Oracle® Enterprise Manager

Business Transaction Management Online Help

Release 12.1.0.6

E52792-01

June 2014

ORACLE®

Oracle Enterprise Manager Business Transaction Management Online Help, Release 12.1.0.6

E52792-01

Contents

4   Discovering Services and Dependencies

5   Working with Transactions

6 Monitoring Services

7 Working with Consumers

8   Configuring and Viewing Alerts

9   Creating Reports

10   Commands and Scripts

11   Utilities

12   Administration of Business Transaction Management

Index

Preface

Oracle Real User Experience Insight User's Guide explains how you use Business Transaction Management 12*c* Release 5 (12.1.0.6) to discover services, define and track transactions, monitor performance, define service level agreements, monitor failures and perform root-cause analysis to locate errors and improve performance.

Audience

This document is intended for developers and system administrators interested in developing and managing distributed applications.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Overview of Oracle Business Transaction Management

This chapter introduces Business Transaction Management. It describes the following:

- Using Business Transaction Management
- New Features and Changes
- Business Transaction Management Architectural Overview

If you are upgrading from an earlier version, please see Section 1.4, "Functional Upgrade Issues," for more information.

1.1 Using Business Transaction Management

Using Business Transaction Management, a component of the Oracle Management packs, you can do the following:

- discover the services deployed in your application environment and understand how they work together

  For a complete and up-to-date list of the types of services and components that Business Transaction Management can discover and monitor, refer to the Business Transaction Management Certification Matrix. You can locate this document by searching for "BTM certification" at `http://support.oracle.com`.

- define and track transactions, and understand the business context of each transaction, including the transaction's consumers and business payload
- monitor the performance of services and transactions in real time
- define service-level agreements to set performance targets based on business criteria and to monitor performance relative to those targets
- generate alerts that warn you of unusual conditions
- monitor failures and perform root-cause analysis to locate bottlenecks, errors, and incomplete transactions

For a discussion of the workflow involved in working with Business Transaction Management, see Chapter 2, "Getting Started with Business Transaction Management."

You access the Business Transaction Management feature set by way of the Management Console. For information about using the console, read Chapter 3, "Using the Management Console."

You can also use a command line interface (CLI) to execute scripts and individual commands. For information about using the CLI, see Chapter 10, "Commands and Scripts."

It is also possible to access some Business Transaction Management functionality from the Enterprise Manager console. See "New Features and Changes" for more information.

1.2 New Features and Changes

This section describes the new features added and the changes made since the last release.

- **SOA_B2B** probe – Monitors SOA B2B (Business to business) adapters of SOA applications.

- **Out of band discovery** - enables dynamic discovery period for when an observer attempts to discover new components. For more information on this feature, enabled in the advanced settings for observers, see Section 12.1.2.11, "Advanced Settings Field Reference."

- **SOA 12.1.3** - Business Transaction Management is now compatible with Oracle SOA Suite Release 12.1.3.

- **SOA_REST adapter** -Monitors SOA REST adapters of SOA applications (SOA Release 12.1.3 only)

- **OSB 12c** - Business Transaction Management is now compatible with Oracle Service Bus Release 12c. Install the WebLogic 12 observer to use this feature.

For a complete and up-to-date list of the types of services and components that Business Transaction Management can discover and monitor, refer to the Business Transaction Management Certification Matrix. You can locate this document by searching for "BTM certification" at `http://support.oracle.com`.

1.3 Architectural Overview

At the highest level, Business Transaction Management consists of three types of components:

- **Central servers** – The central servers are application EAR files that you deploy to an application server. There are three central servers. You deploy only one instance of each of these servers, and for performance considerations you should deploy each to a separate application server. You must not deploy any of the central servers to an application server that hosts services or components you intend to monitor. The central servers are as follows:

  - **Main Server** (btmMain.ear) – Contains all the central Business Transaction Management system services and user interface applications, including the *sphere*. The sphere is the Business Transaction Management component that manages the Business Transaction Management environment. In addition, btmMain.ear contains a subdeployment for the F5 intermediary.

  - **Performance Server** (btmPerformanceServer.ear) – Contains the service-level management components. Deploy btmPerformanceServer.ear on an application server other than where btmMain.ear or btmTransactionServer.ear are deployed.

  - **Transaction Server** (btmTransactionServer.ear) – Contains the transaction management components. Deploy btmTransactionServer.ear on an application

server other than where btmMain.ear or btmPerformanceServer.ear are deployed.

- **Observers** – Observers are sets of libraries that you install into the application server that hosts the business applications you want to monitor. The observers monitor messages and calls between the components of your applications. Observers are capable of monitoring many types of components, and are classified according to the type that they monitor, for example, JavaEE, OSB, WCF, etc.

- **Monitors** (btmMonitor.ear) – Monitors collect application performance and usage measurements from observers. The monitor is an application EAR file that you deploy to an application server. For large systems, you can deploy multiple monitors, either as singletons or replicates. For performance reasons, you should not deploy the monitor on an application server where the central servers are deployed.

Business Transaction Management also requires access to an Oracle RDBMS for storing performance measurements, logging messages, and maintaining the environment model and Business Transaction Management configuration.

The diagram below shows a typical distributed application environment, and the relationship of the Business Transaction Management components to that environment.

Figure 1–1   Deployment of Business Transaction Management components in a typical application environment
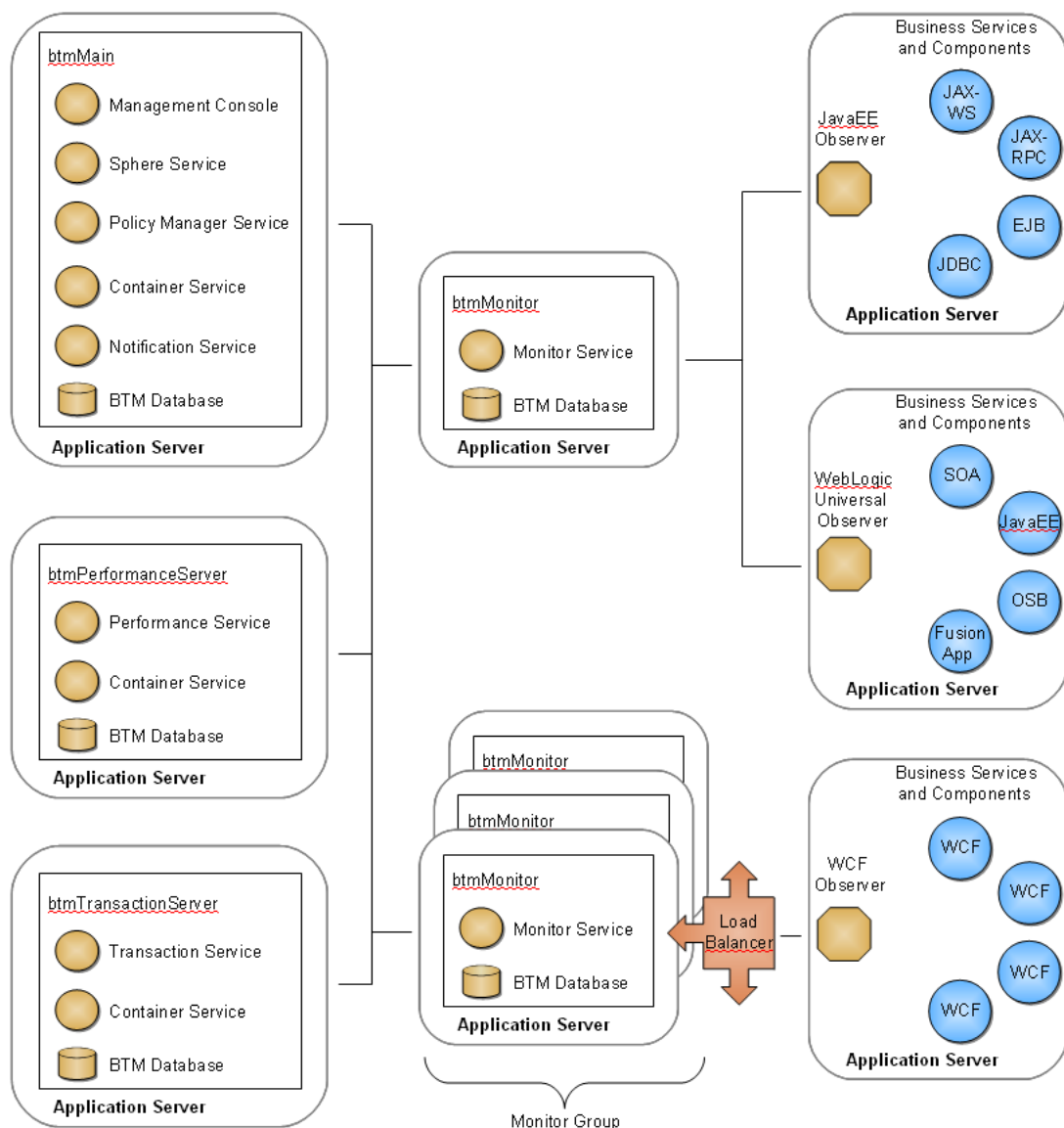
Business Transaction Management is designed for use in a distributed application environment in which the various Business Transaction Management components are deployed onto multiple machines and application servers.

Technically, you can install all the central servers into a single application server, but such a deployment scenario is not recommended for production environments. Installation in a single application server can be useful for demonstrations and for learning how to use the product, but this scenario might not scale successfully with a large number of business services or high volume of message traffic, just to name a few factors.

We recommend that you deploy each of the central servers to separate application servers. The Performance and Transaction components, in particular, typically perform a large amount of performance analysis computations. Dividing processes across application servers allows you to control memory and processor resources.

You should also deploy the monitor to an application server separate from the central servers. Depending on your monitoring requirements, you might need to deploy multiple monitors. You can deploy monitors either as singletons or as replicates behind a load balancer. For information about replicating the monitor, refer to the *Business Transaction Management Installation Guide*.

Observers must always be installed outside the application server hosting the central servers or monitors.

1.4 Functional Upgrade Issues

Business Transaction Management aims to provide the simplest process possible for upgrading from the 12.1.0.2, 12.1.0.3, 12.1.0.4, or 12.1.0.5 release.

The upgrade process seeks to minimize downtime while preserving management configurations, object definitions, and existing operational data.

For a detailed description of the process used to back up and upgrade your system, please consult the *Business Transaction Management Installation Guide*.

This section describes the changes introduced in this upgrade that might affect the ability of Business Transaction Management to monitor your transactions and to secure communications. You should read through these sections to make sure you understand how changes to the product might affect your transaction definitions and the ability of Business Transaction Management to observe and monitor your transactions. After you upgrade, we recommend that you run some traffic and audit your transaction instances to make sure the system is observing and monitoring your transactions as you expect it to. If it is not, you might need to adjust observer communication policies, update transaction definitions, or add manual keying between nodes as appropriate to your situation.

If, after reading through the following subsections, you are still unclear about the implications of upgrading to the 12.1.0.5 release, please open a service request with Oracle Technical Support.

1.4.1 Effect of Universal Observer for WebLogic 10.3 Servers

In releases previous to 12.1.0.3, a variety of observers were provided for installing into WebLogic 10.3 servers. Each type of observer contained a set of probes that gave it the ability to monitor a particular set of component types. For example, the JavaEE observer contained probes for monitoring JavaEE components and the SOA observer contained probes for monitoring SOA components. Starting with release 12.1.0.3, only a single "universal" observer is provided for installing into WebLogic 10.3 servers. This universal observer contains a superset of all the probes contained in the older observers, making it capable of monitoring the superset of component types that were previously monitorable.

After you upgrade to the universal observer, your system will have the ability to discover and monitor new types of components. This enhanced capability might result in the discovery of new objects and could impact your ability to view and monitor transactions as explained below.

If you have a transaction A -> B -> C, and the newly observed operation is B1 (the resulting call chain is A -> B -> B1 -> C), two outcomes are possible, depending on the means of correlation used:

■ Auto-correlation: Operation C will be shown in gray. If there is a missing message condition defined from B to C, you will be alerted. You will need to redefine the transaction to make C visible and to have the transaction be monitored. In those

rare cases where the message from A to B to C does not change, the system will continue to observe and monitor C.

