) In the Related Items area, click the **Key stores and certificates** link and in the table click the **NodeDefaultTrustStore** link.
   c) In the Additional Properties area, click the **Signer certificates** link and in the table that is displayed, select the `root` entry check box.
   d) Click **Extract** and in the page that is displayed, in the **File name** field, enter a certificate file name (*`certficate`*`.arm`), for example, `c:\tivpc064ha1.arm`.
   e) From the **Data Type** list select the **Base64-encoded ASCII data** option and click **OK**.
   f) Locate the extracted signer certificate and copy it to the computer running the IBM HTTP Server.

   Note: This steps are particular to Dashboard Application Service Hub, for general WebSphere Application Server details and further information, see: http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.doc/info/aes/ae/tsec_sslextractsigncert.html
4. On the computer running the IBM HTTP Server, follow these steps to import the extracted signer certificate into the key database:
   a) Start the key management utility (iKeyman), if it is not already running, from *HTTP_SERVER_PATH*`/bin`:
      - Linux UNIX At the command line, enter `./ikeyman.sh`
      - Windows At the command line, enter `ikeyman.exe`
   b) Open the CMS key database file that is specified in plugin-cfg.xml, for example, *HTTP_SERVER_PATH*`/plug-ins/etc/plug-in-key.kdb`.
   c) Provide the password (default is `WebAS`) for the key database and click **OK**.
   d) From the **Key database content**, select **Signer Certificates**.
   e) Click **Add** and select the signer certificate that you copied from the node to the computer running the IBM HTTP Server and click **OK**.
   f) Select the **Stash password to a file** check box and click **OK** to save the key database file.

   Note: For more information on certificates in WebSphere Application Server, see: http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_ikeyscca.html
5. Repeat these steps for each node in the cluster.
6. For the changes to take effect, stop and restart all nodes in the cluster and also restart the computer running the IBM HTTP Server.
   a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
      - Windows `stopServer.bat server1`
      - Linux UNIX `stopServer.sh server1`

Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

- Windows `startServer.bat server1`
- Linux UNIX `startServer.sh server1`

c) Restart the IBM HTTP Server.

For more information, see http://publib.boulder.ibm.com/infocenter/wasinfo/fep/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_startihs.html.

What to do next

You should now be able to access the load balanced cluster through `https://`*`http_server_hostname`*`/ibm/console` (assuming that the default context root (`/ibm/console`) was defined in at the time of installation.

Importing stand-alone instance data to a cluster

If you created a cluster from a stand-alone application server instance, you can then import the data that you exported prior to configuring the stand-alone instance as a cluster node.

About this task

Import the previously exported data file to any node in the cluster.

Important: The instructions in this topic apply only to importing data that was exported when preparing to create a load balanced cluster from a stand-alone application server instance, as described in "Exporting data from a stand-alone server to prepare for load balancing" on page 300.

Procedure

1. At the command line, change to the following directory:

   `/opt/IBM/JazzSM/profile/bin`

2. On one of the nodes in the cluster (most likely the node that was previously set up as a stand-alone server instance), run the following command to import the data file:

   - Linux UNIX `restcli.sh import -username` *`tbsmadmin`* `-password` *`tbsmadmin_password`* `-source` *`data_file`*
   - Windows `restcli.bat import -username` *`tbsmadmin`* `-password` *`tbsmadmin_password`* `-source` *`data_file`*

   Where:

   tbsmadmin
   Specifies the administrator user ID.

   tbsmadmin_password
   Specifies the password associated with the administrator user ID.

   data_file
   Specifies the path and file name to the data file that is to be imported, for example, `c:/tmp/data.zip`.

Results

The data from the initial application server is imported to the node and replicated across the other cluster nodes.

Removing a node
Follow these steps to remove a node from the load balancing cluster.

About this task
The following parameters are used on the `disjoin` option when a node is removed.

- **`-Dusername`** - specify the DB2 administrator's username
- **`-Dpassword`** - specify the DB2 administrator's password

Procedure
1. From a command prompt, change to the `/opt/IBM/WebSphere/AppServer/bin` directory and issue this command:
   - Windows `..\ws_ant.bat -f uninstall.ant disjoin -Dusername=`*`DB2_username`* `-Dpassword=`*`DB2password`*
   - Linux UNIX `../ws_ant.sh -f uninstall.ant disjoin -Dusername=`*`DB2_username`* `-Dpassword=`*`DB2password`*
2. Update the network dispatcher (for example, IBM HTTP Server) to remove the node from the configuration.

Removing a remote node

About this task
This command should be used only in the rare occasions where physical access to the node is not available or a serious hardware or software failure has occurred. If the node is remotely disjoined but continues to function, some problems with synchronization might arise that can lead to problems with data consistency and synchronization.

Procedure
1. From a command prompt, change to the `/opt/IBM/WebSphere/AppServer/bin` directory and issue this command:
   - Windows `..\ws_ant.bat -f uninstall.ant remote-disjoin –DremoteHost=`*`remote_host`* `–DremotePort=9044 -Dusername=`*`DB2_username`* `-Dpassword=`*`DB2_password`*
   - Linux UNIX `../ws_ant.sh -f uninstall.ant remote-disjoin –DremoteHost=`*`remote_host`* `–DremotePort=9044 -Dusername=`*`DB2_username`* `-Dpassword=`*`DB2_password`*
2. Update the network dispatcher (for example, IBM HTTP Server) to remove the node from the configuration.

Removing a load balancing cluster
Follow these steps to remove the last node from a cluster and thereby the cluster itself.

Before you begin
Make sure you have removed all other nodes from the cluster. This command should be issued from the last active node remaining in the cluster.

About this task
The following parameters are used on the uninstall option when the node is removed.

- **`-Dusername`** - specify the DB2 administrator's username

- **-Dpassword** - specify the DB2 administrator's password

Procedure

From a command prompt, change to the /opt/IBM/WebSphere/AppServer/bin directory and issue this command:

- Windows ..\ws_ant.bat -f uninstall.ant uninstall -Dusername=*DB2_username* -Dpassword=*DB2_password*
- Linux UNIX ../ws_ant.sh -f uninstall.ant uninstall -Dusername=*DB2_username* -Dpassword=*DB2_password*

Monitoring a load balancing cluster

If synchronized data fails to be committed to a node in the cluster, that node should be removed from the cluster for corrective action. Use the diagnosis tool to identify any unsynchronized nodes in the load balancing cluster.

To determine if changes to global data are not committed to any of the nodes, use the **HATool** command script to check the synchronization of modules and repositories on the nodes in a cluster. For the HATool, you must provide the DB2 administrator's credentials.

Query synchronization of modules

Use this command to determine if all nodes have identical sets of modules deployed.

```
HATool.bat/sh modules username password -byNodes -showAll
```

The following parameters are optional.

- **-byNodes**

  Specifies that the results of the command are ordered by the node in the cluster. This parameter is optional. The default is to list the results by module.

- **-showAll**

  Specifies that all modules and nodes in the cluster should be returned. This parameter is optional. The default is to return only modules for unsynchronized nodes.

Query the synchronization of global repositories

Use this command to determine if all repositories are synchronized on all nodes.

```
HATool.bat/sh repositories username password -byNodes -showAll
```

The following parameters are optional.

- **-byNodes**

  Specifies that the results of the command are ordered by the node in the cluster. This parameter is optional. The default is to list the results by repository.

- **-showAll**

  Specifies that all repositories and nodes in the cluster should be returned. This parameter is optional. The default is to return only repositories for unsynchronized nodes.

Release the global lock

Use this command to manually release the global lock placed on all of the console nodes when the cluster is in maintenance mode. This command is used when a node cannot commit a change during synchronization and has to be taken offline.

```
HATool.bat/sh release-lock username password
```

Configuring Tivoli Access Manager in Dashboard Application Service Hub

You can configure Dashboard Application Service Hub to use Tivoli Access Manager WebSEAL Version 6.1 to manage authentication.

You must install and configure Tivoli Access Manager WebSEAL Version 6.1. To set up and configure Tivoli Access Manager WebSEAL, see http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itame.doc/am611_install196.htm#webseal.

For more information on administering Tivoli Access Manager WebSEAL, see http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itame.doc/am611_webseal_admin.htm.

Attention: The IBM Tivoli Netcool Impact user interface contained within IBM Tivoli Business Service Manager may not function within the Tivoli Access Manager WebSEAL environment. If you need to access the Netcool/ Impact user interface (for example, to edit Impact policies or access Operator Views), you must do so outside of the Tivoli Access Manager WebSEAL environment.

Configuring single sign-on using ETai

In a WebSphere Application Server (WAS) environment, Tivoli Access Manager WebSEAL can be used as a reverse proxy to intercept incoming http or https requests to ensure that users are authenticated and authorized and are passed to the relevant Dashboard Application Service Hub Server .

ETai is the component that implements the WebSphere Application Server trust association interceptor interface to achieve single sign on from WebSEAL to the Dashboard Application Service Hub Server.

Dashboard Application Service Hub supports single sign-on (SSO) with perimeter authentication services such as reverse proxies through trust associations. When trust associations are enabled, the WebSphere Application Server is not required to authenticate a user if a request arrives from a trusted source that has already performed authentication.

Once a trust association is configured between WebSEAL and the Dashboard Application Service Hub Server, a user can login into Tivoli Access Manager and then access the Dashboard Application Service Hub Server without having to re-authenticate. The ETai must be configured in Dashboard Application Service Hub Server server and is responsible for establishing trust against the WebSEAL server. ETai simplifies the use of Tivoli Access Manager and the configuration required to achieve SSO. One advantage is that Tivoli Access Manager and Dashboard Application Service Hub can use different user registries and still be able to perform SSO. It also provides the mapping between different registry formats.

Installing ETai

Use these instructions, to install the Tivoli Access Manager Extended Trust Association Interceptor in a Dashboard Application Service Hub environment.

Before you begin

Source a copy of `com.ibm.sec.authn.tai.etai_6.0.jar` from your installation media.

About this task

To install ETai:

Procedure

1. Copy `com.ibm.sec.authn.tai.etai_6.0.jar` to the `plugins` directory.
2. At the command line, depending on your operating system, run the relevant command:
   - Windows `C:\Program Files\IBM\JazzSM\profile\bin\Osgicfginit.bat`
   - Linux UNIX `/opt/IBM/JazzSM/profile/bin/Osgicfginit.sh`
3. Copy `pd.jar` to `/opt/IBM/WebSphere/AppServer/java/jre/lib/ext`

What to do next

Configure ETai in a Dashboard Application Service Hub environment.

Enabling a trust association for ETai

You must enable a trust association between the Tivoli Access Manager Extended Trust Association Interceptor in the Dashboard Application Service Hub environment.

About this task

To configure a trust association for ETai:

Procedure

1. Log in to the portal and click **Settings** > **WebSphere Administrative Console**.
2. In the **WebSphere Administrative Console** page, click **Launch WebSphere administrative console**.
3. In the **WebSphere Administrative Console** navigation pane, click **Global security**.
4. In the **Global security** page, expand **Web security** and click **Trust association**.
5. In the General Properties area, click the **Enable trust association** option if it is disabled and click **Apply**.

   Your update is saved and you are returned to the **Global security** page.
6. In the **Global security** page, expand **Web security** and click **Trust association** to display the **Trust association** page.
7. In the Additional properties area, click the **Interceptors** link to display the **Interceptors** page.
8. If `com.ibm.sec.authn.tai.TAMETai` is not listed on the page, click **New**.
9. In the **Interceptor class name** field enter the string `com.ibm.sec.authn.tai.TAMETai` and click **Apply**.
10. In the Messages area, click the **Save** link to commit your change.

What to do next

Configure ETai in the a Dashboard Application Service Hub environment.

Configuring custom properties for ETai

Once you have enabled a trust association for the Tivoli Access Manager Extended Trust Association Interceptor in the Dashboard Application Service Hub environment, you must configure its custom properties.

About this task

To configure custom properties for the ETai:

Procedure

1. Log in to the portal and click **Settings** > **WebSphere Administrative Console**.
2. In the **WebSphere Administrative Console** page, click **Launch WebSphere administrative console**.
3. In the **WebSphere Administrative Console** navigation pane, click **Global security**.
4. In the **Global security** page, expand **Web security** and click **Trust association** to display the **Trust association** page.
5. In the Additional properties area, click the **Interceptors** link to display the **Interceptors** page.
6. From the list of interceptor classes, select the `com.ibm.sec.authn.tai.TAMETai` entry.
7. In the Additional properties area, click the **Custom properties** link to display the **Custom properties** page.
8. Review the details for the custom properties listed in the following table:

Table 157. ETai custom properties

Property details	Notes

Property name:
`com.ibm.websphere.security.webseal.useWebSphereUserRegistry`
Type:
string
Required:
Yes
Values:
`true` or `false`
Default value:
`true`
	ETai authenticates the trusted user against the WebSphere Application Server user registry or the Tivoli Access Manager Authorization Server. If this property is set to `true`, the resulting `Subject` will not contain a `PDPrincipal` as the Tivoli Access Manager Authorization Server is required to build the `PDPrincipal`. Any other value for this property will result in a `PDPrincipal` being added to the `Subject`.

Property name:
`com.ibm.websphere.security.webseal.tamUserDnMapping`
Required:
Yes
Value:
`WAS`
Default value:
`TAM`
	The ETai adds users' credential information into the JAAS `Subject`. This information includes the users dn. Maps this dn to the WebSphere Application Server dn, or (Value = `WAS`). If a mapping is attempted for a user that does not exist in the WebSphere Application Server user registry, it is ignored and not added to the JAAS `Subject`.

Property name:
`com.ibm.websphere.security.webseal.tamGroupDnMapping`
Required:
Yes
Value:
`WAS`
Default value:
`TAM`
	The ETai adds users' credential information into the JAAS `Subject`. This information includes the group dn's. The ETai can be configured to either:
	Map these dn's to the WebSphere Application Server dn's, or (Value = `WAS`).
	If a mapping is attempted for a group that does not exist in the WebSphere Application Server user registry, it is ignored and not added to the JAAS `Subject`.

Table 157. ETai custom properties (continued)

Property details	Notes

Property name:
`com.ibm.websphere.security.webseal.loginId`
Type:
String
Required:
Yes
Value:
`websealSSOID`
Default value:
None
	The value of this property must exist as a valid user in the user registry.
	If necessary, create a new user in the Dashboard Application Service Hub registry called `websealSSOID`.
	The ETai must be configured with the username of the WebSEAL trusted user. This is the single sign-on user that is authenticated using the password in the Basic Authentication header inserted by WebSEAL in the request. The format of the username is the short name representation.
	This property interacts with the following property:
	`com.ibm.websphere.security.webseal.useWebSphereUserRegistry`
	If `com.ibm.websphere.security.webseal.useWebSphereUserRegistry` is set to `true` then the specified user must exist in either the WebSphere Application Server user registry or the Tivoli Access Manager user registry.

Property name:
`com.ibm.websphere.security.webseal.checkViaHeader`
Type:
String
Required:
Yes
Value:
`true`
Default value:
`false`
	The ETai can be configured so that the `Via header` can be ignored when validating trust for a request. This property is required, if WebSEAL is to allow requests into the Dashboard Application Service Hub only from particular hosts.
	This property interacts with the following properties:
	• `com.ibm.websphere.security.webseal.hostnames`
	• `com.ibm.websphere.security.webseal.ports`
	If `com.ibm.websphere.security.webseal.checkViaHeader` is set to `false` then the values set for the two associated properties are not used.

Property name:
`com.ibm.websphere.security.webseal.id`
Required:
Yes
Value:
`iv-creds`
Default value:
`iv-creds`
	`Iv-creds` carrys end user credentials, which is used by Dashboard Application Service Hub for authorization.
	Note: Any additional values set for this property are added to a list along with `Iv-creds`, that is, `Iv-creds` is a required header for the ETai.

Table 157. ETai custom properties (continued)

Property details	Notes

Property name:
`com.ibm.websphere.security.webseal.hostnames`
Required:
Yes
Value:
A comma separated list of strings.
Default value:
There is no default value for this property.
	The ETai can be configured so that the request must arrive from a list of expected hosts. If any of the hosts in the `Via header` of the HTTP request are not listed in the values set for this property, the request is ignored by the ETai.
	This property interacts with the following property:
	`com.ibm.websphere.security.webseal.ports`
	All of the values listed for `com.ibm.websphere.security.webseal.hostnames` are used with the ports listed for `com.ibm.websphere.security.webseal.ports` to indicate a trusted host.
	For example, if:
	`com.ibm.websphere.security.webseal.hostnames` is set to `abc,xyz`
	`com.ibm.websphere.security.webseal.ports` is set to `80,443`
	Then, the `Via header` is checked for these hostname/port combinations: `abc:80`; `abc:443`; `xyz:80`; `xyz:443`.
	If `com.ibm.websphere.security.webseal.checkViaHeader` is set to `false` then the values set for `com.ibm.websphere.security.webseal.hostnames` are not used.

Property name: `com.ibm.websphere.security.webseal.ports` **Required:** Yes **Value:** 443 **Default value:** There is no default value for this property.	This property interacts with the following property: `com.ibm.websphere.security.webseal.hostnames` All of the values listed for `com.ibm.websphere.security.webseal.hostnames` are used with the ports listed for `com.ibm.websphere.security.webseal.ports` to indicate a trusted host. For more information, see the notes for `com.ibm.websphere.security.webseal.hostnames`.

Table 157. ETai custom properties (continued)	
Property details	**Notes**

Property name:
`com.ibm.websphere.security.webseal.ssoPwdExpiry`
Required:
No
Value:
A positive integer.
Default value:
`600`
	Once trust has been established for a request, the password for the Single sign-on user is cached for subsequent trust validation of requests. This saves the ETai from having to re-authenticate the single sign-on user with the user registry for every request, therefore increasing performance. The cache timeout period can be modified by setting this property to the required time in seconds. If the password expiry property is set to `0`, the cached password does not expire.

Property name:
`com.ibm.websphere.security.webseal.groupRealmPrefix`
Required:
Yes
Value:
`“group:”`
Default value:
`"group:"`
	This property is needed to map the group realm prefix from Tivoli Access Manager to group realm prefix in WebSphere Application Server registry.

Property name:
`com.ibm.websphere.security.webseal.userRealmPrefix`
Required:
Yes
Value:
`“user:”`
Default value:
`"user:"`
	This property is needed to map the user realm prefix from Tivoli Access Manager to user realm prefix in WebSphere Application Server registry.

9. If a custom property does not exist, click **New** to configure a custom property and provide a name, value, and optional description and click **Apply** to add the custom property.
10. If the custom property exists, but is not in line with the details provided in the table above, click on the custom property entry, update its details and click **Apply** to modify the custom property.
11. Stop and restart the Dashboard Application Service Hub Server:
    a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
    - Windows `stopServer.bat server1`
    - Linux UNIX `stopServer.sh server1`

      Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

    b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
    - Windows `startServer.bat server1`

- Linux UNIX `startServer.sh server1`

What to do next

Configure the Tivoli Access Manager WebSEAL by creating a WebSEAL junction and creating a junction mapping table.

Checking your Tivoli Access Manager configuration

To ensure that your Tivoli Access Manager configuration is valid, you can carry out a number of checks.

Before you begin

Ensure that you have the following software versions installed:

- Tivoli Access Manager version 6.1
- Dashboard Application Service Hub Server, version 1.1 fix pack 11 or later

About this task

This topic describes how to check the following items:

- The status of the Tivoli Access Manager server.
- Connecting to the Dashboard Application Service Hub Server.

Procedure

1. To check the status of the Tivoli Access Manager server, at the command line, enter `pd start status`.

   The following output indicates that the Tivoli Access Manager server is running:

   ```
   pdmgrd        yes    yes
   pdacld        yes    no (sometimes yes)
   pdmgrproxyd        no        no
   webseald-ip1    yes    yes
   ```

2. To check if the Lightweight Directory Access Protocol (LDAP) user registry is active:
   a) At the command line, enter `pdadmin -a sec_master -p` *`sec_master_password`*.

      Note: This command assumes that `pdadmin` is in the path.

      Expected output:

      ```
      pdadmin -a sec_master -p sec_master_password
      ```

   b) At the command line, enter `user list * 10`.

      Example output:

      ```
      sec_master
      ivmgrd/master
      ivacld/ip1
      ip1-webseald/ip1
      ```

   c) To quit, at the command line, enter `quit`.
3. If the Tivoli Access Manager processes are not started, at the command line enter `pd start start`.

   If the processes are already started, the following output can be expected:

   ```
   Starting the: Access Manager authorization server
   Could not start the server
   ```

4. To check that you can connect from the Dashboard Application Service Hub Server to the Tivoli Access Manager computer:

a) On the Dashboard Application Service Hub Server use a Web browser to connect to `http://tam_server_hostname`. A security message may be displayed, confirm the Tivoli Access Manager self-signed certificate to display an authorization dialog.

b) Enter a username and password to display the Tivoli Access Manager WebSEAL splash screen (username = `sec_master`, password = *sec_master_password*).

What to do next

Configure the WebSEAL keystore.

Configuring the WebSEAL keystore

To allow the application server to use Tivoli Access Manager WebSEAL, you must import Dashboard Application Service Hub Server security certificate to the WebSEAL keystore.

About this task

To export the Dashboard Application Service Hub Server security certificate and import it into the WebSEAL keystore:

Procedure

1. Log in to the Dashboard Application Service Hub console.
2. Export the Dashboard Application Service Hub X.509 certificate.

   The process for exporting varies depending on your browser. Refer to your browser documentation for assistance.

   For example, the following substeps describe how you can export the certificate using a Firefox browser:

   a) Double-click on lock icon that appears in the browser window to display the **Security** dialog for the Web page.

   b) Click **View Certificate** and in the **Certificate Viewer** dialog and then click the **Details** tab.

   c) Click **Export** and in the **Save Certificate To File** dialog and select a directory to export the Dashboard Application Service Hub X.509 certificate.
3. Copy the exported certificate file to the Tivoli Access Manager computer.
4. On the Tivoli Access Manager computer, at the command line, change to the directory that hosts the IKeyman utility.

   For example, the following directories reflect typical locations for the IKeyman utility, but it may vary depending on your environment:

   - Linux UNIX *WAS_home*`/profiles/`*profile_name*`/bin/`
   - Windows *WAS_home*`\java\jre\bin\`
5. Start the IKeyman utility and complete the substeps:

   - Linux UNIX At the command line, enter `./ikeyman.sh`
   - Windows At the command line, enter `ikeyman.exe`

   a) On the toolbar, click **Open** to display the **Open** window.

   b) Select CMS as the key database type.

   c) Click **Browse** and from `/var/pdweb/www-ip1/certs`, select `pdsrv.kdb` to display the **Password Prompt** dialog.

   The default password reflects the file name, that is, `pdsrv`.

   d) In the Key database content section, select **Signer Certificates** and click **Add**.

   e) In the **Add CA's Certificate from a File** dialog, for the **Data type**, select the `Base64-encoded ASCII data` option and click **Browse**.

f) Locate the Dashboard Application Service Hub X.509 certificate and enter a label for the certificate (for example, `tipmachine`).

g) Click **Save** to add the certificate to the WebSEAL keystore (do not change the certificate's file name).

6. To restart Tivoli Access Manager WebSEAL, at the command line, enter `pdweb restart`.

   The following is the expected output:

```
            Stopping the: webseald-ip1
            Starting the: webseald-ip1
```

What to do next

Create a WebSEAL junction.

Creating a WebSEAL junction

A WebSEAL junction is an HTTP or HTTPS connection between a front-end WebSEAL server and a back-end Web application server, for example the Dashboard Application Service Hub Server.

About this task

Junctions logically combine the Web space of the back-end server with the Web space of the WebSEAL server, resulting in a unified view of the entire Web object space. To create a junction:

Procedure

1. On the Tivoli Access Manager computer, at the command line, enter `pdadmin -a` *`sec_master_account`* `-p` *`sec_master_password`*.
2. At the command line, enter `s l`.

   The following is the expected output:

```
   ivacld-ip1
   ip1-webseald-ip1
```

   Note: Where `ip1` is the hostname of the Tivoli Access Manager computer.

3. Enter `s t ip1-webseald-ip1 list`.

   The following is the expected output:

```
   /
```

4. Enter `s t ip1-webseald-ip1 create -t ssl -c iv-creds -b supply -h` *`tbsm_hostname/ip`* `-p` *`tbsmadmin_console_secure_port`* `/tip`.

   Where:

```
   s t = server task
   ip1-webseal-ip1 = WebSEAL instance name
   -t ssl = transport type is SSL
   -c iv-creds = needed for single sign on (SSO) to work, carry credential
of user
   -b supply = basic authorization header needed for SSO to work
```

   The following is the expected output:

```
   Created junction at /tip
```

   Note: If you want to delete a junction, enter `s t ip1-webseald-ip1 delete /tip`.

   Note: If you want to show details for a junction, enter `s t ip1-webseald-ip1 show /tip`.

What to do next

Create a WebSEAL junction mapping table.

Creating a WebSEAL junction mapping table

A junction mapping table maps specific target resources to junction names. Junction mapping is an alternative to a cookie-based solution for filtering dynamically generated server-relative URLs.

About this task

To create a WebSEAL junction mapping table:

Procedure

1. On the Tivoli Access Manager computer, in a text editor open the WebSEAL configuration file, `/opt/pdweb/etc/webseald-ip1.conf`.
2. In the `[junction]` section, edit the `jmt-map` path so that it reads `jmt-map = lib/jmt.conf`.

   Note: This path is relative to the server root path. Check the server root path in the `[server]` section of the file and take a note of the full `jmt-map` path. For example, `/opt/pdweb/www-ip1/lib/jmt.conf`.
3. In a text editor create or edit open the `jmt.conf` file and add or modify the following:
   - `/tip /ibm/console/*`

     Note: The `/ibm/console/` element of the path shown assumes that the Dashboard Application Service Hub root context path was not reconfigured at installation time.
   - `/tip /ibm/sla/*`
4. To load the `jmt.conf` file into WebSEAL, enter `s t ip1-webseald-ip1 jmt load`.

   The following is the expected output:

   ```
   DPWWM1462I    JMT Table successfully loaded
   ```
5. To restart the WebSEAL server, enter `pdweb restart`.

   The following is the expected output:

   ```
   Stopping the: webseald-ip1
   Starting the: webseald-ip1
   ```

What to do next

Test the WebSEAL junction.

Testing the WebSEAL junction

Once you have created a WebSEAL junction, you can test it.

About this task

To test a WebSEAL junction:

Procedure

1. In your Web browser's address bar, enter `https://`*`tam_server_hostname`*`/`*`tip`*`/ibm/console`, where *`tip`* is the name of the WebSEAL junction.

   The Dashboard Application Service Hub login page is displayed.
2. To test if Tivoli Access Manager challenges you when you try to access the Dashboard Application Service Hub:

   a) Close all instances of your Web browser.

   b) Start your Web browser and go to `https://`*`tam_server_hostname`*`/tip/ibm/console/`.

Note: The `/ibm/console/` element of the URL shown assumes that the Dashboard Application Service Hub root context path was not reconfigured at installation time.