- Manual-correlation: Operation C will be visible and correlated to B. The newly observed operation B1 will be visible in the dependency graph, but not in the transaction definition. You will need to redefine the transaction to make B1 visible. In this case, the transaction will be monitored, but B1 will not be included in it.

During the upgrade process, the values in existing observer communication policies are carried forward. This means that if an existing policy has most of its probe fields activated and you upgrade to the WebLogic universal observer and the application server hosts component types that weren't discovered before but are of a type for which there is now an activated probe, then these will be discovered and monitored.

To limit the number of components discovered and monitored to those of interest, before upgrading, you might want to review and adjust the active probes in your observer communication policy. To check your work and determine the changes introduced by the use of the universal observer, you should audit all transactions that contain operations running on WebLogic 10.3 containers.

Information about observers is provided in Section 12.1.1, "About Observers." Instructions for upgrading observers are provided in the *Business Transaction Management Installation Guide*.

1.4.2 Effect of New Default Setting for Limiting Discovery

Depending on the underlying technology, some messages flow directly from a client to a service endpoint; others flow through a host of intermediate endpoints before they reach their actual destination. Such intermediate endpoints might comprise the implementation of a messaging system, a job scheduling system, a distributed system, and so on.

When installing probes for technologies that use intermediate endpoints, Business Transaction Management allows you to specify whether you want to monitor all endpoints or just the endpoints at the edge of such systems; often these are the endpoints that directly represent the business services of interest. In the current release, by default, the monitoring of intermediary endpoints is turned off. This improves monitoring performance and eliminates data that is not essential to monitoring your distributed applications.

If you are upgrading from a system where the monitoring of intermediate endpoints was turned on, Business Transaction Management retains the knowledge of these endpoints after upgrading to the new default setting. You will be able to see these endpoints in the dependency graphs; they will be grayed out in the transaction graphs. You can edit your transaction definitions to remove these endpoints.

As the default setting implies, we recommend turning off the monitoring of intermediate endpoints.

For more information on limiting discovery, see Section 4.1.3, "Limiting Discovery."

1.4.3 Effect of New Authentication Settings

The current release of Business Transaction Management introduces observer authentication and sets SSL on by default for communication between observer and monitor. These features are described in the *Business Transaction Management Installation Guide*. In the current release, if you create a new observer communication policy, client authentication and SSL is on by default.

Upgrading your system should not affect expected behavior in most cases. Combining old observers with new communication policies is the one case that will affect communication between observers and monitors as explained below.

- An existing observer is communicating with a monitor that was not using SSL. When you upgrade, nothing changes. The new observers will behave like the old observers. To change behavior, edit the observer communication policy to add SSL or observer authentication.

- An existing observer is communicating with a monitor over SSL. When you upgrade, this will continue to work the same way. If you wish to add client authentication, you must edit the observer communication policy to do so.

- You have an old observer in the field, and you upgrade the central server and monitors. If you then create a new observer communication policy that applies to the old observer, the old observer will fail because it will not be able to authenticate to the monitor, and the monitor will not be able to recognize anything the old observer is sending to it. If a new observer is available and you want to use client authentication, install the new observer. If a new observer is not available, turn off authentication in the observer communication policy.

1.4.4 Effect of New Probes

The release 12.1.0.6 universal observer for WebLogic 10.3 provides these new probes:

- **SOA_B2B** – Monitors SOA B2B (Business to business) adapters of SOA applications.

The release 12.1.0.5 universal observer for WebLogic 10.3 provides these new probes:

- **SOA_JCA** – Monitors SOA JCA adapters including AQ, Database, File, FTP, JMS, MQ Series, Socket, and Oracle Applications.

The release 12.1.0.4 universal observer for WebLogic 10.3 provides these new probes:

- **SOA_WORKFLOW** – Monitors Oracle SOA Human Workflow components and notifications.
- **SOA_DIRECT** – Monitors direct binding calls between SOA composites and between SOA and Oracle Service Bus.

After upgrading to an observer that contains new probes, the probes are immediately available in all new Observer Communication policies that you create. However, the probes are not automatically added to your existing policies. If you want to use one of the new probes in an existing policy, you must manually add it to the policy. For information on how to add a probe to a policy, see Section 12.1.2.3, "Adding Probes."

Once activated, the new probe types will discover new types of components. If any of these newly discovered components are called within the flow of a transaction that you have defined, then complete monitoring of the transaction will no longer be possible. Monitoring of the transaction will end at the newly discovered component.

In order to monitor the complete transaction, you must edit the transaction in one of this ways:

- Return the transaction to its original state by manually keying the disjoined nodes back together.
- Enlarge the transaction by adding the newly discovered component as a new node in the transaction and then connecting it to the disjoined nodes.

1.4.5 Effect of New JMS Option in the Model Configuration Settings

Upgrading your central servers and monitors adds a new JMS option to the **Model Configuration** section of the **Advanced Settings** for all new Observer Communication policies that you create (existing policies are not affected). By default, this new JMS option is set to **Model Edge of Flow**. At this setting, only JMS topic message listeners are modeled. If you want to model all JMS topics, JMS queues, and their associated message listeners, then you must edit the JMS option and set it to **Model All**.

1.4.6 Effect of Improved JMS Probe

Release 12.1.0.4 provides a new and significantly improved JMS probe for WebLogic that replaces the JMS probe found in earlier releases. The new JMS probe is contained in these observers:

- Universal Observer for WebLogic 10.3

- JavaEE Observer for WebLogic 12.1

After you upgrade the central servers and monitors, they will be compatible with both the new and old JMS probes. You are not required to upgrade your observers, and, if you don't, they will continue to perform as before. You are also permitted to run a mixed system in which some of your observers are upgraded and some are not. However, we highly recommend that you upgrade all of your observers.

Once upgraded, your observers will rediscover the JMS components in your system and model them as new types of JMS services. Queues and topics are modeled as services of type **JMS Resource**, and message listeners are modeled as services of type **Message Listener**. Your old JMS services (discovered by the old JMS probe) will continue to be displayed but will not register business traffic flowing through the system. These old services are of type **JMS Message Consumer**. You should unregister these old JMS services.

Transactions that include any of the old JMS services will cease to function correctly. You should delete these old transactions and redefine them using the newly discovered JMS services.

2

Getting Started with Business Transaction Management

This chapter describes a basic workflow for working with Business Transaction Management. It also serves as an introduction to the major sections of the online help. You should read this chapter before you proceed to the rest of the book.

To view a visual demonstration on how you can use BTM, access the following URL and click Begin Video:

https://apex.oracle.com/pls/apex/f?p=44785:24:539253248237801::NO:24:P2
4_CONTENT_ID,P24_PREV_PAGE:6366,1

2.1 Basic Workflow

The basic workflow for working with Business Transaction Management includes the following steps:

- Run Traffic

- Discover Services and Dependencies

- Identify Areas of Interest

- Define Transactions

- Assess Monitoring Needs for Transactions

- Create Scripts

2.1.1 Run Traffic

The Business Transaction Management discovery process is entirely traffic-based. If messages are not flowing through the observed endpoints, the system cannot discover any application components nor can it discover the dependencies between these components. So, the first step in working with Business Transaction Management is to run traffic and allow a little time to elapse to give the system a chance to observe message traffic and to build a picture of your deployed system.

If this is the first time you are using Business Transaction Management, the next thing you will want to do is to look at the Management Console and get a sense of its basic parts and how it is organized. If you have run some traffic, you will be able to see services and their dependencies when you open the Management Console. Take a few minutes to get acquainted with the basic workings of the console:

- "Getting to Know the Management Console" describes the parts and organization of the Management Console

- "Viewing Data" provides a more detailed explanation of how you use the Management Console to view different types of data. It introduces the basic elements used to present performance data: maps, charts, tabular views, and dashboards. You might want to look at this now, or refer to it later as you view and interpret performance information.

2.1.2 Discover Services and Dependencies

After you run traffic, Business Transaction Management immediately starts gathering data to give you a complete picture of your environment and to help you understand the flow of work through it. Identifying the components that make up a distributed application and understanding how these components relate to one another allows you to answer two questions: "What do I have?" and "What is it doing?

Business Transaction Management uses tabular and graphical views to display topological and performance information. The following screen shot illustrates how endpoint dependencies, status, and performance data are represented in one graphical view.
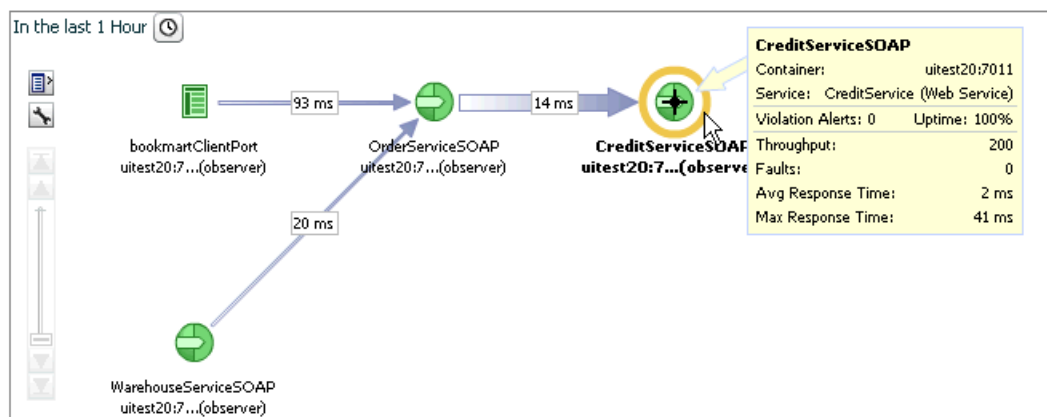
When you first look at your environment, you might want to know about the location and status of the containers running the observed components. Next, you might want to view services to see that all the services you are interested in observing have been discovered. You can then look at how services are interacting with one another and how traffic is distributed among them in the case of replicated services.

It might now be necessary for you to make adjustments to the picture the system has constructed by registering services that could not be found, by resolving replication or duplication issues, by addressing versioning problems, and so on.

2.1.3 Identify Areas of Interest

Having formed an accurate and complete picture of the services operating in your environment, you can now turn to look at operational data. By default, the system displays data for throughput, traffic, faults, average response time, maximum response time, and uptime. This should allow you to identify bottlenecks, faulty components, slow components, and unusually light or heavy traffic.

You can narrow the number of services displayed by using filters. You might also want to customize your view by adding instruments to the core measurements displayed in tabular views.

Using this information, you could decide to change the topology of your deployment: replicating some services, adjusting the processing capacity of others, and taking a closer look at faulty components.

2.1.4 Define Transactions

A transaction is a sequence of operations that you want to monitor as a single unit. Business Transaction Management focuses on the monitoring of transactions to help you identify and resolve issues related to performance, to profiling usage, and to identifying the cause of failing components in a business process.

Based on the dependencies revealed by discovery, you can define a transaction to include services whose interactions interest you. You might exclude services that are not under your control, or you might decide to include ancillary services that you believe have an appreciable effect on performance. Defining a transaction is simple: you identify the beginning and end operation of a sequence of operations. You can then enable additional features that allow you to look at your transaction from different perspectives.

2.1.5 Assess Monitoring Needs for Transactions

By default Business Transaction Management simply monitors the basic performance of a transaction. You can modify the default transaction definition to enable monitoring features that provide exactly the level of detail you need about performance and usage. For example you can add features that allow you to do the following:

- Segment performance information by host address or by usage. Usage might be organized by user, by supplier, or by any criteria that helps you get useful performance information.

- Log transaction instances

- Log message content, which then allows you to search logged messages for specific information. For example, "which message had an order for 15,000 books?"

In addition to the performance information that is routinely gathered and that you can view at all times, you can have Business Transaction Management alert you in the following special cases:

- You define a service level agreement based on some fixed value or a historical baseline, and you ask to be notified whenever performance crosses defined thresholds. For example, you want to know when throughput for the transaction dips below 100 requests in the last ten minutes.