If the WebSEAL junction is working as expected, an **Authentication Required** dialog is displayed and you have to provide Tivoli Access Manager account (`sec_master`) details to proceed.

What to do next

Edit `customizationProperties.xml` to ensure that when you log out of Dashboard Application Service Hub that you also log out from Tivoli Access Manager.

Configuring single sign off for Tivoli Access Manager and Dashboard Application Service Hub

To ensure that you when you log out from the Dashboard Application Service Hub that you also log out from Tivoli Access Manager, you must edit `customizationProperties.xml`.

About this task

To configure single sign off for the Dashboard Application Service Hub Server and the Tivoli Access Manager computer:

Procedure

1. In a text editor, open `/opt/IBM/JazzSM/profile/config/cells/JazzSMNode01Cell/applications/isc.ear/deployments/isc/isclite.war/WEB-INF/customizationProperties.xml`.

   Windows For example: `C:\IBM\tivoli\tipv2C:\program Files\IBM\JazzSM\profile\config\cells\JazzSMNode01Cell\applications\isc.ear\deployments\isc\isclite.war\WEB-INF\customizationProperties.xml`
2. Edit the `TAMJunctionName` property, as follows:

   ```
   <consoleproperties:console-property id="TAMJunctionName" value="tip"/>
   ```

   ```
   <consoleproperties:console-property id="WebSealServerName" value=""/>
   ```

   Where:

   - `TAMJunctionName` is the junction name in Tivoli Access Manager that is configured to point at the Dashboard Application Service Hub Server.
   - `WebSealServerName` is a Tivoli Access Manager WebSEAL server instance name. This property allows the Dashboard Application Service Hub Server process requests from declared WebSEAL hosts.

Results

When you log out from the Dashboard Application Service Hub, a `Successful Logout` message is displayed in your browser. This indicates that you logged out from both the Dashboard Application Service Hub and Tivoli Access Manager.

Setting form-based authentication for WebSEAL

Tivoli Access Manager provides form-based authentication as an optional alternative to the standard Basic Authentication mechanism.

About this task

For information on WebSEAL authentication and changing from basic mode to the form-based mode refer to Tivoli Access Manager documentation at http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itame.doc_6.1/am61_webservers_admin74.htm#chpt4_amwebpi_authent:

Configuring access for HTTP and HTTPS

By default, the application server requires HTTPS (Hypertext Transfer Protocol Secure) access. If you want some users to be able to log in and use the console with no encryption of transferred data, including user ID and password, configure the environment to support both HTTP and HTTPS modes.

Before you begin

After installing *Dashboard Application Service Hub* and before beginning this procedure, log in to the portal to ensure that it has connectivity and can start successfully.

About this task

Configuring for HTTP and HTTPS console access involves editing the `web.xml` file of Web components. Use this procedure to identify and edit the appropriate Web XML files.

Procedure

1. Change to the following directory: `/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/`.
2. From this location, locate the `web.xml` files in the following directories:
   - For the Integrated Solutions Console web application archive: `isc.ear/deployments/isc/isclite.war/WEB-INF`
   - For the Dashboard Application Service Hub Charts web application archive: `isc.ear/deployments/isc/tip.charts.war/WEB-INF/`
   - For the Dashboard Application Service Hub Change Password web application archive: `isc.ear/deployments/isc/TIPChangePasswd.war/WEB-INF/`
   - Additionally for IBM Tivoli Business Service Manager, the `web.xml` files in the following directories must be updated to allow the use of HTTP:
     - `isc.ear/deployments/isc/sla.war/WEB-INF/`
     - `isc.ear/deployments/isc/twa.war/WEB-INF/`
     - `isc.ear/deployments/isc/impactAdmin.war/WEB-INF/`
3. Open one of the `web.xml` files using a text editor.
4. Find the `<transport-guarantee>` element.

   The initial value of all `<transport-guarantee>` elements is `CONFIDENTIAL`, meaning that secure access is always required.
5. Change the setting to `NONE` to enable both HTTP and HTTPS requests.

   The element now reads: `<transport-guarantee>NONE</transport-guarantee>`.
6. Save the file, and then repeat these steps for the other `web.xml` deployment files.
7. Log in to *Dashboard Application Service Hub*.
8. In the navigation pane, click **Settings** > **Websphere Administrative Console** and click **Launch Websphere Administrative Console**.
9. In the WebSphere Application Server administrative console, select **Security** > **Global security** and click the **External authorization providers** link.
10. In the **External authorization providers** page, select the **Update with application names listed** option.
11. In the text pane, type `isc` and click **Apply**.
12. In the messages area at the top of the page, click the **Save** link to commit your changes to the master configuration.
13. Stop and restart the Dashboard Application Service Hub Server:
    a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

- Windows `stopServer.bat server1`
- Linux UNIX `stopServer.sh server1`

  Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

- Windows `startServer.bat server1`
- Linux UNIX `startServer.sh server1`

Example

The following example is a section of the `web.xml` file for `TIPChangePasswd` where the transport-guarantee parameter is set to `NONE`:

```
<security-constraint>
    <display-name>
    ChangePasswdControllerServletConstraint</display-name>
    <web-resource-collection>
        <web-resource-name>ChangePasswdControllerServlet</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <description>Roles</description>
        <role-name>administrator</role-name>
        <role-name>operator</role-name>
        <role-name>configurator</role-name>
        <role-name>monitor</role-name>
        <role-name>iscadmins</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

What to do next

Users must now specify a different port, depending on the mode of access. The default port numbers are as follows:

http://*<host_name>*:16310/ibm/console
Use the HTTP port for logging in to the Dashboard Application Service Hub on the HTTP port .

https://*<host_name>*:16311/ibm/console
Use the HTTPS secure port for logging in to the Dashboard Application Service Hub.

Note: If you want to use single sign-on (SSO) then you must use the fully qualified domain name of the Dashboard Application Service Hub host.

Configuring the LPTA token timeout value

You can configure the Lightweight Third Party Authentication (LTPA) token timeout value for Dashboard Application Service Hub in the WebSphere Application Server console.

Before you begin

Dashboard Application Service Hub is enabled for single sign-on.

About this task

The default timeout for an LTPA token is 120 minutes. An LTPA timeout causes you to be logged out from Dashboard Application Service Hub and can also cause an authentication popup message, if the first request after the timeout is an AJAX request from a portlet. To configure the LTPA token timeout:

Procedure

1. In the Dashboard Application Service Hub navigation pane, click **Settings** > **WebSphere Admin Console**.
2. Click **Launch WebSphere Admin Console** to start the WebSphere Application Server console.
3. In the WebSphere Application Server console navigation pane, click **Security** > **Global security**.
4. In the Authentication area of the **Global security** page, click the **LTPA** link.
5. In the LTPA timeout area of the **LTPA** page, edit the value for the LTPA timeout and click **OK**.
6. In the Messages area at the top of the **Global security** page, click the **Save** link and log out of the WebSphere Application Server console.

What to do next

In a load balanced environment, you must set the LTPA token timeout value on each of the Dashboard Application Service Hub Server instances.

Deleting a data source definition

Before you create a CMS data source, in some circumstance you many want to delete an existing data source definition.

About this task

As part of the Data Integration Services (DIS) database creation, the `DBConfig` installer also creates an external CMS database. Dashboard Application Service Hub applications use an external CMS database to both publish their CMS launch definitions as well as to obtain the launch definitions from other products. Tivoli Business Service Manager creates a data source definition in WebSphere Application Server for the Data Integration Services (DIS) database, CMS infers the CMS external database location from this since the CMS tables are created in the DIS database. If the CMS external database tables reside in the DIS database, then there may not be an existing CMS data source and the DIS datasource is used instead. If this is the case then the data source does not need to be removed.

To delete a data source:

Procedure

1. Run the following command to list existing data sources:

   ```
   $AdminConfig list DataSource
   ```

2. Run the following command to remove the data source:

   ```
   $AdminConfig remove ds_name_string
   ```

   Where *ds_name_string* is the name of the data source (which was displayed after you completed Step 1) that you want to remove.

3. Save your changes:

   ```
   $save
   ```

Configuring the CMS database

The Context Menu Service (CMS) is a component of Dashboard Application Service Hub which can be used by TBSM to share information outside of the Dashboard Application Service Hub environment.

CMS facilitates *launch-in-context* capability between products. The term *launch-in-context* is used to describe the ability for one application to invoke a function or launch a user interface provided by another application while also passing in data that the function or user interface may immediately process. CMS enables *launch-in-context* by allowing a product to register launch points for itself and locate launch points for other products. Launch points provide information to allow an application to invoke a function or user interface from another application.

If you want to change the location of the CMS database after installation, you must update the Dashboard server.

Creating a database for CMS

Copy CMS scripts from your Dashboard Application Service Hub installation to your remote computer and create a database.

About this task

To create a remote database for CMS:

Procedure

1. On the computer running Dashboard Application Service Hub, at the command line, change to the following directory:

   `/opt/IBM/JazzSM/profile/bin`

   The CMS directory contains a number of scripts that are provided by Dashboard Application Service Hub. The script that you use depends on the type of database and the operating system of the database computer:

   - Linux UNIX `db2_scripts.tar` for a DB2 database
   - Linux UNIX `MsSql_scripts.tar` for a Microsoft SQL Server database
   - Linux UNIX `Oracle_scripts.tar` for an Oracle database
   - Windows `db2_scripts.zip` for a DB2 database
   - Windows `MsSql_scripts.zip` for a Microsoft SQL Server database
   - Windows `Oracle_scripts.zip` for an Oracle database

   The steps described here reflect setting up a DB2 database on a on a Microsoft Windows system.
2. Transfer a copy of the relevant script file from the CMS directory to your remote database computer and take note of the location in which you save the file.

   For example, for a DB2 database running on a Microsoft Windows system, you need to transfer a copy of `db2_scripts.zip` to the remote computer.
3. On the remote database system, extract the file that you copied to a known location and at the command line change to that directory.

   For example, for a DB2 database: `cd C:\demo\db2scripts\db2`
4. Open the `CMS_`*`database_type`*`_Readme.txt` file, in this case `CMS_DB2_ReadMe.txt`, in a text editor.

   This file provides instructions and samples on how to use the scripts provided.
5. Open a database command window, so that you can execute database commands.

   For example, for a DB2 database running on a Windows system, click **Start** > **IBM DB2** > **DB2COPY1 (default)** > **Command Line Tools** > **Command Window**.
6. In the command window, change to the directory that contain your extracted script files.

   For example, `cd demo\db2_scripts\db2`
7. Run the database **setup** command providing the relevant arguments to the parameters outlined in the `CMS_`*`database_type`*`_Readme.txt` file for the database **setup** command.

   For example, run `CMS_DB2Setup.bat -d` *`database_name`* `-u` *`database_user_name`* `-p` *`database_user_password`* .

   Where:

database_name
The name of the database that you want to create. You can also provide the name of an existing database.

database_user_name
The user name for the database.

database_user_password
The user password associated with the specified user name.

The database is now ready to communicate with a Dashboard Application Service Hub data source.

What to do next

When you have set up a remote database, you can configure a data source in Dashboard Application Service Hub that CMS can use.

Configuring a hostname to be used by CMS

Configure a hostname to be used by CMS.

About this task

You need to set a hostname that CMS can use. For example, in a load balanced environment, it may not be obvious which hostname CMS should use. To specify a hostname to CMS:

Procedure

1. On the computer running Dashboard Application Service Hub, at the command line, change to the following directory:

   `/opt/IBM/JazzSM_bkup/ui/bin/cms`
2. Run the **cmssetconf** command to view details of the different options that are available to you in setting up CMS to use the remote database.

   Linux `./cmssetconf.sh`

   Windows `cmssetconf.bat`

   One of the settings that you apply using the **cmssetconf** command, is the hostname.
3. Run the following command to specify the hostname that you want to use:

   Linux `./cmssetconf.sh -hostname hostname -port DASH_port_number`

   Windows `cmssetconf.bat -hostname hostname -port DASH_port_number`

   The hostname in now configured.
4. Run the following command to review your CMS configuration and verify that you have correctly specified the hostname:

   Linux `./cmsshowconf.sh -hostname hostname -port DASH_port_number`

   Windows `cmsshowconf.bat -hostname hostname -port DASH_port_number`
5. Stop and restart the Dashboard Application Service Hub Server:
   a) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

   - Windows `stopServer.bat server1`
   - Linux UNIX `stopServer.sh server1`

     Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

b) In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:

- Windows `startServer.bat server1`
- Linux UNIX `startServer.sh server1`

What to do next

When you have configured the hostname, you can set up logging for CMS.

Administering

The administrator tasks involve configuring and customizing the environment and controlling access to it.

In a single installation the Dashboard Application Service Hub provides a product design environment and customization, with services that enable multiple-product integration.

Logging in

Log in to the portal whenever you want to start a work session.

Before you begin

The Dashboard Application Service Hub Server must be running before you can connect to it from your browser.

About this task

Complete these steps to log in:

Procedure

1. In a Web browser, enter the URL of the Dashboard Application Service Hub Server: `http://`*`host.domain`*`:16310/ibm/console` or `https://`*`host.domain`*`:16311/ibm/console` if it is configured for secure access.
   - *`host.domain`* is the fully qualified host name or IP address of the Dashboard Application Service Hub Server (such as *MyServer.MySubdomain.MyDomain.com* or 9.51.111.121, or `localhost` if you are running the Dashboard Application Service Hub Server locally).
   - `16310` is the default nonsecure port number for the portal and `16311` is the default secure port number. If your environment was configured with a port number other than the default, enter that number instead. If you are not sure of the port number, read the application server profile to get the correct number.
   - `ibm/console` is the default path to the Dashboard Application Service Hub Server, however this path is configurable and might differ from the default in your environment.
2. In the login page, enter your user ID and password and click **Log in**.

   This is the user ID and password that are stored with the Dashboard Application Service Hub Server.

   Attention: After authentication, the web container used by the Dashboard Application Service Hub Server redirects to the last URL requested. This is usually `https://<host>:<port>/ibm/console`, but if you manually change the page URL, after being initially directed to the login page, or if you make a separate request to the server in a discrete browser window before logging in, you may be redirected unexpectedly.

   Note: If you have more than one instance of the Dashboard Application Service Hub Server installed on your computer, you should not run more than one instance in a browser session, that is, do not log in to different instances on separate browser tabs.

Results
After your user credentials have been verified, the Welcome page is displayed. If you entered the localhost or port number incorrectly, the URL will not resolve. View the application server profile to check the settings for localhost, port, and user ID.

What to do next
Select any of the items in the navigation tree to begin working with the console.

While you are logged into the Dashboard Application Service Hub Server, avoid clicking the browser **Back** button because you will be logged out automatically. Click **Forward** and you will see that your are logged out and must resubmit your credentials to log in again.

Note: If you want to use single sign-on (SSO) then you must use the fully qualified domain name of the Dashboard Application Service Hub host.

Stopping and starting the application server

The Dashboard Application Service Hub Server starts automatically after it has been installed, and on systems running Windows, whenever the computer is started.

About this task
You can manually stop the Dashboard Application Service Hub Server before beginning certain configuration tasks or as needed.

Note: For environments using a central user repository, for example LDAP, a user must be given the *Administrator* role in the WebSphere Application Server administrative console before they can stop the Dashboard Application Service Hub Server. For information on assigning WebSphere Application Server roles, see: http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_tselugradro.html

Procedure
1. In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
   - Windows `stopServer.bat server1`
   - Linux UNIX `stopServer.sh server1`

     Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.
2. In the `/opt/IBM/JazzSM/profile/bin` directory, depending on your operating system, enter one of the following commands:
   - Windows `startServer.bat server1`
   - Linux UNIX `startServer.sh server1`

Port assignments

The application server requires a set of sequentially numbered ports.

The sequence of ports is supplied during installation in the response file. The installer checks that the number of required ports (starting with the initial port value) are available before assigning them. If one of the ports in the sequence is already in use, the installer automatically terminates the installation process and you must specify a different range of ports in the response file.

Viewing the application server profile

Open the application server profile to review the port number assignments and other information.

About this task

The profile of the application server is available as a text file on the computer where it is installed.

Procedure

1. Locate the `/opt/IBM/JazzSM/profile/logs` directory.
2. Open `AboutThisProfile.txt` in a text editor.

Example

This is the profile for an installation on in a Windows environment as it appears in `/opt/IBM/JazzSM/profile/logs\AboutThisProfile.txt`:

```
Application server environment to create: Application server
Location: C:\Program Files\IBM\JazzSM\profile
Disk space required: 200 MB
Profile name: DASHProfile
Make this profile the default: True
Node name: TIPNode Host name: tivoliadmin.usca.ibm.com
Enable administrative security (recommended): True
Administrative consoleport: 16315
Administrative console secure port: 16316
HTTP transport port: 16310
HTTPS transport port: 16311
Bootstrap port: 16312
SOAP connector port: 16313
Run application server as a service: False
Create a Web server definition: False
```

What to do next

If you want to see the complete list of defined ports on the application server, you can open `/opt/IBM/JazzSM/var/JazzSMProfile_portDef.properties` in a text editor:

```
#Create the required WAS port properties for TIP
#Mon Oct 06 09:26:30 PDT 2008
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=16323
WC_adminhost=16315
DCS_UNICAST_ADDRESS=16318
BOOTSTRAP_ADDRESS=16312
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=16321
SOAP_CONNECTOR_ADDRESS=16313
ORB_LISTENER_ADDRESS=16320
WC_defaulthost_secure=16311
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=16322
WC_defaulthost=16310
WC_adminhost_secure=16316
```

Changing passwords

You can use the **Change Your Password** portlet to change your password from the default provided by the administrator.

About this task

When you log in to the portal, you can change your own password using the **Change Your Password** portlet. Administrators can change passwords for other users using the **Manage Users** portlet.

Attention: If you are an administrator and you want to change the password for the `tbsmadmin` administrator and the Tivoli Netcool/OMNIbus ObjectServer root user, you must use the **Settings > WAS Admin Console** portlet to change their password. Do not use the **Users and Groups** > **Manage Users** portlet.

Tip: For security reasons, change the password of the Tivoli Netcool/OMNIbus ObjectServer root user after installation.

To change passwords:

Procedure

- To change your own password, follow these steps:
  a) Log in to the portal using the user ID whose password you would like to change.
  b) In the navigation pane, click **Settings** > **WAS Admin Console**.
  c) Enter your new password in the relevant fields and click **Set Password**.
- As an administrator, to change the password for a user, follow these steps:
  a) In the navigation pane, click **Users and Groups** > **Manage Users** and click the user's name from the **User ID** column.
  A **User Properties** page is displayed.
  b) In the **General** tab, enter the new password in the relevant fields and click **OK**.

Attention:

If you authenticate to a Microsoft Active Directory server, it must be configured for SSL before you can use the **Change Your Password** portlet. If SSL is not enabled, you will receive an error when attempting to change the password for any user who is registered on the Active Directory Server.

```
TIPCP0005E  Could not set the password via the underlying security system.
This could be because a password rule was not met, you do not have
access to change the password, or another reason.
```

Changing password on nameserver

The Dashboard Application Service Hub server uses the properties in the `$JAZZSM_HOME/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/nameserver.props` file to determine the Name Server connection details.

If for any reason the `impactadmin` password changes (for example, if you switch from file-based user registry to LDAP), you must update the password details for the `impactadmin` user in the `nameserver.props` file. To do so, you must manually encrypt the password and then edit the `nameserver.props` file.

For example, to retrieve the encrypted password for `impactadmin`, use the following command:

```
/opt/IBM/WebSphere/AppServer/bin/crypto.sh {impactadmin user password}
```

Then edit the `nameserver.props` file with the encrypted password:

```
impact.nameserver.userid=impactadmin
impact.nameserver.password={AES}A1FD4E0DE2389E727873104C811FB744
```

Exporting and importing

You can export customized configuration data from an existing *Dashboard Application Service Hub* installation to another by exporting the data and subsequently importing the exported data.

Exporting and importing customized settings can be done at the command line through the `tipcli.bat|.sh Export` and `tipcli.bat|sh Import` commands.

Note: The `tipcli.bat|.sh Export` and `tipcli.bat|sh Import` commands are case sensitive. Also, if you make a typing error, that is, if you type a parameter incorrectly, or use the incorrect case, then the commands runs as if no parameters were specified and no warning message is displayed.

You can export and import the following elements:

- Custom pages and customized system page elements, with the exception of core and system pages, including:
  - Page name and layout.
  - Portlet entities.

    Note: Copies of a portlet entity are not exported; either through the console Export Wizard or through the **`tipcli.bat|.sh Export`** command.
  - View profiles.
  - Events and wires.
  - Access permissions.
  - Navigation structure.
- Custom views (or customized system views).

  Note: You can also export pages associated with a view if the `exportpageinview` parameter is set to `true`.
- Custom roles, including:
  - Role name, creation date, and update date.
  - Role mapping information in relation to users and groups.
  - Associated role preference, that is, the relevant console preference profile.
- Console properties and customization properties, including:
  - Transformations.
  - Themes and images.
  - Bundles.

In a load balanced environment the import operation migrates imported elements across all the computers in the pool, with following conditions:

- All the required applications (WAR files) must be deployed on all computers in the pool.
- The load balanced pool configuration must be locked during the import operation.
- The import operation must be ran on one of the nodes in the pool.

  Restriction: In a load balanced environment that includes charting, the `ListRestore` command only runs successfully on the node that is used for the import operation because backup files are stored locally on that node and are not synchronized across other nodes in the cluster.
- You must provide the load balancing manager an updated file list to update the load balancing scope. The migration tool plugin provides the file list.
- The load balanced pool configuration, can then be unlocked.
- The import of transformations in a load balanced environment is not supported. Transformations must be imported to each node independently.

  The **`haSupport`** command controls this aspect of the import operation:
  - If it is set to `True`, then only load balancing information is imported, that is, no transformation data.
  - If it is set to `False`, then only transformation data is imported, that is, no load balancing data.
  - If it is set to `Both`, then transformation data and load balancing data is imported.

Basic export commands

You can export pages, views and profile preferences using the basic export commands.

Exporting pages in simplified mode

By using the **ExportPage** command you can export specific pages without having to provide additional qualifying parameters.

Before you begin

Ensure that the Dashboard Application Service Hub Server is running.

About this task

To export specific pages in simplified mode for an instance of Dashboard Application Service Hub:

Procedure

1. At the command line change to: `/opt/IBM/JazzSM/profile/bin`.
2. To return a list of customized pages that can be exported, run the following command:
   - Windows `C:\program Files\IBM\JazzSM\ui\bin\tipcli.bat ListPages --customizePages true`
   - Linux UNIX *tip_home_dir*`/opt/IBM/JazzSM/ui/bin/tipcli.sh ListPages --customizePages true`

   Note: The page ID is the last element of the returned records, for example, the page ID for the following record is `BIXRjLkKYngNsRavnu0fYpx1279539744250`:

   ```
   com.ibm.isclite.global.custom.module-SPSVS-
   com.ibm.isclite.admin.PortletPicker.navigationElement
   .pagelayoutA
   .modified.BIXRjLkKYngNsRavnu0fYpx1279539744250
   ```
3. Review the list of returned page records and take note of the page IDs for the pages that you want to export.
4. To export specific pages, run the following command:
   - Windows *tip_home_dir*`C:\program Files\IBM\JazzSM\ui\bin\tipcli.bat ExportPage --uniqueName` *pageID_1*`,`*pageID_2*`,`*pageID_3* `--username` *tbsmadmin_user_name* `--password` *tbsmadmin_password*
   - Linux UNIX *tip_home_dir*`/opt/IBM/JazzSM/ui/bin/tipcli.sh ExportPage --uniqueName` *pageID_1*`,`*pageID_2*`,`*pageID_3* `--username` *tbsmadmin_user_name* `--password` *tbsmadmin_password*

   Note: The file `portletEntities.xml` is always exported, even if you specify `NONE` as an argument to the `uniqueName` parameter.

Results

When the command completes, a `Data.zip` file is created in `/opt/IBM/JazzSM/ui/output`.

What to do next

Locate `/opt/IBM/JazzSM/ui/output/Data.zip` and copy it to the computer where you intend to apply the exported customization data.

Exporting views in simplified mode
By using the **ExportView** command you can export specific views without having to provide additional qualifying parameters.

Before you begin

Ensure that the Dashboard Application Service Hub Server is running.

About this task

To export specific views in simplified mode for an instance of Dashboard Application Service Hub:

Procedure

1. At the command line change to: `/opt/IBM/JazzSM/profile/bin`.
2. Optional: To return a list of customized views that can be exported, run the following command:
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ListViews`
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh ListViews`
3. Review the list of returned view records and take note of the view IDs for the views that you want to export.
4. To export specific views, run the following command:
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ExportView --uniqueName` *viewID_1, viewID_2, viewID_3*
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh ExportView --uniqueName` *viewID_1, viewID_2, viewID_3*

   Note: The file `portletEntities.xml` is always exported, even if you specify `NONE` as an argument to the `uniqueName` parameter.

Results

When the command completes, a `Data.zip` file is created in `/opt/IBM/JazzSM/ui/output`.

What to do next

Locate `/opt/IBM/JazzSM/ui/output/Data.zip` and copy it to the computer where you intend to apply the exported customization data.

Exporting console preference profiles in simplified mode
By using the `ExportProfile` command you can export console preference profiles without having to provide additional qualifying parameters.

Before you begin

Ensure that the Dashboard Application Service Hub Server is running.

About this task

To export console preference profiles in simplified mode:

Procedure

1. At the command line change to: `/opt/IBM/JazzSM/profile/bin`.
2. Optional: To return a list of console preference profiles that can be exported:

- Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ListPreferenceProfiles`
- Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh ListPreferenceProfiles`

3. Review the list of returned records and take note of the unique names for the console preference profiles that you want to export.
4. To export specific console preference profiles, run the following command:
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ExportProfile --uniqueName` *`profile_ID1,profile_ID2,profile_ID3`*
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh ExportProfile --uniqueName` *`profile_ID1,profile_ID2,profile_ID3`*

   Note: The file `portletEntities.xml` is always exported, even if you specify `NONE` as an argument to the `uniqueName` parameter.

Results

When the command completes, a `Data.zip` file is created in `/opt/IBM/JazzSM/ui/output/`.

What to do next

Locate `/opt/IBM/JazzSM/ui/output/Data.zip` and copy it to the computer where you intend to apply the exported customization data.

Advanced export commands

You can use the advanced `tipcli Export` commands and apply a number of parameters to define which items you want to include and exclude in relation to the export operation.

Exporting all customization data

You can export all customization data for an instance of Dashboard Application Service Hub in one command.

Before you begin

Ensure that the Dashboard Application Service Hub Server is running.

About this task

To export all customization data for an instance of Dashboard Application Service Hub:

Procedure

1. At the command line change to: `/opt/IBM/JazzSM/ui/bin`.
2. Optional: To return a list of plugins that will be run during the export operation, run the following command:
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh ListExportPlugins`
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ListExportPlugins`
3. To export all customization data, run the following command:
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh Export --username` *`tbsmadmin_user_name`* `--password` *`tbsmadmin_password`*
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat Export --username` *`tbsmadmin_user_name`* `--password` *`tbsmadmin_password`*

Results

When the `Export` command completes, a `Data.zip` file is created in `/opt/IBM/JazzSM/ui/output/`.

Note:

Refer to the links at the end of the page to view details of customs parameters that can be applied to the `Export` command.

What to do next

Locate `/opt/IBM/JazzSM/ui/output/Data.zip` and copy it to the computer where you intend to apply the exported customization data.

Exporting using a properties file

You can specify your export requirements in properties file instead of specifying your requirements using separate parameters at the command line.

Before you begin

By default, the `tipcli` command uses the `/opt/IBM/JazzSM/ui/etc/tipcli.properties` file unless this behavior is overridden by the specifying a discrete settings file using the `settingFile` parameter.

Ensure that the Dashboard Application Service Hub Server is running.

About this task

To export customization data using a properties file:

Procedure

1. Create a properties file that specifies the data that you want to export and save it as *`export-settings`*`.properties` in a known location.

   Below is example content for an export properties file:

   ```
   import.includePlugins=ImportPagePlugin
   export.includePlugins=ExportPagePlugin
   import.backupDir=c:/tmp/bkups
   export.exportFile=c:/tmp/extest.zip
   import.importFile=c:/tmp/extest.zip
   username=tbsmadmin
   password=tbsmadmin_password
   import.haSupport=true
   ```

   Note: Some parameters are import or export specific. Import specific parameters should be prefixed by `import.` and export specific parameters should be prefixed by `export.`. For example, `import.backupDir=c:/tmp/bkups`.
2. At the command line change to: `/opt/IBM/JazzSM/ui/bin`.
3. To export customization data based on the contents of a specific properties file, run the following command:

   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh Export --username` *`tbsmadmin_user_name`* `--password` *`tbsmadmin_password`* `--settingFile` *`export_properties_file`*
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat Export --username` *`tbsmadmin_user_name`* `--password` *`tbsmadmin_password`* `--settingFile` *`export_properties_file`*

   Where:

export_properties_file
An argument to the `settingFile` parameter that provides the location and name of the export properties file, for example, `C:\\tmp\\export.properties`.

Note: Windows You must use double backslashes characters (\\) when specifying the path to your settings file.

Note: If there is a conflict between settings specified in the properties file and parameters provided at the command line, then the command line parameters take precedence.

Results

When the **Export** command completes, a `extest.zip` file is created in the root temporary directory, for example on Windows systems the file is saved in `c:\tmp`.

What to do next

Locate `extest.zip` and copy it to the computer where you intend to apply the exported customization data.

Exporting specific pages
When exporting Dashboard Application Service Hub data, you can specify that you want to export particular pages.

Before you begin

Ensure that the Dashboard Application Service Hub Server is running.

About this task

To export specific pages for an instance of Dashboard Application Service Hub:

Procedure

1. At the command line change to: `/opt/IBM/JazzSM/ui/bin/`.
2. To return a list of customized pages that can be exported, run the following command:
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ListPages --customizePages true`
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin//tipcli.sh ListPages --customizePages true`

   Note: The page ID is the last element of the returned records, for example, the page ID for the following record is `BIXRjLkKYngNsRavnu0fYpx1279539744250`:

   ```
   com.ibm.isclite.global.custom.module-SPSVS-
   com.ibm.isclite.admin.PortletPicker.navigationElement
   .pagelayoutA
   .modified
   .BIXRjLkKYngNsRavnu0fYpx1279539744250
   ```
3. Review the list of returned page records and take note of the page IDs for the pages that you want to export.
4. To export specified pages, run the following command:
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh Export --username` *`tbsmadmin_user_name`* `--password` *`tbsmadmin_password`* `--pages` *`pageID_1, pageID_2, pageID_3`*

- Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat Export --username tbsmadmin_user_name --password tbsmadmin_password --pages pageID_1, pageID_2, pageID_3`

Results
When the command completes, a `Data.zip` file is created in `/opt/IBM/JazzSM/ui/bin/output/`.

What to do next
Locate `/opt/IBM/JazzSM/ui/bin/output/Data.zip` and copy it to the computer where you intend to apply the exported customization data.

Exporting specific views
When exporting Dashboard Application Service Hub data, you can specify that you want to export particular views.

Before you begin

Ensure that the Dashboard Application Service Hub Server is running.

About this task

To export specific views for an instance of Dashboard Application Service Hub:

Procedure

1. At the command line change to: `/opt/IBM/JazzSM/ui/bin/`.
2. Optional: To return a list of customized views that can be exported, run the following command:
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ListViews`
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh ListViews`
3. Review the list of returned view records and take note of the view IDs for the views that you want to export.
4. To export specific views, run the following command:
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh Export --username tbsmadmin_user_name --password tbsmadmin_password --views viewID_1,viewID_2,viewID_3 --exportpageinviews [true|false]`
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat Export --username tbsmadmin_user_name --password tbsmadmin_password --views viewID_1,viewID_2,viewID_3 --exportpageinviews [true|false]`

   Where:

   exportpageinviews
   An optional parameter, when set to `true` ensures that you also export pages associated with the views that you have specified.

   Note: Whether the optional parameter `exportpageinviews` is set to `true` or `false`, if a view has a default node in the navigation pane associated with it, then the page associated with the node is always exported. This is also true, even if you specify NONE as the argument to the `--pages` parameter.

Results
When the command completes, a `Data.zip` file is created in `/opt/IBM/JazzSM/ui/bin/output/`.

What to do next

Locate `/opt/IBM/JazzSM/ui/bin/output/Data.zip` and copy it to the computer where you intend to apply the exported customization data.

Rules for exporting

When exporting customized configuration data, it is important to know the rules governing the export function and the options available to you.

The following rules apply when exporting customized configuration data from a Dashboard Application Service Hub environment:

Rules and options for pages

1. You can export a particular page by page ID or choose to export all pages.
2. You can export pages associated with a particular view.
3. You can export pages that are associated with a particular portlet from a particular WAR.
4. If a page contains multiple portlets, but only some from a specified WAR, then all elements of the page are exported.
5. Pages that are targets of a wire for a specified page are exported.
6. The default export scope is `All` if you do not define pages to be exported under rule 2 and rule 3.
7. The default export scope is `NONE` if you define pages to be exported under rule 2 and rule 3.

Rules and options for views

1. You can export a particular view by view ID or choose to export all views.
2. You can optionally export all views that contains a specified page.
3. The default export scope is `All`.
4. You can optionally export all pages associated with the views that you want to export.
5. If an view has a default node in the navigation pane associated with it, then that page is automatically exported with the view.
6. Views that match the following conditions should not be exported as the subsequent import of that view will fail:
   - An empty view, that is, a view that contains no pages or roles.
   - A view that contains roles, but no pages.
   - A view that contains empty pages, that is, the page exists but it does not contain portlets.

Rules and options for custom roles and role preferences (console preference profiles)

1. You can export a particular role by role ID or choose to export all roles.
2. You can export a custom role and role preference that is associated with a specified page or view.
3. The default export scope is set to `All`, unless the **`includeEntitiesFromApps`** parameter has been specified for a page or view, whereby it is then set to `REQUIRED`.
4. If a console preference profile has a custom view as its default view, then that view is automatically exported. If the exported view has a default node in the navigation pane, then the associated page is automatically exported with the view.

Rules and options for user preferences

1. You can export user preferences by user ID or choose to export preferences for all users.
2. The default export scope is set to `All`, unless the **`includeEntitiesFromApps`** parameter has been specified for a page or view, whereby it is then set to `REQUIRED`.

Rules and options for console properties and customization properties

All console properties and customization properties are exported.

Rules and options for transformations

All transformations are exported.

Import commands

You can use the **tipcli Import** commands and apply a number of parameters to define which items you want to include and exclude in relation to the import operation.

Importing previously exported data

You can import data that was exported from another instance of Dashboard Application Service Hub.

Before you begin

Ensure that the Dashboard Application Service Hub Server is running.

Ensure that you have run the export operation on an originating instance of the Dashboard Application Service Hub Server and that you have copy the output file (`data.zip`) to the following directory on the other instance:

```
/opt/IBM/JazzSM/ui/bin/output
```

About this task

To import data from a `data.zip` file that was exported from another instance Dashboard Application Service Hub Server:

Procedure

1. At the command line change to: `$JAZZSM_HOME/profile/bin`.
2. Optional: To return a list of plugins that will be run during the import operation, run the following command:
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat ListImportPlugins`
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh ListImportPlugins`
3. To import the customization data, run the following command:
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat Import --username` *`tbsmadmin_user_name`* `--password` *`tbsmadmin_password`*
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh Import --username` *`tbsmadmin_user_name`* `--password` *`tbsmadmin_password`*

Results

When the **Import** command completes, the imported data is merged with the existing Dashboard Application Service Hub environment.

Rolling back imports

After you import data you can rollback your configuration to the pre-import state provided you have made no changes to the environment.

Before you begin

If you have performed multiple imports, you can also consecutively rollback individual imports. In all cases, you must have not had made changes to the environment.

Ensure that the Dashboard Application Service Hub Server is running.

About this task

To roll back imports for a Dashboard Application Service Hub environment:

Procedure

1. At the command line change to: `/opt/IBM/JazzSM/ui/bin/`.
2. To rollback an import, run the following command:
   - Windows `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat Import --rollback ALL`
   - Linux UNIX `/opt/IBM/JazzSM/ui/bin/tipcli.sh Import --rollback ALL`

   When the command completes successfully, the Dashboard Application Service Hub environment is restored to the state that prevailed before the latest import operation was performed.
3. Optional: If you performed multiple imports and you want to roll back more than the most recent import operation, you can re-run the `tipcli.bat Import --rollback ALL` command. You can re-run the rollback command multiple times to consecutively roll back a number of import operations.

   When you re-run the rollback command a second or subsequent time, the Dashboard Application Service Hub environment is restored to the state that prevailed prior the settings for that particular import operation being applied.

Rules for importing
When importing customized configuration data, it is important to know the rules governing the import function and the options available to you.

The following rules apply when importing customized configuration data for a Dashboard Application Service Hub environment:

Rules and options for pages

1. You can import all pages included in an exported package.
2. You can exclude system customized pages that do not exist in the new environment.
3. You can exclude pages associated with a WAR that is not deployed in the new environment and thereby avoid introducing empty pages.
4. If a page contains multiple portlets and some of portlets are associated with a WAR that is not deployed in the new environment, the page is not imported.

Rules and options for views

1. All views included in an exported package are imported.
2. Views that match the following conditions should not be imported as the import operation for the view fails:
   - An empty view, that is, a view that contains no pages or roles.
   - A view that contains roles, but no pages.
   - A view that contains empty pages, that is, the page exists but it does not contain portlets.

Rules and options for custom roles and role preferences (console preference profiles)
All roles included in an exported package are imported.

Rules and options for user preferences
All user preferences included in an exported package are imported.

Rules and options for console properties and customization properties
All console properties and customization properties included in an exported package are imported.

Rules and options for transformations
All transformations included in an exported package are imported, if the **haSupport** parameter is set to `Both` or `False`.

Table 1 provides details how various elements are processed during import:

Table 158. Rules for overwriting and merging during import		
Element	**Action**	**Comments**
Pages	Overwritten	In relation to pages, roles are merged, view memberships remain unchanged, and positions are modified.
Views	Overwritten	In relation to views, existing page memberships are merged with imported pages
Roles	Skipped	In relation to roles, user and group mappings are merged.
Console preference profiles	Skipped	
Credential data	Merged	
Property files	Merged	
Transformations	Skipped	
Charts	Overwritten	

Changing the default security registry

The default security registry can be set at install time. Use this procedure to change the default registry after installation.

Before you begin

These steps require that your user ID has the Administrator role and that you know the base entry value of your repository. For LDAP or Microsoft Active Directory, this is usually a string like `ou=company,dc=country,dc=region`. For the ObjectServer, the base entry is `o=netcoolObjectServerRepository`.

About this task

If you want to change the default to a different registry, complete these steps:

Procedure

1. Log into the Dashboard Application Service Hub.

   Your ID must have the Administrator role.
2. In the navigation pane, click **Settings** > **Websphere Admin Console** and click **Launch Websphere Admin Console**.
3. In the WebSphere Application Server administrative console navigation pane, click **Security** > **Secure administration, applications, and infrastructure**.
4. In the User account repositories area, select **Federated repositories** from the Available realm definitions, then click **Configure**.
5. Click **Supported entity types** under **Additional Properties**.
6. Click the entity type, then edit the **Base entry for the default parent** and **Relative Distinguished Name properties**.
7. After you click **OK** to save your changes, repeat the previous step to configure the other entity types.

   For Microsoft Active Directory, the entity types (PersonAccount, Group, and OrgContainer) must be configured with a base DN and the RDN for PersonAccount should be `cn` instead of `uid`.
8. Stop and restart the Dashboard Application Service Hub Server:

a) In the/opt/IBM/JazzSM/profile/bin directory, depending on your operating system, enter one of the following commands:

- Windows `stopServer.bat server1`
- Linux UNIX `stopServer.sh server1`

  Note: On UNIX and Linux systems, you are prompted to provide an administrator username and password.

b) In the /opt/IBM/JazzSM/profile/bin directory, depending on your operating system, enter one of the following commands:

- Windows `startServer.bat server1`
- Linux UNIX `startServer.sh server1`

CGI support

Use the initialization parameters to control the behavior of CGIServlet.

CGIServlet

CGI scripts run on a Web server and use the Common Gateway Interface (CGI) to perform tasks. The support for CGI in Dashboard Application Service Hub is provided by *CGIServlet*, extracted from Apache Tomcat. The Tomcat CGI support is largely compatible with the Apache HTTP Server but there are some limitations (such as only one cgi-bin directory). To change the configuration, edit `web.xml` in the directory where the CGI application is installed.

Servlet initialization parameters

Several initialization parameters are available for configuring the behavior of the CGIServlet.

cgiPathPrefix
The CGI search path will start at the Web application root directory + File.separator + this prefix. Default setting: `cgiPathPrefix is Web-INF/cgi`.

debug
Determines the level of debugging detail for messages that are logged by the servlet. Default setting: `0`.

executable
This is type of the program to be used to run the script. Default setting: `perl`.

parameterEncoding
Names the parameter encoding to be used with the CGI servlet. Default setting: `System.getProperty("file.encoding","UTF-8")`.

passShellEnvironment
Determines whether shell environment variables, if there are any, shall be passed to the CGI script. Default setting: `false`.

Setting Java Virtual Machine memory for DASHProfile

You can increase the amount of memory available to the Dashboard Application Service Hub.

About this task

To increase (or decrease) the amount of memory available to the Java Virtual Machine (JVM), carry out the following steps:

Procedure

1. Manually stop the application server.

2. Change to the `/opt/IBM/JazzSM/profile/bin` directory.
3. Use the **wsadmin** command to increase the heap size for the JVM, as follows:
   `wsadmin.sh -lang jython -conntype NONE`
4. At the `wsadmin>` prompt, issue the following commands, where *xxx* is the new heap size value, in megabytes.