- You define a condition that tests for message content or for a certain fault, and you ask to be notified if this condition is met. For example, you ask to be notified when the value of a property associated with quantity ordered is greater than 1,000.

- You define a condition that tests for the arrival of a message within a certain period, and you ask to be notified if the message does not arrive. You might use this to alert you about transactions that do not complete.

As you can see, Business Transaction Management offers a variety of options in monitoring your transactions. Which of these monitoring techniques you use depends on the questions you want to answer and on the issues you're trying to resolve. These questions might change as you proceed from development, to testing, to production. By creating a system that allows you to selectively enable and disable monitoring features, you can be sure to get exactly the information you need while minimizing the management system's performance costs.

2.1.6 Create Scripts

Finally, as you learn to work with Business Transaction Management and identify processes that you want to archive and execute at will, particularly administrative tasks, you might want to create a script using the command line interface. You might use such scripts to do the following:

- Configure your system
- Resolve discovery problems
- Register services
- Resolve replication problems
- Migrate data
- Generate reports

3

Using the Management Console

This chapter describes the process of working with the Business Transaction Management console; it includes the following sections:

- Getting to Know the Management Console
- Viewing Data
- Creating New Views
- Management Console Reference

3.1 Getting to Know the Management Console

The Business Transaction Management console displays information about the objects known to the sphere and allows you to access tools for monitoring and managing these objects. The information is presented using different types of *views*; these include dashboards, charts, maps, graphs, and tables. This section introduces the main areas of the console: the Navigator, the main area, the tabs area, and the main menu.

When you first install Business Transaction Management, the console offers you a set of default views of the objects in the sphere. You can create additional views and save these views to make it easier to access the objects and instruments that interest you. In addition, you can use the custom data explorer to view any object in the system in relationship to any other object.

After you have read through this section, continue reading through the rest of this chapter to learn about the following:

- controls used in each type of view
- interpreting default views
- modifying views
- creating new views

In addition to using the console, you can also use the Business Transaction Management Command Line Interface (CLI) to issue management commands from the command line or through the use of scripts. You use CLI commands to configure the system, manage DNS aliases, resolve replication issues, register services, manage metadata, control monitoring, apply policies, migrate data, or generate data needed for reporting.

3.1.1 General Console Design

The figure below shows the main areas of the management console.
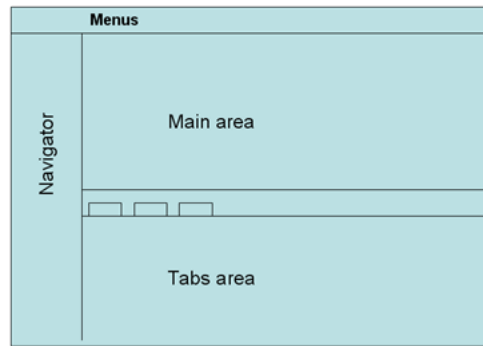
The following sections describe each area in some detail. Here is a brief summary:

- The **Navigator** provides a list of views that are available by default. Selecting one of these items displays all objects of this type in the main area.

- The **main area** displays the view selected in the navigator. Double clicking any item in the main area opens the tabs area if it is not already opened, and displays detailed information about that item. The main area includes controls at the top that you can use to filter the objects displayed.

- The **menus** provide actions that you can perform on the item selected in the main area, or it provides actions unrelated to the selected item, like creating a new object, or performing some administrative action.

- The **tabs area** provides additional information about the item selected in the main area.

3.1.2 The Navigator

The navigator is a facilitator that allows you to quickly get to the objects and relationships that interest you. The default navigator sections are the following (they are normally displayed in their expanded form):
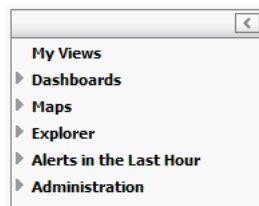
The navigator includes two kinds of controls:

- A minimizing control (arrow) in the upper right hand corner. Click this control to collapse the Navigator, which allows more room for the main area.

- Expansion controls (triangles) to the left of each view name that you can click to expand and view category contents.

Note that each item in the Explorer view contains a numeric value in parenthesis; this indicates the number of such objects known to the sphere. If you were to filter the corresponding main view to show fewer items, the number shown in the Navigator would remain unchanged.

The table below summarizes the contents of the default Navigator categories.

Category	Description
My Views	A list of any views you have modified or created that you have saved.
Dashboards	A dashboard is a heterogeneous view comprised of smaller views, typically used to summarize information of interest. It provides a summary view of the state of the system and monitored objects. By default, the console provides three dashboards: one that summarizes the operational health of the system, and two that provide summary information for the top ten services and transactions.
Maps	Graphical views of services and their dependencies, and of containers.
Explorer	Tabular views of monitored and non-monitored objects and their relationships. Also includes the custom data explorer, which allows you to view any object in the sphere in relation to other objects.
Alerts in the last hour	Tabular views of all alert types: SLA, condition, and system.
Administration	Tabular views of system service containers, system services, system policies, monitors, and unassigned endpoints.

3.1.3 Main Area

For any item selected in the Navigator, except those in the **Dashboards** and **Maps** categories, the main area displays a tabular view and, just above the table, a series of controls that you can use to filter the columns and the number of items shown. Here is an example of what is shown in the main view when Containers is selected in the Navigator.

Use the up/down control in the lower right hand corner of the main area to open and close the tabs area. You can control the size of the main area by moving the slider control bordering the main area and the tabs area.

3.1.4 Menus

Menu items relate directly to the item currently selected in the main area or the tabs area. You might think of the items in the main or tabs area as the nouns and of the menu items as the verbs. Thus, selecting a menu item will result in taking some action that affects the item selected in the main or tabs view.

Although menu items are never hidden, they might be disabled if the operation cannot be applied to the current selection, if the currently selected object is not in an appropriate state, or if the user is not authorized to operate on the target. Look here for additional information about user roles.

3.1.5 Tabs Area

The tabs area is organized into tabbed panels. The tabs shown vary with the item selected in the summary area.

Use the arrow control in the lower right hand corner of the main area to collapse the tabs area. You can expand it again either by double clicking an item in the main area or by clicking the up arrow at the bottom right hand corner of the summary pane.

3.1.6 Object Detail Window

The console provides access to an inspector window or a separate window that includes the tabs area for the selected object. This view is available for any object or area that shows a tear off control (magnifying glass). Clicking the control opens a separate window with additional detail information about the selected item. The information shown varies with the selected object.

3.2 Viewing Data

All data known to the sphere about the services, endpoints, and operations that make up your application are presented using different types of views: graphical views, charts, tabular views, and dashboards. You can view this information using the format that makes it easiest for you to find and interpret data. In addition, you can use a variety of controls to filter and sort the data to highlight selected aspects of application performance.

This section describes the controls and views used to display data. It begins by describing controls that are available for all types of views and continues to describe the characteristics of each type. It covers the following topics:

- Viewing Controls

- Viewing Maps

- Viewing Charts

- Viewing Tables

- Viewing Dashboards

3.2.1 Viewing Controls

Data is displayed in the console using graphs, tables, charts, and dashboards. This section describes the controls that are available no matter how you view data. Controls specific to a single type of view are described in the section for that type.

For all types of views, you can use controls to filter the information displayed, to add or delete columns in a table or to change time intervals when data is reported; you can also use controls to display additional information or to display information in a separate window.

3.2.1.1 Filtering Controls

You can use filtering controls to filter service maps and tabular data. Business Transaction Management provides two kinds of filters: deep and shallow. Not all views provide both types. The figure below illustrates these controls.

Deep filter   Shallow filter

Deep filters modify a view by re-fetching data from the sphere according to the constraints you specify using the filter control. *Shallow filters* simply limit what is

currently shown in the console. So, for example if you used a shallow (Look for) filter to show only items that have "Order" in their name, the current view would be refreshed to show you only those items.

You can define deep filters in two ways:

■ Select one of the pre-defined filters from the drop-down list. The choices listed in the drop-down list include those filtering criteria most commonly needed for the object selected in the Navigator.

■ Select the **Choose Filter...** item from the drop down list to open the Filter tool and specify the desired criteria.

The filter tool allows you to filter objects according to their salient characteristics (attributes); these will differ depending on the object selected in the Navigator: services, containers, policies, devices, consumers, and so on. If the Filter tool does not list an attribute, that means you cannot filter a view based on that attribute.

For more information about the Filter tool, see Section 3.4.3, "Using the Filter Tool."

3.2.1.2 Column Chooser and Sizing Control

Controls for modifying tabular displays (by adding or deleting columns) and for resizing columns are shown just above table views. The figure below shows the column chooser and its drop down list. It also shows the *sizing control*, which allows you to resize columns so that all columns are visible in the table.
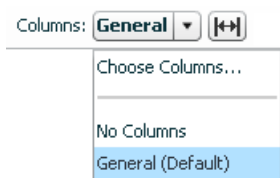
Choose **No Columns** to display no columns. If you select **Choose Columns....** Business Transaction Management displays a tool you can use to add or delete columns.

3.2.1.3 Time Interval Control

The time interval control allows you to specify the interval during which instrument data is fetched and displayed. This control is shown in different locations and has a different effect depending on where it is set. It is displayed either as a clock icon or a drop-down list labeled **Time Period**.
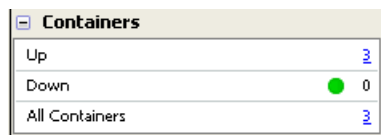
■ In the column chooser (Instruments tab), the control allows you to specify the intervals over which you want to get instrument data. You can choose this interval to be dynamic, or you can choose fixed periods of 10 minutes, 1 hour, 1 day, or 7 days. If you specify "dynamic," the interval is set by the time interval control shown in the main pane.

■ In the main area, if displayed, the time interval control allows you to specify the intervals for which instrument data is reported. You have the choice of 10 minutes, 1 hour, 1 day, or 7 days.

■ In the Analysis pane, which gives you the finest granularity, you can look at measurements taken in the last fixed period of 10 minutes, 1 hour, 1 day, or 7 days; you can look at measurements taken since a given date and time; or you can look at measurements taken between two specified dates and times.

■ In the Summary pane you can look at measurements taken in the last fixed period of 10 minutes, 1 hour, 1 day, or 7 days; you can also use it to set a time range.

Note that for day and week intervals, the interval is expanded so that you get at least as much data as you ask for. For example, if today is 11:20 a.m. on 12/22/10, and you pick **last 1 day**, data is returned from 12/21/10 11:00 until now. Similarly, if you pick **last week**, data is returned from 12/15/10 00:00 till now.

3.2.1.4 Pop-up Links and Inspectors

Some items in tables are underlined to indicate that they are links you can click to open an inspector. For example, in the Operational Health Summary dashboard, the number specifying up containers is such a link.

Click the link to open an inspector window that displays additional information about containers that are currently up.

3.2.1.5 Tear-off Controls

The looking glass icon, indicates that you can open a new window showing information about the selected object, including the tabs area.

3.2.2 Viewing Maps

A map is a good way to represent relationships; maps provide a visual representation of how services, endpoints, or operations are related.

■ To view all services and their dependencies, select **Maps > Service Map** from the navigator.

■ To view containers, select **Maps > Container Map** from the navigator.

■ To view dependencies between a subset of related services, endpoints, or operations, select the object of interest and click the **Dependency** tab.

■ To view dependencies between the operations that make up a transaction, select the transaction and click the **Summary** tab.

The following figure shows a map that presents endpoint dependencies.
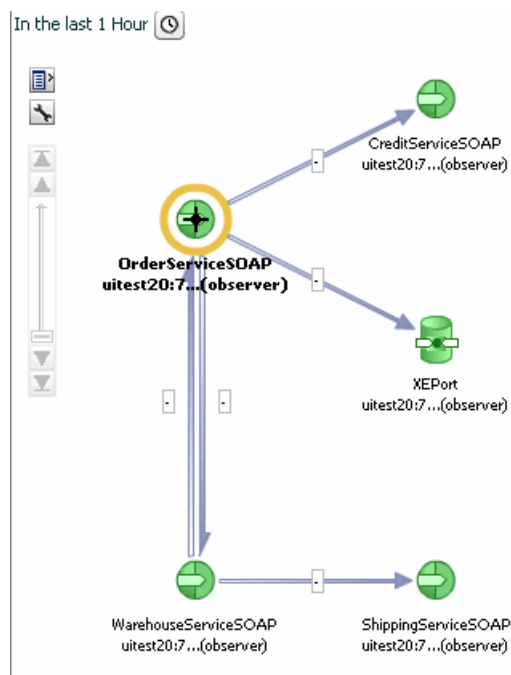
Because map views focus on relationships, Business Transaction Management attempts to present as clean a picture as possible so as not to obscure these relationships. Additional information is usually available if you move the cursor over the links that connect objects or if you hover over the object itself. In this case, hovering over an endpoint icon displays information about the endpoint's container, service, and core instrument values.

All map views also provide specialized controls that you can use to get additional information or to filter available information. These are described in the following sections.

3.2.2.1 Displaying Tabular Data for a Map

If you click the table icon in any map view, Business Transaction Management displays the corresponding inventory in a table to the left of the map.

 The tabular tree view allows you to see one or more layers above or below the objects shown in the graphic. Note the shallow filter at the top of the display; you can use this to further filter the contents of the table.

To close the tabular tree view, click the table icon again.
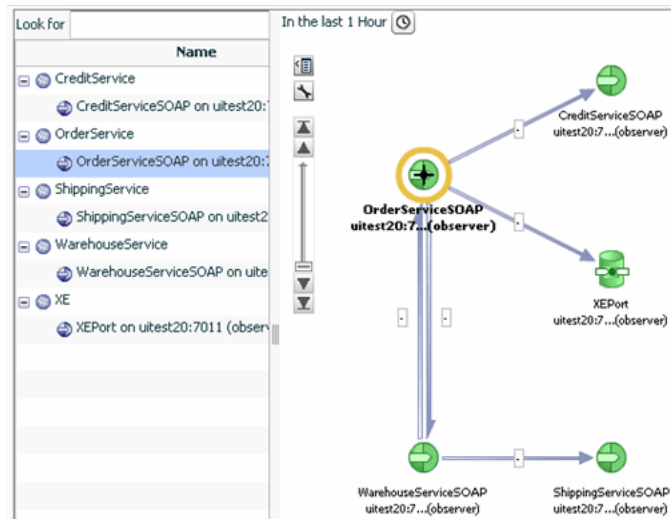
3.2.2.2 Filtering and Adjusting the Map View

If you click the wrench icon in any map view, Business Transaction Management offers several controls.

- The **Layout** control allows you to choose the direction and type of layout used.

- The **Show Related** control allows you to display additional objects that are only indirectly related to the objects whose dependencies are shown. These might be objects that you might want to manually add to the map by registering them.

- In some maps, this control might also allow you to filter the current view.

3.2.2.3 Scaling a Map View

The scaling icon allows you to expand and shrink the currently displayed map.

Slide the scaling bar up and down to resize the map to the desired size. If scaling does not produce the desired results, you can try using the region control, described next.

3.2.2.4 Focusing on Different Regions of a Map

If a map cannot fit in the current window, Business Transaction Management displays a thumb view control in the lower right hand corner of the map window. You use this control to reposition the map so that you can view all its branches.

To use this control, click on the white rectangle and drag it to position the map as desired. Use the triangle icon in the lower right hand corner to collapse or open the thumb-view control.

3.2.3 Viewing Charts

Charts provide a graphic presentation of how instrument values change over time. The following chart displays the response time for a transaction. The chart is accompanied by a table that serves both as a key and as a tabular presentation of the charted data. You can get more information from charts by passing the cursor over the graphed instrument values. As you pass the cursor over the nodes in the chart, Business Transaction Management displays instrument values for the point in time associated with that node.
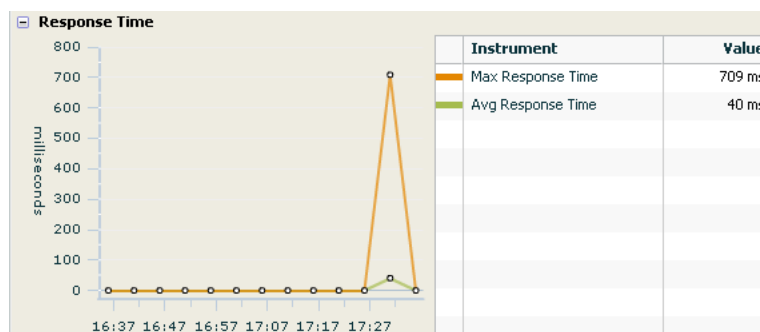
3.2.4 Viewing Tables

Tables are shown in the main area of the console.

Controls situated above the table allow you to filter the contents of the table, to change the columns shown, or to re-size columns to fit data. You can also sort columns, change column widths, and move columns.

- To view sorting options for a given column, pass the cursor over the column heading. A pop-up will inform you of your options.

- To change column widths, place the cursor on the line separating column headings until the resizing icon appears, then drag the icon to change the width of the column.

- To move columns in different positions, click the column heading and drag to the desired position.

Any changes you make to a table will be lost when you close or refresh your browser. To save a modified view, you must choose **View > Save current view** from the menu. After naming and saving the view, it will be listed under **My Views** in the navigator.

3.2.5 Viewing Dashboards

Dashboards are containers that include graphical and tabular elements. They are used to give you a snapshot of some aspect of your system: transaction performance, top ten services, and operational health. You can display dashboards by selecting one from the **Dashboards** in the Navigator.

3.3 Creating New Views

You can create new temporary or permanent views in the following ways:

- You can modify existing tables by adding or deleting columns, rearranging columns, and resorting columns.

- You can create new tables to focus on objects and relationships for which there are no default views.

The views you create are available only for the current session. But you can save a view for future use. Once saved, your custom views are shown in the **My Views** category of the Navigator.

3.3.1 Modifying Tables

Controls for modifying tabular displays (by adding or deleting columns) and for resizing columns are shown just above table views. The figure below shows the column chooser and its drop-down list. It also shows the sizing control, which allows you to resize columns so that all columns are visible in the table.
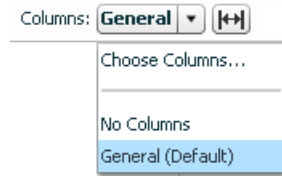
Choose **No Columns** to display no columns. If you select **Choose Columns...**. Business Transaction Management displays a tool you can use to add or delete columns.
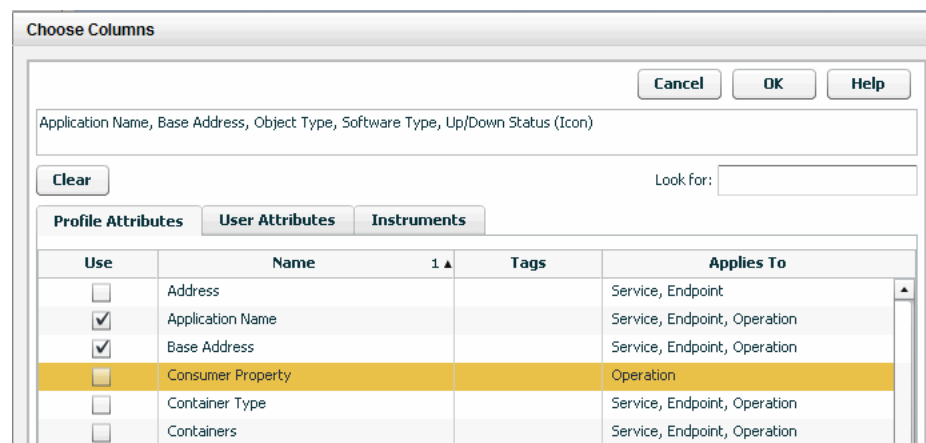
The **Choose Columns** tool shows the attributes that are currently chosen from any of the three tabs: Profile Attributes, User Attributes, and Instruments.

The top box lists all columns that are currently chosen from any of the three tabs: Profile Attributes, User Attributes, and Instruments.

You can select columns to add or subtract in three tabbed categories. To start, click a tab, and use the **Look for** filter to narrow the options shown if needed.

■ To add a column click the box in the **Use** column for the desired attribute or instrument.

■ To remove a column click the checked box in the **Use** column for the desired attribute or instrument.

User attributes are used only in graphic displays. They dictate whether clients are shown for endpoints or services.

The time control displayed for the **Instruments** tab applies to all instruments chosen, you can select more than one interval for instruments.

When you are satisfied with your changes, click **OK**. Look over the resulting view to make sure you have the information you need. You might still want to move or resize columns before saving this as a permanent view.

■ To view sorting options for a given column, pass the cursor over the column heading. A pop-up will inform you of your options.

- To change column widths, place the cursor on the line separating column headings until the resizing icon appears, then drag the icon to change the width of the column.

- To move columns in different positions, click the column heading and drag to the desired position.

Any changes you make to a table will be lost when you close or refresh your browser. But you always have the option to save a view you have created.

3.3.2 Using the Custom Data Explorer to Create a View

By default, the console displays objects and relationships thought to be most useful to most users. If the default views do not include the objects or relationships that interest you, use the *custom data explorer* to display any object known to the system and its relationships to other associated objects. Using this information, you can create your own custom views.

To use the Custom Data Explorer, choose **Explorer > Custom Data Explorer** from the Navigator.

The custom data explorer uses the main area with which you are already familiar. At the top of the main area, it includes some new controls: Two drop down lists allow you to select an object and one or more associated objects to include in a tree view.

- The list of possible primary objects is quite long; use the scroll bar to find the object of interest. For a sample view, **Transaction** is chosen as a primary object.
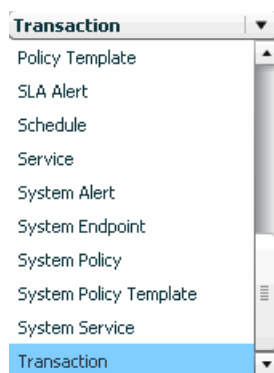
- The list of possible associated objects, to the right of object list, contains all the objects that the selected object can relate to. For example, the **Transaction** object can relate to the following:
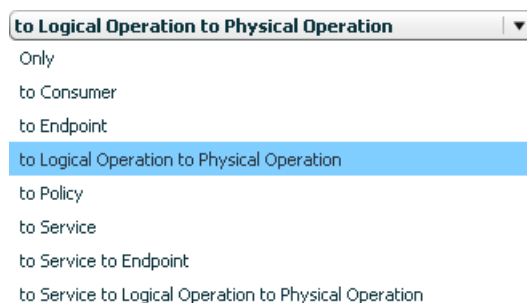
- Choosing **Only** displays no related objects. Choosing a related object displays that object below the primary object in a tree view.

Once you have chosen an item from both lists, the main view will display a tree view with the primary object at the top and one or more subordinate objects underneath. For the sample view, **Transactions** is chosen as the primary object, and **to Logical Operation to Physical Operation** is chosen for the associated objects.

You could now use the Column Chooser tool to display attributes and instruments for these primary and associated objects.

When you are done, you can save the view to make it permanent. This process is described in the next section.

3.3.3 Saving a View

To save a view that you have created by modifying an existing view, do the following:

1. With the view in focus, select **View > Save current view as** from the main menu.

2. If you want this to be a new view, specify its name. This is the name used to reference the view in the **My Views** category in the Navigator.

3. If this will replace a view by the same name, click the check box **Replace the view if it already exists?**

4. Click **OK.**

The view you saved will be displayed in the Navigator in the category **My Views**. To set the view as a default view, select **Set current view as default** from the **View menu.**

3.4 Management Console Reference

This section provides reference information about using controls and tools that are common to many views. It describes the following controls and tools:

- Using the Analysis Tab
- Health Summary Dashboards
- Using the Filter Tool
- Adding a Custom Attribute for Filtering
- Keyboard Shortcuts

3.4.1 Using the Analysis Tab

The **Analysis** tab displays detailed current performance and usage information for the selected object, either a transaction, service, endpoint, operation, or consumer. Because some types of information are relevant to only certain types of objects, the layout of the Analysis tab can vary from object to object.