```
jvm=AdminConfig.list("JavaVirtualMachine")

AdminConfig.modify(jvm, '[[initialHeapSize xxx]]')
AdminConfig.modify(jvm, '[[maximumHeapSize xxx]]')
AdminConfig.save()

exit
```

5. Restart the Dashboard Application Service Hub Server.
   The changes take effect when the Dashboard Application Service Hub Server is restarted.

Attention: If you attempt to start the Dashboard Application Service Hub Server with a maximum heap size that is too large, error messages that are similar to the following are generated in the `/opt/IBM/JazzSM/profile/logs/server1/native_stderr.log` file:

```
JVMJ9GC019E -Xms too large for -Xmx
JVMJ9VM015W Initialization error for library j9gc23(2): Failed to initialize
Could not create the Java virtual machine.
```

Checking hostname settings

The value of the Hostname property in the `/opt/IBM/JazzSM/profile/properties` file is used by Dashboard Application Service Hub to convert incoming browser requests (for example, http://*<SystemName>*:16310) to the appropriate Dashboard Application Service Hub non-secure access (for example, http://*<HostnameValue>*:16315)/ibm/console), which is then converted to the Dashboard Application Service Hub secure access (for example, https://*<HostnameValue>*:16316/ibm/console/login.jsp).

About this task

The Hostname property should contain the fully qualified hostname. This is required if the web browser being used to access Dashboard Application Service Hub is running on a machine in a different DNS domain to the Dashboard Application Service Hub Server (application server).

The value of the `/opt/IBM/JazzSM/profile/properties/tip.properties` file's Hostname entry is set during installation by a routine built into Java that checks the `/etc/hosts` (or `%WinDir%\system32\drivers\etc\hosts`) entry for the system; if the fully qualified domain name (FQDN) is not set in `/etc/hosts`, the Java routine returns either the short name or the IP address of the machine, depending on the type of operating system (all but AIX).

Therefore, before the *Dashboard Application Service Hub* installer is run, ensure that a line exists in `/etc/hosts` of the following form:

IP address FQDN shortname
For example: 9.10.11.12 yourserver.domainname.com yourserver

This line ensures that the FQDN is set as the Hostname entry at install time in `/opt/IBM/JazzSM/profile/properties/tip.properties`.

If you try to connect to the application server and the URL conversion to the non-secure access appears to be working incorrectly, you should check Hostname property entry in `tip.properties`.

Procedure

1. Open the `/opt/IBM/JazzSM/profile/properties/tip.properties` file in a text editor.
2. Check the Hostname property and make sure the value can be correctly resolved by the web browser being used to access the application server.

3. Edit the Hostname entry to the FQDN of the application server and save the changes.
4. Stop and restart the application server.

   The changes take effect when the application server is restarted.

Accessing Context Menu Service features

To access Context Menu Service features from within *Dashboard Application Service Hub*, you must be assigned the Monitor role in Dashboard Application Service Hub.

About this task

The Context Menu Service, a component of Dashboard Application Service Hub, facilitates launch-in-context capability between products. This capability enables one application to invoke a function or launch a user interface that is provided by another application while also passing data that the function or user interface can immediately process. To access Context Menu Service features, for example, CMS command line functions, you must be assigned the *Monitor* role in Dashboard Application Service Hub.

To assign the *Monitor* role to a user in Dashboard Application Service Hub:

Procedure

- You can assign roles to users in the portal or by using the `tipcli` command:
  - To assign the *Monitor* role to a user in the portal, from the navigation pane, click **Users and Groups** > **User Roles**. Search for the user, assign the *Monitor* role and save your changes.
  - To assign the *Monitor* role to a user using the `tipcli` command, at the command line change to `/opt/IBM/JazzSM/ui/bin/` and enter the following command:

    Windows `tipcli.bat MapUsersToRole --username` *`DASH_username`* `--password` *`DASH_user_password`* `--roleName monitor --usersList` *`user_ID`*

    Linux UNIX `tipcli.sh MapUsersToRole --username` *`DASH_username`* `--password` *`DASH_user_password`* `--roleName monitor --usersList` *`user_ID`*

Command reference

Use the Dashboard Application Service Hub command line interface *tipcli* commands for writing scripts for passing information between applications.

The `tipcli` commands are entered in the `/opt/IBM/JazzSM/ui/bin` directory, for example, `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat` on Windows or `/opt/IBM/JazzSM/ui/bin/tipcli.sh` on Linux or UNIX.

The `tipcli` component provides help for its various commands:

`Help [--command` *`command_name`*`]`
: Access help for all commands or optionally you can use the `command` argument to return detailed help for a specific command.

The following returns help for the `AddUpdatePreferenceProfile` command:

```
tipcli.bat Help --command AddUpdatePreferenceProfile
```

```
Help
----
AddUpdatePreferenceProfile --username <TIPusername> --password <passwordForUser>
 --profileName <profileName> [--newProfileName <newProfileName>] [--themeDir <th
emeDir>] [--showNavTree <true|false>] [--componentDir <default|ltr|rtl>] [--text
Dir <default|contextual|ltr|rtl>] [--views <viewList>] [--roles <roleList>] [--d
efaultView <defaultView>]
where
  <TIPusername> is the username on TIP that has iscadmins role.
  <passwordForUser> is the password for the user.
  <profileName> is profile name which will be created or updated.
  <newProfileName> is the new name for the existing preference profile.
```

```
  <themeDir> is the directory name of the installed theme. Example: TIPLight
  <showNavTree> specify if show navigation tree by default after login the conso
le.
  <componentDir> specify component direction for the console.
  <textDir> specify text direction for the console.
  <viewList> is views assignment for the preference profile.
  <roleList> is roles assignment for the preference profile.
  <defaultView> specify which view is displayed by default after login the conso
le.

CTGWA4017I The command completed successfully.
```

Working with roles

Use these `tipcli` commands for to manipulate roles.

ListRoles

Use the **ListRoles** command to list all roles configured for a portal instance.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh ListRoles`
- Windows `tipcli.bat ListRoles`

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh ListRoles
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance that you want to query.

AddRole

Use the **AddRole** command to add a specified role to the portal instance. Portal users are granted access to resources based on the role to which they are assigned. All roles created with this command have a resource type of `Custom`.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh AddRole --username` *`DASH_username`* `--password` *`DASH_user_password`* `--roleName` *`role_name`*
- Windows `tipcli.bat AddRole --username` *`DASH_username`* `--password` *`DASH_user_password`* `--roleName` *`role_name`*

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
role_name is the name of the role to be added.

Note: Arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh AddRole --username DASH_username --password
DASH_user_password --roleName role_name
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance involved.

UpdateRole
Use the **UpdateRole** command to change the name of a custom role.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh UpdateRole --username DASH_username --password DASH_user_password --roleName role_name --newRoleName new_role_name`
- Windows `tipcli.bat UpdateRole --username DASH_username --password DASH_user_password --roleName role_name --newRoleName new_role_name`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
role_name is the name of the role to be modified.
new_role_name is the new name you want for the specified role.

Note: Arguments to the **role_name** and **newRoleName** parameters must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh UpdateRole --username DASH_username --password
DASH_user_password --roleName role_name --newRoleName new_role_name
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance involved.

DelRole
Use the **DelRole** command to delete a custom role.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh DelRole --username DASH_username --password DASH_user_password --roleName role_name`
- Windows `tipcli.bat DelRole --username DASH_username --password DASH_user_password --roleName role_name`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
role_name is the name of the role to be modified.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh DelRole --username DASH_username --password
DASH_user_password --roleName role_name
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance involved.

ListRolesFromGroup
Use the **ListRolesFromGroup** command to list all roles associated with a specified user group.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh ListRolesFromGroup --username DASH_username --password DASH_user_password --groupID group_ID`
- Windows `tipcli.bat ListRolesFromGroup --username DASH_username --password DASH_user_password --groupID group_ID`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
group_ID is the name of the user group associated with the roles that you want to list.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh ListRolesFromGroup --username DASH_username --password DASH_user_password --groupID group_ID
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance involved.

MapRolesToGroup
Use the **MapRolesToGroup** command to associate a comma-separated list of roles to a specified user group.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh MapRolesToGroup --username DASH_username --password DASH_user_password --groupID group_ID --rolesList role_name1, role__name2`
- Windows `tipcli.bat MapRolesToGroup --username DASH_username --password DASH_user_password --groupID group_ID --rolesList role_name1, role__name2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
group_ID is the name of the user group associated with the roles that you want to map.
role_name1, *role__name2* is a comma-separated list of roles that are to be associated with the specified user group.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh MapRolesToGroup --username DASH_username --
password DASH_user_password --groupID group_ID --rolesList role_name1,
role__name2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

RemoveRolesFromGroup
Use the **RemoveRolesFromGroup** command to disassociate a comma-separated list of roles from a specified user group.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh RemoveRolesFromGroup --username DASH_username --password DASH_user_password --groupID group_ID --rolesList role_name1, role__name2`
- Windows `tipcli.bat RemoveRolesFromGroup --username DASH_username --password DASH_user_password --groupID group_ID --rolesList role_name1, role__name2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
group_ID is the name of the user group from which roles are to be removed.
role_name1, *role__name2* is a comma-separated list of roles to be removed from the user group.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh RemoveRolesFromGroup --username DASH_username
--password DASH_user_password --groupID group_ID --rolesList role_name1,
role__name2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance involved.

ListRolesForPage
Use the **ListRolesForPage** command to list all roles associated with a specified page.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh ListRolesForPage --pageUniqueName page_unique_name`
- Windows `tipcli.bat ListRolesForPage --pageUniqueName page_unique_name`

Where:

page_unique_name is the unique ID for the page.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh ListRolesForPage --pageUniqueName
page_unique_name
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

MapRolesToPage
Use the **MapRolesToPage** command to associate a comma-separated list of roles with a specified page and set an access level for each role.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh MapRolesToPage --username DASH_username --password DASH_user_password --pageUniqueName page_unique_name --rolesList role_name1, role__name2 --accessLevelList level1, level2`
- Windows `tipcli.bat MapRolesToPage --username DASH_username --password DASH_user_password --pageUniqueName page_unique_name --rolesList role_name1, role__name2 --accessLevelList level1, level2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
page_unique_name is the page ID with which to associate with the list of roles.
role_name1, *role__name2* is a comma-separated list of roles that are to be associated with the page.
level1, *level2* is a comma-separated list of page access levels that relate to the list of specified roles. Each of the listed roles is assigned the access level that corresponds to its position in each list. For example, the second argument in the list associated with **rolesList** is assigned to the second argument associated **accessLevelList**.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh MapRolesToPage --username DASH_username --
password DASH_user_password --pageUniqueName page_unique_name --rolesList
role_name1, role__name2 --accessLevelList level1, level2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

RemoveRolesFromPage
Use the **RemoveRolesFromPage** command to disassociate a comma-separated list of roles with a specified page.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh RemoveRolesFromPage --username DASH_username --password DASH_user_password --pageUniqueName page_unique_name --rolesList role_name1, role__name2`
- Windows `tipcli.bat RemoveRolesFromPage --username DASH_username --password DASH_user_password --pageUniqueName page_unique_name --rolesList role_name1, role__name2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
page_unique_name is the page ID associated with the roles that you want to remove.
role_name1, *role__name2* is a comma-separated list of roles that are to be disassociated with the page.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh RemoveRolesFromPage --username DASH_username
--password DASH_user_password --pageUniqueName page_unique_name --rolesList
role_name1, role__name2 --accessLevelList level1, level2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

ListRolesForPortletEntity
Use the **ListRolesForPortletEntity** command to list all roles associated with a specified portlet.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh ListRolesForPortletEntity --portletEntityUniqueName portlet_entity_unique_name`
- Windows `tipcli.bat ListRolesForPortletEntity --portletEntityUniqueName portlet_entity_unique_name`

Where:

portlet_entity_unique_name is the unique ID for the portlet.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh ListRolesForPortletEntity --
portletEntityUniqueName portlet_entity_unique_name
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

MapRolesToPortletEntity
Use the **MapRolesToPortletEntity** command to associate a comma-separated list of roles with a specified portlet.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh MapRolesToPortletEntity --username DASH_username --password DASH_user_password --portletEntityUniqueName portlet_entity_unique_name --rolesList role_name1, role__name2 --accessLevelList level1, level2`
- Windows `tipcli.bat MapRolesToPortletEntity --username DASH_username --password DASH_user_password --portletEntityUniqueName`

portlet_entity_unique_name --rolesList *role_name1, role__name2* --
accessLevelList *level1, level2*

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
portlet_entity_unique_name is the unique portlet ID with which to associate with the list of roles.
role_name1, *role__name2* is a comma-separated list of roles that are to be associated with the portlet.
level1, *level2* is a comma-separated list of access levels that relate to the list of specified roles. Each of the listed roles is assigned the access level that corresponds to its position in each list. For example, the second argument in the list associated with **rolesList** is assigned to the second argument associated **accessLevelList**.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh MapRolesToPortletEntity --username
DASH_username --password DASH_user_password --portletEntityUniqueName
portlet_entity_unique_name --rolesList role_name1, role__name2 --
accessLevelList level1, level2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

RemoveRolesFromPortletEntity
Use the **RemoveRolesFromPortletEntity** command to disassociate a comma-separated list of roles with a specified portlet.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh RemoveRolesFromPortletEntity --username DASH_username --password DASH_user_password --portletEntityUniqueName portlet_entity_unique_name --rolesList role_name1, role__name2`
- Windows `tipcli.bat RemoveRolesFromPortletEntity --username DASH_username --password DASH_user_password --portletEntityUniqueName portlet_entity_unique_name --rolesList role_name1, role__name2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
portlet_entity_unique_name is the portlet ID associated with the roles that you want to remove.
role_name1, *role__name2* is a comma-separated list of roles that are to be disassociated with the portlet.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh RemoveRolesFromPortletEntity --username
DASH_username --password DASH_user_password --portletEntityUniqueName
portlet_entity_unique_name --rolesList role_name1, role__name2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

ListRolesFromUser
Use the **ListRolesFromUser** command to list all roles associated with a specified user.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh ListRolesFromUser --username DASH_username --password DASH_user_password --userID user_ID`
- Windows `tipcli.bat ListRolesFromUser --username DASH_username --password DASH_user_password --userID user_ID`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
user_ID is the unique ID for the user.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

`/opt/IBM/JazzSM/ui/bin/tipcli.sh ListRolesFromUser --username DASH_username --password DASH_user_password --userID user_ID`

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

MapRolesToUser
Use the **MapRolesToUser** command to associate a comma-separated list of roles with a specified user ID.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh MapRolesToUser --username DASH_username --password DASH_user_password --userID user_ID --rolesList role_name1, role__name2`
- Windows `tipcli.bat MapRolesToUser --username DASH_username --password DASH_user_password --userID user_ID --rolesList role_name1, role__name2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
user_ID is the unique user ID with which to associate with the list of roles.
role_name1, *role__name2* is a comma-separated list of roles that are to be associated with the user.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh MapRolesToUser --username DASH_username --
password DASH_user_password --userID user_ID --rolesList role_name1,
role__name2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

RemoveRolesFromUser
Use the **RemoveRolesFromUser** command to disassociate a comma-separated list of roles with a specified user ID.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh RemoveRolesFromUser --username DASH_username --password DASH_user_password --userID user_ID --rolesList role_name1, role__name2`
- Windows `tipcli.bat RemoveRolesFromUser --username DASH_username --password DASH_user_password --userID user_ID --rolesList role_name1, role__name2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
user_ID is the user ID associated with the roles that you want to remove.
role_name1, *role__name2* is a comma-separated list of roles that are to be disassociated with the specified user ID.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh RemoveRolesFromUser --username DASH_username
--password DASH_user_password --userID user_ID --rolesList role_name1,
role__name2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

ListRolesForView
Use the **ListRolesForView** command to list all roles associated with a specified view.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh ListRolesForView --viewUniqueName view_name`
- Windows `tipcli.bat ListRolesForView --viewUniqueName view_name`

Where:

view_name is the unique name for the view.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh ListRolesForView --viewUniqueName view_name
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

MapRolesToView
Use the **MapRolesToView** command to associate a comma-separated list of roles with a specified view and set an access level for each role.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh MapRolesToView --username DASH_username --password DASH_user_password --viewUniqueName view_name --rolesList role_name1, role__name2 --accessLevelList level1, level2`
- Windows `tipcli.bat MapRolesToView --username DASH_username --password DASH_user_password --viewUniqueName view_name --rolesList role_name1, role__name2 --accessLevelList level1, level2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.
view_name is the unique view name with which to associate with the list of roles.
role_name1, *role__name2* is a comma-separated list of roles that are to be associated with the view.
level1, *level2* is a comma-separated list of page access levels that relate to the list of specified roles. Each of the listed roles is assigned the access level that corresponds to its position in each list. For example, the second argument in the list associated with **rolesList** is assigned to the second argument associated **accessLevelList**.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh MapRolesToView --username DASH_username --password DASH_user_password --viewUniqueName view_name --rolesList role_name1, role__name2 --accessLevelList level1, level2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

RemoveRolesFromView
Use the **RemoveRolesFromView** command to disassociate a comma-separated list of roles with a specified view.

Syntax

This command has the following syntax:

- Linux UNIX `tipcli.sh RemoveRolesFromView --username DASH_username --password DASH_user_password --viewUniqueName view_name --rolesList role_name1, role__name2`
- Windows `tipcli.bat RemoveRolesFromView --username DASH_username --password DASH_user_password --viewUniqueName view_name --rolesList role_name1, role__name2`

Where:

DASH_username is the portal administrator user ID.
DASH_user_password is the password associated with the portal administrator user ID.

view_name is the unique view name associated with the roles that you want to remove.

role_name1, *role__name2* is a comma-separated list of roles that are to be disassociated with the specified view.

Note: Individual role name arguments to the **rolesList** parameter must not include spaces.

Example

Linux UNIX For example, in a UNIX or Linux environment, use the following command:

```
/opt/IBM/JazzSM/ui/bin/tipcli.sh RemoveRolesFromView --username DASH_username
--password DASH_user_password --viewUniqueName view_name --rolesList
role_name1, role__name2
```

Where */opt/IBM/JazzSM/* is location of the Dashboard Application Service Hub instance.

Working with views

`tipcli` commands for working with views.

The `tipcli` commands are entered in the *tip_home_dir*`/profiles/TIPProfile/bin` directory, for example, `C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat` on Windows or `/opt/IBM/JazzSM/ui/bin/tipcli.sh` on Linux or UNIX.

ListViews
List all views.

AddViewMembers --username *DASH_username* --password *DASH_user_password* --view *view_unique_name* [--members *members1, member2*] [--launchMembers *launch_member1, launch_member2*]
Add members or launch members for a specified view.

Important: When you add members to a view at the command line, your updates are not reflected in the portal until the next time that you log in.

ListViewsForRole --roleName *role_name*
List the views associated with a specified role.

MapViewsToRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --viewList *view_unique_name1, view_unique_name2* --accessLevelList *level1, level2*
Associate a comma separated list of views with a particular role and set the access level for the role for each view.

RemoveViewsFromRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --viewList *view_unique_name1, view_unique_name2*
Disassociate a comma separated list of views from a particular role.

Working with users

`tipcli` commands for working with users.

ListUsersFromRole --roleName *role_name*
List the users associated with a specified role.

MapUsersToRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --usersList *user_ID1*:*user_ID2*
Associate a colon (`:`) separated list of user IDs with a particular role.

Note: Arguments to the `usersList` parameter should not include a colon (`:`).

RemoveUsersFromRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --usersList *user_ID1*:*user_ID2*
Disassociate a colon (`:`) separated list of user IDs from a particular role.

Working with preference profiles

`tipcli` commands for working with preference profiles.

DeletePreferenceProfile --username *DASH_username* --password *DASH_user_password* --profileName *profile_name*

Delete the specified preference profile.

ListPreferenceProfiles [--name *profile_name*]

Return a list of console preference profiles. Optionally, you can specify a comma separated lists of preference profiles, to return their unique names.

ShowPreferenceProfile --uniqueName *profile_unique_name*

List all the attributes for a specified profile preference.

AddUpdatePreferenceProfile --username *DASH_username* --password *DASH_user_password* --profileName *profile_name* [--newProfileName *new_profile_name*] [--themeDir *theme_dir*] [--showNavTree true|false] [--componentDir default|ltr|rtl] [--textDir default|contextual|ltr|rtl] [--views *view_unique_name1, view_unique_name2*] --roles *role_name1, role_name2*] [--defaultView *view_unique_name*]

Use the `AddUpdatePreferenceProfile` command to create a new profile preference or update an existing profile.

Table 159. `AddUpdatePreferenceProfile` command arguments

Parameter and arguments	Description
`--username` *DASH_username*	Mandatory parameter. A user with the `iscadmins` role.
`--password` *DASH_user_password*	Mandatory parameter. The password for the user with the `iscadmins` role.
`--profileName` *profile_name*	Mandatory parameter. The name of the profile that is to be created or modified.
`[--newProfileName` *new_profile_name*`]`	Optional parameter. The new name for the specified profile.
`[--themeDir` *theme_dir*`]`	Optional parameter. Used to specify the directory for the theme that you want to apply.
`[--showNavTree true|false]`	Optional parameter. Used to specify whether or not you want the navigation pane to be displayed for preference profile.
`[--componentDir default|ltr|rtl]`	Optional parameter. Used to specify component display direction, that is, whether you want items to display left-to-right, right-to-left, or to use the default browser settings.
`[--textDir default|ltr|rtl]`	Optional parameter. Used to specify text direction, that is, whether you want text to display left-to-right, right-to-left, or to use the default browser settings.
`[--views` *view_unique_name1*`,` *view_unique_name2*`]`	Optional parameter. Used to specify the views that you want to assign to the preference profile. Comma separated list.
`--roles` *role_name1*`,` *role_name2*`]`	Optional parameter. Used to specify the roles that you want to assign to the preference profile. Comma separated list.

Table 159. AddUpdatePreferenceProfile command arguments (continued)	
Parameter and arguments	**Description**
[--defaultView *view_unique_name*]	Optional parameter. Used to specify the view that you want displayed when a user logs into the portal.

Working with portlets

tipcli commands for working with portlets.

The tipcli commands are entered in the *tip_home_dir*/profiles/TIPProfile/bin directory, for example, C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat on Windows or /opt/IBM/JazzSM/ui/bin/tipcli.sh on Linux or UNIX.

ListPortletEntitiesForRole --roleName *role_name*]
List the portlets entities associated with a specified role.

MapPortletEntitiesToRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --portletEntityList *portletEntity_unique_name1, portletEntity_unique_name2* --accessLevelList *level1, level2*
Associate a comma separated list of portlets with a particular role and set the access level for the role for each portlet.

RemovePortletEntitiesFromRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --portletEntityList *portletEntity_unique_name1, portletEntity_unique_name2*
Disassociate a comma separated list of portlets with from particular role.

Working with pages

tipcli commands for working with pages.

ListPages [--viewList *view_unique_name1, view_unique_name2*] [--customizePages true|false]
List all pages. You can optionally filter the list by using the viewlist parameter and providing a comma separated list of views. You can also use the customizePages (set totrue) to return a list of custom pages only.

ListPagesForRole --roleName *role_name*
List the pages associated with a specified role.

MapPagesToRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --pageList *page_unique_name1, page_unique_name2* --accessLevelList *level1, level2*
Associate a comma separated list of pages with a particular role and set the access level for the role for each page.

RemovePagesFromRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --pageList *page_unique_name1, page_unique_name2*
Disassociate a comma separated list of pages from a particular role.

Working with user groups

tipcli commands for working with user groups.

The tipcli commands are entered in the /opt/IBM/JazzSM/ui/bin directory, for example, C:\Program Files\IBM\JazzSM\ui\bin\tipcli.bat on Windows or /opt/IBM/JazzSM/ui/bin/tipcli.sh on Linux or UNIX.

ListGroupsFromRole --roleName *role_name*
List the user groups associated with a specified role.

MapGroupsToRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --groupsList *group_name1*: *group_name2*
Associate a colon (:) separated list of groups with a particular role.

Note: Arguments to the `groupsList` parameter should not include a colon (`:`).

RemoveGroupsFromRole --username *DASH_username* --password *DASH_user_password* --roleName *role_name* --groupsList *group_name1*: *group_name2*
Disassociate a colon (`:`) separated list of groups from a particular role.

Charting `tipcli` commands

`tipcli` commands for working with charting.

ListCharts --username *DASH_username* --password *DASH_user_password*
Use `ListCharts` to review the charts that are configured in the environment.

ChartConnection --action *action* [--name *name*] [--protocol *protocol* --hostname *hostname* --port *port* -- serviceName *serviceName* --username *username* --password *password*--renderFormat *render_format* --Datasource_Username *datasource_username* --credentialType *credential_type*] --username *DASH_username* --password *DASH_user_password*
`ChartConnection` is used to configure a connection to any IBM Tivoli Charting Web Service. The ITM Web Service is just one example.

ChartExport --dir *output_directory* --type all|customcharts|page [--pageID *page_ID* | --pageName *page_name*] --username *DASH_username* --password *DASH_user_password*
`ChartExport` is used to export chart data.

Table 160. `ChartExport` command arguments

Parameter and arguments	Description
`--dir` *output_directory*	Mandatory parameter. The directory where the exported data is saved. If the directory does not exist, it is created.
`--type all|customcharts|page`	Mandatory parameter. If you set the `--type` to `all`, then all charts are exported. If you set it to `customcharts`, then only customized charts are exported. If you set it to `page`, then you can use either the `--pageID` or the `--pageName` parameter to specify the page for which you want to export chart data.
[`--pageID` *page_ID* | `--pageName` *page_name*]	Optional parameter. If you set the `--type` parameter to `page`, then you can use either the `--pageID` or the `--pageName` parameter to specify the page for which you want to export chart data.
`--username` *DASH user name*	Mandatory parameter. The user name for a user with either the `chartAdministrator` or `chartCreator` role.
`--password` *DASH user password*	Mandatory parameter. The password for the specified user name.

ChartImport --dir *source_directory* --username *DASH_username* --password *DASH_user_password*
`ChartImport` is used to import chart data from a specified directory.

Table 161. `ChartImport` command arguments	
Parameter and arguments	**Description**
`--dir` *`source_directory`*	Mandatory parameter. The directory where the data to be imported is located. BIRT Designer file format is `.rptdesign`.
`--username` *`DASH user name`*	Mandatory parameter. The user name for a user with either the `chartAdministrator` or `chartCreator` role.
`--password` *`DASH user password`*	Mandatory parameter. The password for the specified user name.

`ChartProperties [--name` *`property_name`* `--value` *`property_value`*`] --username` *`DASH_username`* `--password` *`DASH_user_password`*

`ChartProperties` is used to view or modify properties for charting. If you only provide username and password details and no other arguments, then the current properties are listed. It is useful to run this command first so that you can review the current property names and values before you decide to make updates.

Table 162. `ChartProperties` command arguments	
Parameter and arguments	**Description**
`--name` *`property_name`* `--value` *`property_value`*	Optional parameter. The name of the property that you want to update and the value that you want to set. For example, to set the timeout value to 10,000,000 milliseconds, enter `--name AXIS_TIMEOUT --value 10000000`.
`--username` *`DASH user name`*	Mandatory parameter. The user name for a user with the `chartAdministrator` role.
`--password` *`DASH user password`*	Mandatory parameter. The password for the specified user name.

`ListRestoreTimestamp`

Use the `ListRestoreTimestamp` command to return a list of charting store backups by timestamp.

`RestoreChartStore --BackupTimestamp` *`backup_timestamp`* `--username` *`DASH_username`* `--password` *`DASH_user_password`*

Use the `RestoreChartStore` command to restore a chart store by timestamp.

Table 163. `RestoreChartStore` command arguments	
Parameter and arguments	**Description**
`RestoreChartStore --BackupTimestamp`	Mandatory parameter. The timestamp of the charting store backup.
`--username` *`DASH user name`*	Mandatory parameter. The user name for a user with the `chartAdministrator` role.
`--password` *`DASH user password`*	Mandatory parameter. The password for the specified user name.

Dashboard Application Service Hub Export commands

Use these `tipcli` commands for to export Dashboard Application Service Hub customized data.

tipcli *- Export plugins*

Use the `Export` command to export customization data for an instance of Dashboard Application Service Hub. Use the `ListExportPlugins` command to list plugins that are available for export.

Syntax

`ListExportPlugins`

Use the `ListExportPlugins` command to list all plugins that can be exported. Use the list of returned plugins to assist you when you are specifying plugins to be exported.

`Export [--includePlugins|--excludePlugins` *`plugin1,plugin2`*`] [--settingFile` *`setting_file`*`] --username` *`DASH_username`* `--password` *`DASH_user_password`*

Parameters

If you provide no parameters to the `Export` command, all custom data is exported by default.

Note: If you specify additional parameters for the `tipcli.bat|.sh Export` and make a typing error, that is, if you type a parameter incorrectly, or use the incorrect case, then the commands runs as if no parameters were specified and no warning message is displayed.

Table 164. `Export` parameters and arguments

Parameter and arguments	Description
`[--includePlugins|--excludePlugins` *`plugin1,plugin2`*`]`	Optional parameter. You can choose to include or exclude a list of plugins when you run the `Export` command.
`[--settingFile` *`setting_file`*`]`	Optional parameter. You can specify your export requirements in properties file instead of specifying your requirements using separate parameters at the command line. Provide a path to the settings file as the argument to the `settingFile` parameter. On systems running Windows you must use double backslashes characters (\\) when specifying the path to your settings file, for example, `C:\\tmp\\export.properties`. Command line parameters take precedence over entries in the settings file.
`--username` *`DASH user name`*	Mandatory parameter. The user name for a user with the `iscadmin` role.
`--password` *`DASH user password`*	Mandatory parameter. The password for the specified user name.

Example 1 - Return a list of plugins available for exporting

The following example returns a list of plugins that can be exported:

Windows
```
C:\Program Files\IBM\JazzSM\ui\bin>tipcli.bat ListExportPlugins
```

Example 2 - Export a subset of available plugins

The following example exports the CMS plugin only:

```
Windows C:\Program Files\IBM\JazzSM\ui\bin>tipcli.bat Export
--includePlugins com.ibm.tivoli.tip.cli.cms.CmsExportPlugin
--username tbsmadmin --password DASHpassword
```

tipcli - Advanced Export options

Use the ExportPagePlugin tipcli command to export specific Dashboard Application Service Hub data.

Note: If you specify additional parameters for the tipcli.bat|.sh Export and make a typing error, that is, if you type a parameter incorrectly, or use the incorrect case, then the commands runs as if no parameters were specified and no warning message is displayed.

Export [--exportFile *export_file*] [--pages ALL|NONE|*page1,page2*] [--views ALL|NONE|*view1,view2*] [--roles ALL|NONE|REQUIRED|*role1,role2*] [--exportPagesInViews true|false] [--userPreferences ALL|NONE|REQUIRED|*user_ID1,user_ID2*] [--consolePreferenceProfiles ALL|NONE|*pref_ID1,pref_ID2*] [--includeEntitiesFromApp *war1,war2*] [--includeCustomData true|false] [--includeCredentialData true|false] [--includeMytasks true|false] [--includeMyStartupPages true|false] [--includeTransformations true|false] --username *DASH_username* --password *DASH_user_password*

Table 165. ExportPagePlugin command arguments

Parameter and arguments	Description
[--exportFile *export_file*]	Optional parameter. Specifies the path and file name for the exported data, for example, c:/tmp/extest.zip.
[--pages ALL|NONE|*page1,page2*]	Optional parameter. If you do not use the pages parameter, the default setting is ALL unless either exportPagesInViews or includeEntitiesFromApp is defined, then the default setting is NONE. You can also provide a list of pages that you want to export.
[--views ALL|NONE|*view1,view2*]	Optional parameter. If you do not use the views parameter, the default setting is ALL. You can also provide a list of views that you want to export and optionally specify that you want to export all pages associated with the specified views. **Note:** Whether the optional parameter exportpageinviews is set to true or false, if a view has a default node in the navigation pane associated with it, then the page associated with the node is always exported. This is also true, even if you specify NONE as the argument to the --pages parameter.
[--roles ALL|NONE|REQUIRED|*role1,role2*]	Optional parameter. You can export no roles, all roles, or a specific list of roles. The default setting is ALL unless the pages parameter or the includeEntitiesFromApp parameter is specified. Then, the default setting is set to REQUIRED.

Table 165. ExportPagePlugin command arguments (continued)	
Parameter and arguments	**Description**
[--exportPagesInViews true|false]	Optional parameter. Use this parameter, set to true, to export the pages associated with an exported view . The default value is false.
[--userPreferences ALL|NONE|REQUIRED|*user_ID1*,*user_ID2*]	Optional parameter. You can export preferences for all users, no users, or for a specified list of users by user ID. The default setting is ALL. This parameter overrides the includeMytasks and includeMyStartupPages parameters.
[--consolePreferenceProfiles ALL|NONE|*pref_ID1*,*pref_ID2*]	Optional parameter. You can export no preference profile data, all preference profile data, or data for a specific list of preference profiles. The default setting is ALL. **Note:** If a console preference profile has a custom view as its default view, then that view is automatically exported. If the exported view has a default node in the navigation pane, then the associated page is automatically exported with the view.
[--includeEntitiesFromApp *war1*,*war2*]	Optional parameter. You can provide a list of WARs to export pages that contain portlets associated with the listed WARs.
[--includeCustomData true|false]	Optional parameter. The default value is true. If is set to false, no customization data is exported.
[--includeCredentialData true|false]	Optional parameter. The default value is true. If is set to false, no credential data is exported.
[--includeMytasks true|false]	Optional parameter. The default setting is true. This parameter only applies when the includeEntitiesFromApp parameter is also specified.
[--includeMyStartupPages true|false]	Optional parameter. The default setting is true. This parameter only applies when the includeEntitiesFromApp parameter is also specified.
[--includeTransformations true|false]	Optional parameter. The default setting is true.
--username *DASH user name*	Mandatory parameter. The user name for a user with the iscadmins role.
--password *DASH user password*	Mandatory parameter. The password for the specified user name.

tipcli - Charting Export options
Use the ChartExportPlugin tipcli command to export Dashboard Application Service Hub chart data.

Note: If you specify additional parameters for the tipcli.bat|.sh Export and make a typing error, that is, if you type a parameter incorrectly, or use the incorrect case, then the commands runs as if no parameters were specified and no warning message is displayed.

Export [--includeCharts ALL|NONE|*page_ID1,page_ID2*] --username *DASH_username* --password *DASH_user_password*

Table 166. ChartExportPlugin command arguments	
Parameter and arguments	**Description**
[--includeCharts ALL|NONE| *page_ID1,page_ID2*]	Optional parameter. You can export all charts, no charts, or specify a list of charts to be exported. The default setting is ALL. **Note:** If you run the Export command using the --includeCharts parameter, it must be run by the same user that started the Dashboard Application Service Hub Server.
--username *DASH user name*	Mandatory parameter. The user name for a user with the chartAdministrator role.
--password *DASH user password*	Mandatory parameter. The password for the specified user name.

Import tipcli commands

tipcli commands for importing Dashboard Application Service Hub data.

Note: If you specify additional parameters for the tipcli.bat|.sh Import and make a typing error, that is, if you type a parameter incorrectly, or use the incorrect case, then the commands runs as if no parameters were specified and no warning message is displayed.

ListImportPlugins

Use the ListImportPlugins command to list all plugins that are available to be imported.

Import [--includePlugins|--excludePlugins *plugin1,plugin2*] [--settingFile *setting_file*] [--backupDir backup_dir] --username *DASH_username* --password *DASH_user_password*

Use the Import command to import customization data into a Dashboard Application Service Hub environment. If you provide no parameters to the Import command, all custom data is imported by default.

Table 167. Import command arguments	
Parameter and arguments	**Description**
[--includePlugins|--excludePlugins *plugin1,plugin2*]	Optional parameter. You can choose to include or exclude a list of plugins when you run the Import command.
[--settingFile *setting_file*]	Optional parameter. You can specify your import requirements in a properties file instead of specifying your requirements using separate parameters at the command line. Provide a path to the settings file as the argument to the settingFile parameter. On systems running Windows you must use double backslashes characters (\\) when specifying the path to your settings file, for example, C:\\tmp\ \\import.properties. Command line parameters take precedence over entries in the settings file.
[--backupDir backup_dir]	You can specify a directory to save the backup data during an import operation so that if it is required you can subsequently restore settings.

Table 167. `Import` command arguments (continued)	
Parameter and arguments	**Description**
`--username` *DASH user name*	Mandatory parameter. The user name for a user with the `iscadmin` role.
`--password` *DASH user password*	Mandatory parameter. The password for the specified user name.

Additional commands

Additional `tipcli` commands.

`cmsUpdateRemoteEntries [--username` *`username`* `--password` *`password`*`] (-toremote | -fromremote | -deleteremote) [-force]`

Save system information in the file specified.

Table 168. `cmsUpdateRemoteEntries` command arguments	
Parameter and arguments	**Description**
[`--username` *`username`* `--password` *`password`*]	Optional parameters. User name and password for a Dashboard Application Service Hub user. If you do not provide user name and password details at the command line, you must enter the user name and password in an interactive mode.
`-toremote`	Optional parameter. Indicates that the update is to occur to the remote data store, that is, the local information is to be written to the remote database.
`-fromremote`	Optional parameter. Indicates that the update is to occur from the remote data store. Any information saved locally is downloaded and updated from the remote database.
`-deleteremote`	Optional parameter. Indicates that the launch entries provided by this Dashboard Application Service Hub instance to the remote database is to be deleted from the database. Additionally, this command prevents any further updates from being sent to the remote database. On execution, the `cmsUpdateRemoteEntries` command with the `toremote` and `force` options updates the database and re-enables automatic updates to the remote database. **Note:** There is no difference between `deleteremote` with the `force` option and `deleteremote` without the `force` option.
`-force`	Optional parameter. Indicates that any caching or optimization mechanisms for the data should be ignored and that the data should be updated regardless of the state.Any existing cached information is discarded. All data in the database is refreshed for the `toremote` case, including the resource bundles.

`Version`

List the versions of the products and components installed in the environment.

SystemInfo [--outputFile *outputFile*]
Save system information in the file specified.

ITMLogin --hostname *hostname* --port *port* --username *username* --password *password* [--servicename]
ITMLogin is used to configure the ITM Web Service to connect to the Tivoli Enterprise Portal Server. For example, this command in Windows configures the username and password for a new ITM Web Service to be added to the application server instance.

```
C:\IBM\tivoli\tip\bin\tipcli.bat ITMLogin --hostname
localhost --port 1920 --username sysadmin --password
sysadm1n --servicename ITMWebService2
```

You can use the ITMLogin command to change the hostname, port, username, and password of an existing Tivoli Enterprise Portal Server instance. Changing a configured ITM Web Service to a different Tivoli Enterprise Portal Server is not supported, because the two portal servers may have different configurations. If you need to use a different portal server, you can install another instance of the ITM Web Service and use this command (along with the -serviceName option) to configure.

TADDMLogin --hostname *hostname* [--port *port*] --username *username* --password *password*
Log in to the Tivoli Application Dependency Discovery Manager.

Appendix A. 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.

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.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

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.

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Part Number:
Product Number:

(1P) P/N:

SC27-8782-03