The **Analysis** tab is composed of a number of panes that you display one at a time. Each pane displays a different type of information. The set of available panes changes depending on the object you select (in other words, not all panes are available for every type of object). The panes are described in this section.

3.4.1.1 Performance Pane

When you select a transaction, this pane provides started transactions, completed transactions, condition alerts, average response time, and maximum response time instruments. These instruments are displayed in a **Count chart**, **Count table**, **Response Time chart**, and **Response Time table**.

When you select a service, endpoint, operation, or consumer, this pane provides traffic, throughput, faults, fault percentage, average response time, and maximum response time instruments. These instruments are displayed in a **Messages chart**, **Messages table**, **Response Time chart**, and **Response Time table**.

The blue line in the **Count chart** indicates the number of transactions that started; the green line, the number of transactions that completed; and the red line, the number of condition alerts that occurred during the associated time segment, for example:
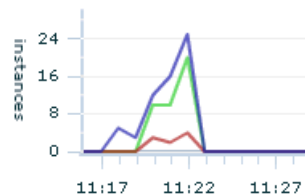
The **Count table** displays these measurements numerically.

The total height of the bars in the **Messages chart** indicates the traffic observed during the associated time segment. The green portion of the bar indicates the throughput, and the red portion indicates the fault count, for example:
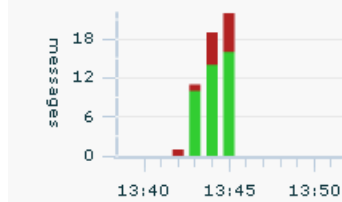
The **Messages table** displays these measurements numerically, with the addition of the fault percentage measurement. Clicking a hyperlinked value in the table opens the **Message Log Search tool** and runs a predefined query to return the messages associated with the measurement value. For example, if you want to examine the messages responsible for the **Traffic value**, click the value:

Instrument	Value
Faults	13
Throughput	40
Traffic	53
Fault %	24

The green line in the **Response Time chart** indicates the average response time for the associated time segment; the orange line indicates the maximum response time, for example:
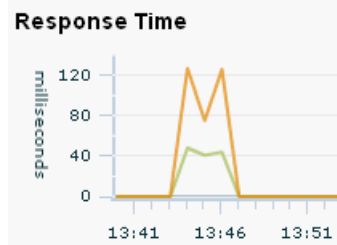
The **Response Time table** displays these measurements numerically.

3.4.1.2 Transaction Performance

Displays transaction performance measurements associated with the selected consumer. See the description of the Performance pane for details concerning the charts and tables.

3.4.1.3 Service Performance

Displays service performance measurements associated with the selected consumer. See the description of the Performance pane for details concerning the charts and tables.

3.4.1.4 Consumer Usage

Displays throughput, faults, fault percentage, average response time, and maximum response time measurements for the selected object, segmented by consumer. The table lists all consumers of the selected object and the aggregated performance measurements associated with each consumer's use of the object.

3.4.1.5 Transaction Usage

Displays started transactions, completed transactions, average response time, and maximum response time measurements associated with the selected consumer. These measurements are segmented by transaction.

3.4.1.6 Service/Endpoint Usage

Displays throughput, faults, and fault percentage associated with the selected consumer. These measurements are segmented by service and endpoint.

3.4.1.7 Condition Alerts

Lists information about the condition alerts that have been triggered in a given time period: the name of the condition that was met, the endpoint where the condition alert was triggered, and the number of condition alerts triggered.

3.4.1.8 Breakdown by Client Address

Displays throughput, faults, and fault percentage for the selected object, segmented by the client address, which is the machine host name from which the request was sent. The table lists all client addresses that sent requests to the selected object and the aggregated performance measurements associated with each client address's use of the object.

3.4.1.9 Callouts

Lists each operation (qualified by endpoint) to which the selected object made an outbound call. The **Operation column** identifies the operation that made the outbound call. Aggregated link throughput, link faults, and link average response time is displayed for each type of outbound call.

3.4.1.10 Custom Charting

Lets you set up a customized chart and table similar to the Performance pane, but with instruments of your choosing.

Click **Choose Instruments** and select the instruments you want displayed in the chart and table. You can select multiple instruments. When you set up a custom chart/table for a transaction, it is available for any selected transaction, likewise for consumers, and for services/endpoints/operations.

3.4.1.11 Custom Breakdown

Lets you set up a custom table of numeric instruments segmented in various ways.

Click **Choose Instruments** and select the instruments you want displayed in the table. You can select multiple instruments. Click **Choose Segments** and select how you want to segment the measurements. You can select multiple segments.

For example, you might set up a table that displays the number of started and completed transactions per consumer, where your instruments are started transactions and completed transactions and your segment is Consumer Name:

Consumer Name 1 ▲	Started Transactions	Completed Transactions
acapone@sanquentin.gov	4	4
bphillips@nfl.org	48	21
bsimpson@fox.net	10	10
drbaxter@science.edu	23	23
jdoe@acme.com	5	5
jsmith@edgility.com	20	20
milhouse@whitehouse.gov	14	14
Unknown	15	15

3.4.2 Health Summary Dashboards

The **Admin Health Summary** dashboard enables you to quickly assess the health of your Business Transaction Management system components. The **Operational Health Summary** dashboard provides that same information plus information about the health of your business components.

To display the Health Summary dashboards

- Choose **Dashboards** > **Admin Health Summary**

- Choose **Dashboards** > **Operational Health Summary**

These dashboards are composed of multiple tables. Each table provides summary information about a particular type of object, such as transactions, services, SLA alerts, and so forth. The rows in each table generally indicate a status that the object can be in, and the numerical value indicates how many instances of the object is in that state. For example, in the following case, 5 services are in a state of failure and 5 are in a state of warning.

The clock icon at the far left indicates that the numeric value is relative to the **Time Period** selected at the top of dashboard. Click the magnifying glass next to a value to pop up a list of the individual objects in that state.

3.4.3 Using the Filter Tool
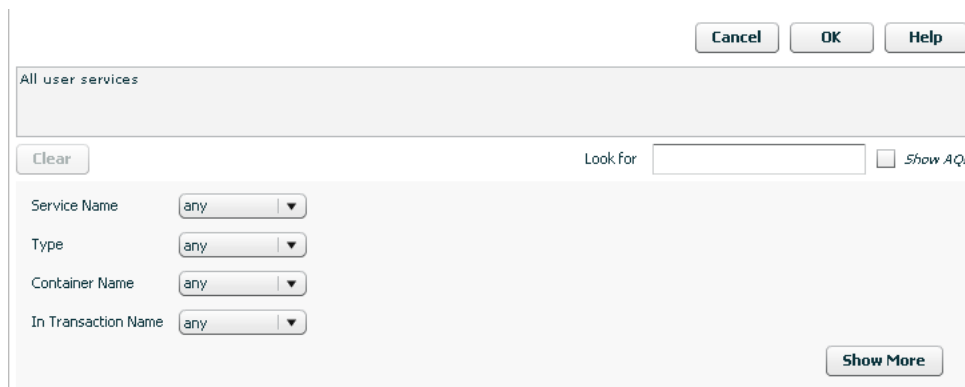
The Filter tool allows you to filter objects according to their salient characteristics, which varies according to the item chosen in the Navigator. Although the basic structure of the Filter tool does not change as the target object changes, the criteria you can select does change depending on whether the filter is used to modify the view of services, containers, policies, consumers, and so on.

In addition to the predefined filters, which are shown in a drop list next to the **Filter** link, you can also click on the link to open the Filter tool and obtain a richer choice of filtering criteria.

Note: You cannot filter condition alerts, observers, or schedules.

To open the filter tool, click on the **Filter** link in any view where it is displayed.

Business Transaction Management displays a dialog like the following:

The tool initially presents a short form that allows you to define a filter that references the most common criteria for the object of interest. The domain of objects for which you are creating a filter is displayed in the text box. For example, in the figure above, the text box informs you that you are creating a filter for all user services. To display additional criteria, click the **Show More** button.

As you select criteria for a query, either from the short form or from the longer form (**Show More**), Business Transaction Management translates your choices into an AQL expression that it uses to search the database. You can see that AQL query statement in the text box at the top of the tool by clicking the **Show AQL** check box.

You can use this feature to define AQL expressions that you can pass to CLI commands that use the `-filterQuery` flag. Simply use the filter tool controls to define your criteria, then copy and paste the AQL expression into the CLI command. Remember to enclose the query in quotation marks if it includes spaces.

When you are done selecting the criteria of interest, click the OK button to use this filter to modify the current view. See Section 3.3.3, "Saving a View" for instructions on saving your view.

3.4.4 Adding a Custom Attribute for Filtering

You can add a custom attribute for any object in the sphere (service, endpoint, container, and so on). This enables you to use the filter tool to find objects that have specific attribute values. For example, you might want to assign priority levels to services and then filter services based on their priority.

Note that unlike default attribute labels, user-defined attribute labels are not automatically translated.

The basic process for adding and using custom attributes for filtering objects is as follows:

1. Add a custom attribute for the object type of interest.

2. Edit an individual object's profile to assign a value to the custom attribute.

3. Use the Filter tool to select only those objects whose custom attributes have the desired value.

3.4.4.1 Adding the Attribute

To add a custom attribute, do the following:

1. Navigate to **Admin > Edit Data Model Attributes**. The Edit Data Model Attributes tool is displayed.

2. In the Settings area, find the object to which you want to add a custom attribute.

3. Click the **add attribute** link.

4. Provide the following information. Only the name and type are required.

    Name: Specify the name of the attribute.

    Type: Select the type of the attribute.

    Form Label: This is the name that will be used to identify the attribute in the **Profile** tab and in the **Filter** tool. You only need to specify a value if you want the label to be different from the name of the attribute.

    Query Label: This is the name that will be used to identify the attribute in the Query text box in the filter tool. You only need to specify a value if you want the label to be different from the name of the attribute.

    Value: Click the **add value** link to provide a suggested value for the attribute. This value is then displayed in the **Profile** tab for the selected object. (Optional)

    Description: Add a description to remind of the use or purpose of this attribute. (Optional)

5. Click **Apply**

3.4.4.2 Assigning an Attribute Value

To assign values to the custom attribute, do the following:

1. Open the **Profile** tab for the object to which you have added a custom attribute. The attribute you have added should be shown in the **Attributes** area of the **Profile** tab.

2. Select **Edit Profile for <Object>** from the **Modify** menu.

3. Specify a value for the custom attribute in the adjoining text box.

4. Click **Apply**. The value you entered should now be shown in the Profile tab.

3.4.4.3 Filtering Based on a Custom Attribute

To filter objects based on a custom attribute, do the following:

1. From the **Filters** drop list, select **Choose Filter...**

2. Expand the attributes link for the object of interest. For example, Service Attributes or Container Attributes.

3. The custom attribute you have created for the object should be listed along with any default attributes defined for the object. If it is not, refresh your browser.

4. Use the drop down list for the custom attribute to specify a search criteria.

5. Click **OK**. All objects matching the specified criteria will be listed in the main area.

3.4.5 Keyboard Shortcuts

You can use keyboard shortcuts to navigate the Business Transaction Management management console. These shortcuts are supported for Microsoft Internet Explorer 8.

For general information on using shortcuts with Internet Explorer, see the information provided for viewing and exploring web pages at the following site:

```
http://windows.microsoft.com/en-US/windows-vista
                                /Internet-Explorer-8-keyboard-shortcuts
```

This section provides additional detail on how to use keyboard shortcuts in the various areas of the management console.

As noted in Section 3.1.1, "General Console Design," the management console is divided into four sections: the menu bar, the navigator, the main area, and the tabs area. In addition, it is possible to access and use different tools while working with the console. Because these areas are created using different underlying technologies, keyboard shortcuts are not necessarily consistent. Please note the differences and limitations for each area; these are described in the following subsections.

General Tab Order

The Tab key is the basic means of navigating through the console. (Tab to go forward; Shift-Tab to go back.) The general tab order is the following: top explorer controls, console menu, log out and refresh controls, console Navigator items, main area controls (filters and column chooser), main area items, tabs area.

Menus

Use the following guidelines to work with management console menus:

- Use tab to move across the menus (shift-tab to move backward).

- Press **Enter** to open a menu.

- Use the up/down arrow keys to move forward/backward through menu items once the menu is open.

- Use the right arrow key to navigate through layered menus.

- Press **Enter** to execute the currently selected menu item.

The Navigator

Use the following guidelines to work with Navigator items:

- Use the Tab key to reach the Navigator.

- Use Tab to move forward and Shift-Tab to move backward through the Navigator. without highlighting items.

- Use the up/down arrows to navigate directly from category to category.

- Use the right/left arrows to navigate sequentially through categories and views.

- To expand or collapse a Navigator category, tab to the category and press **Enter**.

- To select a Navigator item, tab to the item and press **Enter**.

The Main Area

To access the main area, move the focus to the last item in the navigator and press the Tab key. This should move the focus to the Filter control in the main area.

To access items in the grid view in the main area, tab until the focus moves through the main area controls (Filters, column chooser ... and highlights the top line of the grid view.) When the top line of the grid view has the focus, use CTRL-down arrow to move the focus to the first item listed in the grid view.

- To move through items in the grid view in the main area use the up/down arrow key.

- To expand an item, move the focus to that item and press Shift-CTRL-right arrow. You can then use the up/down arrows to move linearly through grid items.

- To open the tabs area for an item, place the focus on the item and press **Enter**.

The Tabs Area

To work in the tabs area you must display the area and then be able to navigate across tabs and into the content of a specific tab.

To display the tab area you can do one of the following:

- Place the focus on the item for which you want to display detailed information, and press **Enter**.

- Tab to the **View** menu and select **Show/Hide Details Pane**. You can then tab through the items shown in the main area to display detailed information about each item.

To access tab contents.

1. In the main area, move the focus to the last item in the grid view using the down arrow key. When you get to the last item, use the Tab key to navigate to the desired tab in the Tabs area.

2. Press **Enter** to open the tab that is currently in focus.

3. To navigate into the tab's contents, use the Tab key to move the focus across all the tabs to the right of the current tab. After the last one, the focus will shift to the open tab's contents.

4. When the focus moves to the tab contents, use the Tab key to move through elements in the tab. To activate a control or select an item, press the **Space** bar or press **Return**.

> **Caution:** Tabbing beyond the last item in a tab might cause you to lose focus. To be safe, back tab through the tab items (shift-tab) to the main area.

Tools and Controls

You can access tools either from the menu or by using the controls displayed in the main area or in the tabs area. Within the tools:

- Use the Tab key to move across items in the tool.

- Press **Esc** to cancel a tool at any time.

- Pressing **Enter** activates the **OK** or **Apply** button for the tool. Unless you mean to do that, pressing the **Space** bar is the means for executing an action within a tool. For example, press the **Space** bar to toggle check boxes or to execute an action specified by a link or a button,

- To select items from a drop list, use the up/down arrows.

- If the tool includes a tab area, use the right/left arrows to move across the tabs.

> **Caution:** Tabbing beyond the last item in a tool might cause you to lose focus. To be safe, back tab through the tab items (Shift-Tab) to move the focus out of the current tab.

4

Discovering Services and Dependencies

This chapter explains how you use Business Transaction Management to discover containers, services, and dependencies. Using this information you can get an exact picture of how your distributed application works, what components have performance issues, and what components require closer monitoring. It includes the following sections:

- About Discovery
- Viewing Discovered Containers
- Viewing Discovered Services
- Looking at Dependencies
- Manually Registering a Service
- Deleting Discovered Objects and Starting Over

For a complete and up-to-date list of the types of services and components that Business Transaction Management can discover and monitor, refer to the Business Transaction Management Certification Matrix. You can locate this document by searching for "BTM certification" at http://support.oracle.com.

4.1 About Discovery

Discovery is the process that allows you to identify the business components that make up a composite application and to understand how they relate to one another (their dependencies.) This section explains the concepts and processes you must understand to get exactly the information you need during the discovery process. It explains

- how Business Transaction Management models a wide variety of component types
- the steps of the discovery process
- how you can limit discovery for technologies that include a large number of intermediate endpoints
- how you can manually register services that are not automatically discoverable
- how you can model JDBC calls

For information about resolving discovery problems, see Section 12.7, "Resolving Discovery Issues."

4.1.1 What Can Be Discovered

Business Transaction Management can discover the following elements:

- Application components: this includes the logical service that designates a deployed component type, the endpoints (instances of that service), and the operations that can be invoked on an endpoint.

- Containers

- The dependencies between components

Business Transaction Management can discover a wide variety of component types and the containers in which they reside. The same model is used to represent interconnected components no matter what the component type: the model consists of services that interact by sending request and response XML messages. The model also assumes that each of the services is described by a WSDL specifying the service's location and its interface. If such a WSDL does not exist because the component is not a web service, Business Transaction Management constructs an artificial WSDL that it uses to enable the system to process the component consistently. The model is illustrated by the following figure.
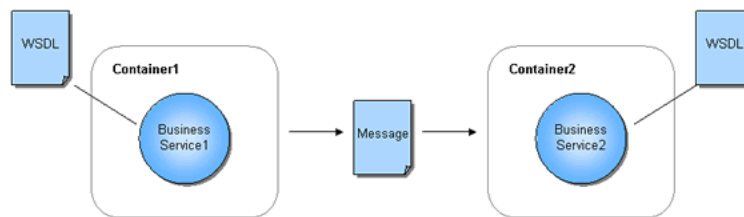
For example, if you have a composite application consisting of a web service that calls an EJB that accesses a database via JDBC, it will be modeled as three services that communicate using XML messages. When you use the Business Transaction Management console to view discovered components, these are listed as *services*, and the messages they exchange are listed as *operations* belonging to these services. A *message* corresponds to either the request or response phase of an operation.

In some cases, the observed traffic type suits this model perfectly; for example, a JAX-WS service or a JAX-RPC service. In other cases, Business Transaction Management must map the component type in a way that is compatible with its basic model. The detail of this mapping might be important if you plan to discover and monitor components that are not web services.

Business Transaction Management discovers components by observing the message traffic that flows from one component to another. Based on the data derived from observing this traffic, Business Transaction Management can also discover how these components are related to one another and draw a map of their dependencies. Dependency information provides an accurate picture of how your composite application is really behaving. It might alert you to the fact that certain components are never called, and it provides a basis for defining business transactions.

4.1.2 The Discovery Process

The most important fact to understand about discovery is that it is entirely traffic-based. If messages are not flowing through the observed endpoints, Business Transaction Management cannot discover any application components nor can it discover the dependencies between these components. When in doubt, send traffic. This section outlines the steps involved in discovering distributed application components in your observed environment. To learn more, read the sections that are linked to from the following steps.

Before discovery can happen, you must install the observers in the environment you want to observe, and you must configure the observer communication policy to define

communication between the observers and the monitor or monitor groups responsible for the further processing of the data discovered by the observers.

Discovery happens in two stages: during the priming stage, observed traffic causes the observer to start communicating with the monitor; during the observation stage, a measurement policy is applied to the data that flows from observer to monitor. This is to say that it might take a little while to build a complete picture of your working system. One symptom of this is that if you send 100 messages and Business Transaction Management reports seeing only 98, the messages that are not accounted for are the messages that served to prime the discovery process.

Discovery involves the following steps:

1. Send traffic through your system.

2. Check the containers being observed. If load balancers are being used, these must also be visible. (Restarting the containers after installing Business Transaction Management is not enough to make them visible, you must send traffic to have them be seen.)

3. View services to see that all the services you are interested in observing have been discovered.

4. Check that measurements are being taken by looking at the **Summary** or **Analysis** tab for a service. Throughput, traffic, maximum response time, and average response time should be available for all the services involved in message traffic. If measurements are inaccurate or missing, it's possible that not enough time has elapsed for Business Transaction Management to calculate traffic measurements or perhaps monitoring was disabled.

5. Look at dependencies in the Service Map to see how traffic is flowing in your system.

6. Adjust as necessary. Generally speaking if the discovered picture does not meet your expectations, the first thing to do is run more traffic to make sure Business Transaction Management has had time to see all pieces of your system. If that does not resolve your problem, you might need to do one or more of the following:

   ■ enable probes that are appropriate for the objects you are trying to discover.

   ■ manually register a service; see Section 4.5, "Manually Registering a Service."

   ■ resolve discovery problems due to versioning policy or replication problem.

4.1.3 Limiting Discovery

Depending on the technology, some messages flow directly from a client to a service; others flow through a host of intermediate endpoints before they reach their actual destination. Such intermediate endpoints might comprise the implementation of a messaging system, a job scheduling system, a distributed system, and so on. When installing probes for technologies that use intermediate endpoints, Business Transaction Management allows you to specify whether you want to monitor all endpoints or just the endpoints at the edge of such systems; often these are the endpoints that directly represent the business services of interest. By default, the monitoring of intermediary endpoints is turned off. This improves monitoring performance and eliminates data that is not essential to monitoring your distributed applications.

Figure 4–1 shows a number of observers monitoring endpoints conveying messages from a client to a service (EP5). Note endpoint EP1 and endpoint EP4 at the edge of the message flow. Note also the dotted line which indicates the relaying of context

information. If you choose to monitor all endpoints, all the endpoints shown in the figure will be discovered and monitored by Business Transaction Management.

Figure 4–1   Modelling Intermediate Endpoints
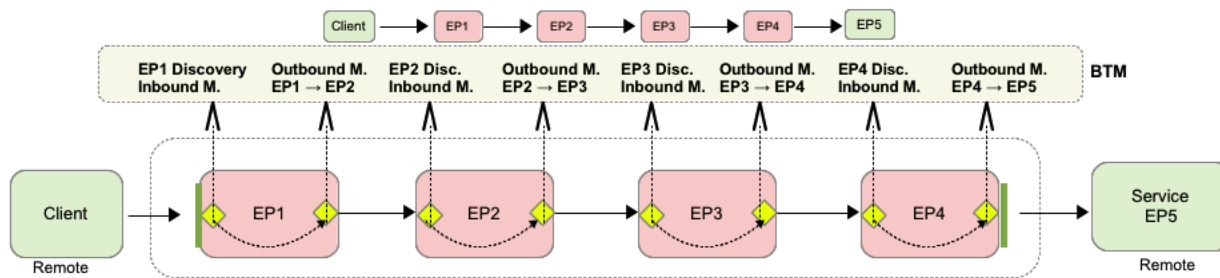
Figure 4–2 shows you how message flow is modeled if you choose to restrict the number of endpoints monitored. In this case, only the client, EP, and EP5 are discovered and monitored. Context information is still conveyed from the client to the final recipient, EP5.

Figure 4–2   Limiting Endpoint Discovery
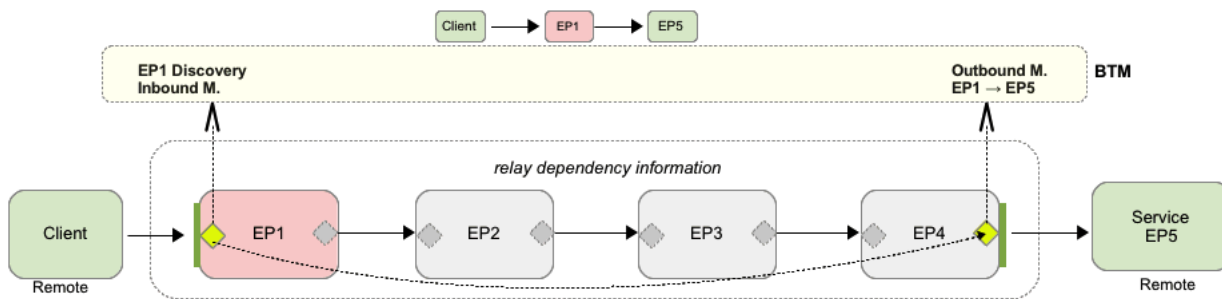
The observer communication policy gives you the option of controlling the monitoring of intermediate endpoints for SOA, OSB, and EJB probes. Options for monitoring different technologies vary slightly. For example, in monitoring EJBs, you have the following options: you can choose to model the edge of flow, which models only the first local EJB in a local request flow; you can choose to model all, which models all local EJBs; and you have the option to model none (no local EJBs). How you model local EJBs has no effect on the modeling and monitoring of remote EJBs, which are always monitored. For additional information about the observer communication policy, see Section 12.1.2, "Configuring the Observer and Monitor," especially Section 12.1.2.11, "Advanced Settings Field Reference," which describes the options you use to control the monitoring of intermediate endpoints.

4.1.4 Modeling JDBC Calls

Very large transactions or high JDBC volume might strain BTM resources when the JDBC probe is enabled and instance logging is turned on. It is also possible that calling one operation can result in a large number of calls to the database. For example, one Fusion Application operation can translate into hundreds of SQL statements. To reduce this overhead and free up resources, you can elect to have Business Transaction Management record a summary of these calls rather than noting each one individually.

If you decide to use JDBC summary mode but also want to investigate individual calls, you can use data base management tools to do so. The main idea behind the JDBC

summary option is to make BTM performance more efficient while, at the same time, allowing you to get information about slow or faulty sql statements.

Settings in the observer's communication policy determine whether JDBC summary mode is enabled. This section describes the information displayed by BTM when JDBC summary is enabled. For more information about the communication policy, see Section 12.1.2.11, "Advanced Settings Field Reference.".

When JDBC summary is enabled, JDBC calls are represented as shown in the figure below.

Figure 4–3   JDBC Call in Summary Mode
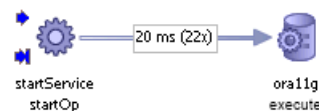
The measurements given for the JDBC call specify the average time for the call and the number of calls. In the case illustrated below, 22 calls have been made, averaging 20 ms each. In summary mode, the operation name is always `execute`. (When JDBC summary is disabled, the `executePrepStmt`, `executePrepStmtBatch`, `executeQuery`, and `executeBatch` statements are all individually discovered.)

4.1.4.1 Understanding the Display of JDBC Summary Data

With JDBC summary enabled, the transaction instance and message log display will include two additional columns showing the summary count and fault count for each calling service. (In the observer communication policy you can specify a time-out value. If SQL call times exceed this value, BTM will send an extra time-out summary message.)

The next figure shows the additional information shown when JDBC summary is enabled. Note the way response time is reported for the `execute` statement (only part of the display is shown). Any row containing summary information will be marked with a sigma symbol.

Figure 4–4   Instance Inspector: New Columns for JDBC Summary

Two additional built-in properties are defined for the request phase of a message to support JDBC summary mode: `aggregateCount` and `aggregateFaultCount`. You can use these properties in searches. For example, you could look for an `execute` message whose aggregate fault count is greater than 100.

When JDBC summary is enabled, BTM records all JDBC operations individually, but when logging message content, the information is sent in one abridged summary observation per caller, which includes payload information only for the number of slowest SQL statements and faults you specified in the observer communication policy settings. (For information on setting options that control logging of summary content, see Section 12.1.2.11, "Advanced Settings Field Reference.")

When you display message content for a summary observation, the xml listing, ExecuteSummaryContent, will look like the following. The main sections of the output are shown in bold type in this text and described following the example.

```
<ExecuteSummaryContent>
    <ExecuteCount>52</ExecuteCount>
    <FaultCount>2</FaultCount>
    <SumResponseTime>77</SumResponseTime>
    <AvgResponseTime>1.54</AvgResponseTime>
    <SummaryStartTime>1376876999364</SummaryStartTime>
    <SummaryEndTime>1376877000390</SummaryEndTime>
    <Caller>
        <Service>LogService</Service>
        <Operation>haveFaults</Operation>
    </Caller>
    <CapturedFaults>
        <Fault>
            <ArriveTime>1376877000364</ArriveTime>
            <Content> <![CDATA[select * from noneexisttable]]> </Content>
            <FaultContent> <![CDATA[ORA-00942: table or view does not exit]]>
</FaultContent>
        </Fault>
        <Fault>
            <ArriveTime>1376877000386</ArriveTime>
            <Content> <![CDATA[select * from noneexisttable]]> </Content>
            <FaultContent> <![CDATA[ORA-00942: table or view does not exist]]>
</FaultContent>
        </Fault>
    </CapturedFaults>
<TopNSlowestMessages>
        <TopNMessage>
            <Content> <![CDATA[INSERT INTO calculationlog
                    (leftoperator,rightoperator,operatortype, result,logdate)
                    VALUES(?,?,?,?,systimestamp)]]> </Content>
            <ResponseTime>4</ResponseTime>
            <ArriveTime>1376876999360</ArriveTime>
            <ExecuteCount>10</ExecuteCount>
            <AverageTime>2.30</AverageTime>
            <MinTime>2</MinTime>
        </TopNMessage>
        <TopNMessage>
            <Content> <![CDATA[DELETE FROM calculationlog where id=?]]> </Content>
            <ResponseTime>4</ResponseTime>
            <ArriveTime>1376877000150</ArriveTime>
            <ExecuteCount>10</ExecuteCount>
            <AverageTime>1.60</AverageTime>
            <MinTime>1</MinTime>
        </TopNMessage>
        <TopNMessage>
                <Content> <![CDATA[UPDATE calculationlog SET leftOperator = ?,
                 rightOperator = ?, operatorType = ?, result = ?,
                 logDate = systimestamp where id = ?]]> </Content>
                <ResponseTime>3</ResponseTime>
```

```
                <ArriveTime>1376876999552</ArriveTime>
                <ExecuteCount>10</ExecuteCount>
                <AverageTime>1.80</AverageTime>
                <MinTime>1</MinTime>
            </TopNMessage>
        </TopNSlowestMessages>
</ExecuteSummaryContent>
```

The section titled `ExecuteSummaryContent` shows the following information for summary observations:

- ExecuteCount: the total JDBC calls made, including exceptions

- FaultCount: count of all JDBC exceptions that occurred in this summary

- SumResponseTime: The sum of all SQL statements' response time

- AvgResponseTime: The average of response times across the summed JDBC call

- SummaryStartTime: See Note below.

- SummaryEndTime: See Note below.

- The calling service, and operation

In the communication policy, you can also specify the number of exceptions you want captured. For each such exception or fault, the following information is shown in the section titled `CapturedFaults`:

- ArriveTime: the time when the exception arrived. See Note below.

- Content: The SQL statement where the exception occurred

- FaultContent: The full text of the SQL exception content

In addition, the section `TopNSlowestMessages` shows the following information for each of the N slowest SQL statements. (You specify a value for N using the communication policy.)

- Content: SQL statement name (but not the parameters passed to these)

- ResponseTime: the execution time in microseconds for this SQL statement

- ArriveTime: Start time. See Note below.

- ExecuteCount: how many times this SQL statement was called

- AverageTime: the average time for all SQL statements recorded

- MinTime: the minimum time for all SQL statements recorded

> **Note:** Times for `ArriveTime`, `SummaryStartTime`, and `SummaryEndTime` are given using a UNIX timestamp (based on seconds since 1/1/1970). To convert this value to a readable date and time, look for a conversion tool on the Internet by searching for "UNIX time conversion."

4.1.4.2 Updating Transaction Definitions After Changing Summary Options

If you have already created a transaction and you decide to change the setting of JDBC summary, you will need to run more traffic and then update your transaction definition. This is because, depending on your JDBC summary setting, BTM will discover the JDBC service operations for SQL execution under different names.

- When JDBC summary is disabled, the `executePrepStmt`, `executePrepStmtBatch`, `executeQuery`, and `executeBatch` statements will all be individually discovered

- When JDBC summary is enabled, the `executePrepStmt`, `executePrepStmtBatch`, `executeQuery`, and `executeBatch` statements will all be combined into a single `execute` operation

If you change the JDBC summary setting after creating a transaction definition, please do the following:

1. Run traffic through the JDBC service again to discover the new operations.

2. Edit the transactions that use JDBC services by removing the old operations and including the newly discovered ones.

If you don't update your transaction definition, BTM will be unable to recognize the newly discovered JDBC service operations as being part of the transaction, and it will not be able to record measurements for traffic to that service.

4.1.5 Registering a Service Manually

There are cases where Business Transaction Management cannot discover SOA-type components directly: for example, the service resides in a container that cannot be observed or the service resides in a container where no observer has been installed. In such cases, it might still be able to discover the object if you manually register the service.

While it is possible to discover services in this way, it is not usually possible to monitor their performance without the services being directly observed. For more information, see Section 4.5, "Manually Registering a Service."

4.2 Viewing Discovered Containers

Discovery displays information about containers hosting Business Transaction Management system services and hosting observed components. This section describes the information available about a container. It also explains how you edit container profile information and how you unregister a container.

4.2.1 Viewing Container Information

You can view container information from the **Explorer > Container** view or from the **Maps> Container Map** view.

To view a container map:

1. Select **Maps >Container Map** in the Navigator.

2. Float-over help specifies the host name for each container. Pass your cursor over the containers to display the help.

3. Opening the **Tabs** area and clicking on a container icon displays the **Profile** and **Services** tabs for the container. See the next section for a description of the tab contents.

4.2.2 Viewing Summary and Detail Information

To view summary information about discovered containers and their contents:

1. Select the **Explorer > Containers** view in the Navigator. Information about discovered containers is shown in the main area. By default Business Transaction

Management shows information about all known containers. Use the **Filter** control to filter the displayed information about containers.

2. Click the plus icon by the container name to display the contents of the container. This allows you to view the application components hosted in the selected container.

3. Click the plus icon by the component name to view the operations that make up that component. The operations are listed in a tree view under the container.

To view detail information about a discovered container:

1. Select the container in the main area.

2. Double click the container name. Two tabs are shown in the details pane: **Profile** and **Services**.

3. The **Profile** tab displays information about the container and the deployed observer; the table below describes the fields of the **Profile** tab. User-defined fields can be filled in by editing profile information.

4. The **Services** tab displays information about the services running in this container.

Field	Description
Notes	User-defined field. Specify any information that helps you understand the use or contents of the container
Base address	The http-based address (entry point) for this container
Aliases	All the IP addresses by which the container might be known to the Sphere. This might include VPN adapter address, user-defined aliases, or other addresses Business Transaction Management discovers by observing message traffic to the container. This information is useful in investigating improper setup of the container.
Container Type	The container type and version number.
OS	The operating system and version
Host	The host name where the server is running.
Identifier	Unique identifier for this object in Business Transaction Management.
Administration UI Console	User-defined field: URL of administrative console for the container. You can use this link to launch the console Although Business Transaction Management supplies a value for this address, it might not be accurate and should certainly be changed if the console is moved. You can change this value by editing the container profile.
Monitoring Details	Information about the observer deployed in the specified container and dates for last discovery, registration, synchronization with the Sphere, and version.
Contact information	User-defined field to provide contact information for the server administrator or other support personnel.
Routing details	Information about any load balancers observed routing messages to the container

4.2.3 Editing Container Profile Information

To edit container profile information.

1. Select the container in the container map area or in the main area.

    2. Select the **Modify > Edit Profile for <container>** item.

    3. Change any of the editable fields (shown in yellow).

    4. Click **Apply**.

4.2.4 Unregistering a Container

Unregistering a container removes the container and all discovered serves deployed in the container from the sphere registry. You might need to do this after completing one of the following tasks:

- Moved a container to another sphere.

- Uninstalled the Business Transaction Management observer from a container.

- Uninstalled or physically removed the container from your system.

You should unregister the container only after you have completed one of these tasks. Otherwise, an unregistered container will register itself upon startup if nothing has changed.

    1. Select the container in the **Maps > Container Map** view or in the **Explorer > Container** view.

    2. Select **Delete***ContainerName***Registration** from the **Modify** menu.

    3. Click **Delete**.

4.3 Viewing Discovered Services

Business Transaction Management displays information about discovered services, service endpoints, routers that distribute message traffic to endpoints, and service operations.

- *services* refer to logical services

- *endpoints* refer to service instances and routers

- *operations* are invoked by request and response messages that one service sends to another

When you start Business Transaction Management, no services are shown. For services to be discovered:

- Services must be deployed to application servers where observers have been installed and the appropriate probes have been activated.

- Traffic must flow from one service to another service (unless you manually register a service).

Once services are discovered, you can access information about them from the **Services To Endpoints** and **Services To Operations** Navigator views. You can view services dependencies in the **Maps > Service Map** view.

4.3.1 Service Types

Business Transaction Management assigns a type to a service based on the type of the first endpoint discovered for the service. Service types include Web Application, Web Service, and Database. (A web application is a component that interacts with the user via HTML pages (screens)). Note that a service might have multiple endpoints of different types and be discovered by different observers. For example a web service

might have multiple endpoints on different containers implemented using different technologies: JAX-RPC, JAX-WS, WCF, and so on.

4.3.2 Deployment Topologies and Service Information

The service information displayed varies with the topology of your deployment. If you have only one instance of a service running, one endpoint for that service is shown. If you have several instances of a service running, several endpoints are shown for that service.

There are a variety of reasons why you might discover and have to monitor several instances of a service:

- The service is replicated for performance or failover. In this case all the replicated instances are shown as the service's endpoints. The router distributing traffic to those instances is also shown; the router name is the endpoint name with the suffix `Router`.

- The service is deployed in a container with both secured (HTTPS) and unsecured (HTTP) ports. One endpoint will be shown for each port only if traffic flows to that port.

- A WSDL includes different bindings for a service, each binding defining a different set of operations that can be implemented by different endpoints. This case is rare, but Business Transaction Management will recognize it and discover the endpoints.

4.3.3 How the Existence of Routers is Inferred

It might take a while to determine that a router is being used to re-direct traffic. Business Transaction Management uses the Host headers in HTTP traffic to detect when messages were originally sent to a different address than the container where they were observed, and connects the caller to the recipient in the dependency graph.

 When the Host header contains a different host name but the same port, Business Transaction Management will initially add aliases for the container and its endpoints. When the Host header contains a different port than the container is actually listening on, or the same Host header is observed in traffic sent to two or more containers, Business Transaction Management infers the existence of a hardware load balancer between the caller and the service, and will add router endpoints to the sphere model as needed to connect the caller and its target endpoints in the dependency graph.

4.3.4 Viewing Services

You can view service using either the **Services To Endpoints** view or the **Services to Operations** view from the Navigator.

To view **Services To Endpoints,** select the view from the Navigator. Business Transaction Management displays information in the main area of the console. The following table describes the contents of this view.

The **Services To Endpoints** view is useful in that it shows endpoint replicates. Replicates are distinguished by their address, shown in the main area. They also have distinct definitional and performance information, as shown in the **Tab** section of this view.

Column name	Description
Name	The Name of the service. Expand the top logical service name to show the endpoints it contains. Expand an endpoint to view its operations.
Up/Down Arrow	Green arrow specifies that the service, endpoint, or operation is running; Red arrow specifies that it's down. Yellow indicates that a service contains endpoints; some of which are running and some of which are down.
Address	Address of the service's container. Replicate endpoints will have different addresses.
Type	The type of the service. This is based on the type of the first service endpoint discovered.
Container type	The type and version of the service container.

To view detailed information about each service, endpoint, or operation double click the desired element to display detailed information in the Tab area, which includes the following data:

Tab	Description
Summary	Shows a summary of performance measurements for the object selected in the main pane.
Analysis	Shows performance information across a specified period of time.
Alerts	Shows alerts for the selected object.
Message Log	Shows available messages if message logging is enabled for a given operation.
SLA Compliance	Shows the compliance of the selected object with service level agreements defined for the given object.
Profile	Shows definition of selected object. You can edit some of this information by selecting **Modify > Edit Profile for** *object name*.
Dependency	Shows the dependencies between the selected object and related objects.
Policies	Lists the policies applied to the currently selected object.
Downtimes	Specifies the scheduled downtimes for the selected object.
Properties	Specifies the properties defined for the selected object. Use the **Edit** button to modify, duplicate, or delete these properties.

To view **Services-To-Operations**, select it from the Navigator. Business Transaction Management. The type of information displayed is the same as for the **Services-to-Endpoints** view. Tab information is also the same.

4.4 Looking at Dependencies

In addition to discovering business components, Business Transaction Management can discover how components relate or *depend* on one another by observing message traffic as it flows from one component to the next. Dependency information provides an accurate picture of how your composite application is really behaving. It might alert you to the fact that certain components are not being called, and it provides a basis for defining *business transactions*.

You can view dependencies between services, endpoints, (service instances), or operations. Each dependency map provides information appropriate to the element selected.

You can also delete dependencies if dependency graphs show out-of-date dependencies.

4.4.1 Showing Related Elements

No matter what type of dependency it is showing, a dependency diagram is rooted in the selected object and displays all the links that lead out from and into that starting point. Anything that is not directly upstream or downstream of the root object is not shown. To display these additional elements, click on the wrench icon to display the **Layout** and **Show-Related** controls.

The default value for **Show Related** is **None**, meaning that only elements directly linked to the root object are shown.

1. Select **One** to show the next level of objects indirectly linked to the root object, or select **All** to show all objects linked to the root object.

2. Click **OK**.

4.4.2 Service Dependencies

To view service dependencies

1. Select the **Services To Endpoints** view or the **Services to Operations view** in the Navigator.

2. Double click on the service of interest to open the tab area if it is not already open. If it is open, just select the service of interest.

3. Select the **Dependency** tab. Business Transaction Management displays the dependencies between the selected service and the other services with which it interacts. Arrows display traffic flow; their thickness indicates relative throughput size.

4. You can move the cursor over the services displayed to have Business Transaction Management display their component type.

You can also display service dependencies by choosing the **Maps > Service Map** view from the Navigator.

4.4.3 Endpoint Dependencies

To view endpoint dependencies:

1. Select the **Services To Endpoints** view in the Navigator.

2. Expand the service of interest to display its corresponding endpoints. A (logical) service might have several corresponding endpoints if the service has been replicated or if different endpoints are used for secure/unsecure communication.

3. Double click on the endpoint of interest to open the Tab area. If it is already open, just select the endpoint of interest.

4. Select the **Dependency** tab. Business Transaction Management displays the dependencies between the selected endpoint and the other endpoints with which it interacts. Arrows display the direction of traffic flow and relative throughput. For each endpoint, the host name and port for the container where the endpoint

resides are also displayed. The means by which the endpoint is discovered is shown in parenthesis; the following table describes the possible types.

Means of discovery	Meaning
observer	The endpoint was discovered by an observer
router	The endpoint was discovered by a router. (The existence of a hardware router was inferred based on discrepancies between the HTTP Host header and the physical address of the observed container(s).)
registered	The endpoint was manually registered
DTA	The existence of the endpoint was inferred based on outbound traffic addressed to it from an observed endpoint.

As you pass the cursor over individual endpoints, Business Transaction Management displays the endpoint name, the container's host and port, and the service and component type of which the endpoint is an instance. Core measurements are also given.

4.4.4 Operation Dependencies

To view dependencies between operations:

1. Select the **Services to Operations** view in the Navigator. You can view dependencies between logical or physical operations. Logical operations are listed immediately under the service; physical operations are listed under logical operations. The name of a physical operation takes the form

   endpointName on *containerName*

   For example: `checkCredit.CreditServiceSOAP on uitest20:7011`

2. Double click the operation of interest to open the tab area. If it is already open, select the desired operation.

3. Click the **Dependency** tab if it is not already selected.

4. The call chain containing the operation is shown.

5. Pass the cursor over the operation to display additional float-over help.

4.4.5 Deleting Dependencies

You might need to delete all dependencies if your application has changed and the dependency graph does not reflect these changes. Dependency data does not age out, so in those cases where the graph includes obsolete data, you might want to clear all dependencies and regenerate the graph by running more traffic.

To delete dependencies:

1. Select **Delete Dependencies** from the **Admin** menu

2. Click **Delete**.

3. To regenerate dependencies, run more traffic.

4.5 Manually Registering a Service

Normally, if a service is not under direct observation, it cannot be discovered or monitored. But you can manually register a service, display it, and obtain measurements for calls going out to it.

You should register a service if you will have outgoing calls to the service but cannot install an observer on the service's container. Once you register the service, the system will be able to record measurements and log messages that are observed on the client side of the message exchange.

After registering the service, you might want to use the `registerExternalContainer` command to group its endpoints into an external container. Otherwise, by default, manually registered endpoints are allocated to the System container, and are displayed under the **Unassigned Endpoints** node.

To register a service:

1. Choose **Admin > Register > Service WSDL** from the main menu.

2. Specify the URL of the service's WSDL and click **Next**.

3. Specify the name by which the service will be registered with the sphere. By default, the name is extracted from the WSDL definition; you can use an alternate name if you like.

4. Optionally, specify a version for the service.

5. Click **OK**.

The service should now be listed in the summary area of the **Services to Endpoints** or **Services to Operations** view.

4.6 Deleting Discovered Objects and Starting Over

Creating a useful discovery configuration can be an iterative process, particularly in the early stages of using Business Transaction Management. You might find that default settings for enabling probes are turning up too much information, or that changing your deployment or the observer-monitor topology results in redundant or erroneous information. To spare you the need to reinstall the system or to manually remove all observed entities and related artifacts, Business Transaction Management provides the deleteAll command. This command deletes objects already discovered along with related artifacts such as transactions, properties, registered services, devices, and containers.

Use this command judiciously to avoid unwanted loss of data, which includes historical data related to observed objects. The command is most appropriate when you start working with Business Transaction Management and are fine tuning your discovery scheme. It should never be used in a production environment.

See Section 10.11, "deleteAll" for more information.

5

Working with Transactions

A *transaction* is a sequence of operations that you want to monitor as a single unit. This chapter discusses the advantages of using transactions, how you define and monitor transactions, how you access JVM diagnostics and Request Instance diagnostics, and how you troubleshoot transactions. It includes the following sections:

- About Transactions
- About Properties
- Working with Properties
- Defining, Modifying, and Deleting Transactions
- Viewing and Monitoring Transaction Performance
- Top 10 Transactions Dashboards
- Viewing Transaction Instances
- Viewing Logged Messages and Properties
- Accessing Other Diagnostic Tools
- Getting Information About a Transaction
- Troubleshooting Transactions

5.1 About Transactions

You define transactions on the logical level by specifying which operations participate in the transaction. Business Transaction Management monitors transactions by looking at individual endpoints, and it displays the level of information you require. The level of detail it provides ranges from the minimum of aggregate performance information to the maximum of message content logging. Using Business Transaction Management to monitor transactions is an iterative process: initially you want to look at the greatest possible number of endpoints to get an overall feel for service usage and for potential trouble spots. As you identify these, you want to look at a narrower set of elements but in greater detail. By using this approach, you can get exactly the information you need without straining system resources.

You might want to define and monitor transactions in order to do the following:

- Assess the performance of business transactions. Business Transaction Management provides information about which transactions are completing, which are failing, and which get stuck. It also tells you how long it takes a transaction to complete as a whole and what part of that duration is used by each operation in the transaction.

- Find transaction instances based on message content.

- Troubleshoot problems with transactions: analyze bottlenecks and diagnose the reason for bad performance or failed transactions.

Users playing different roles might use information about transaction performance as follows:

- Operations personnel need to determine which transactions are impacted by server or component downtime. This helps them prioritize their work and schedule maintenance.

- Developers need to understand which transactions are impacted by implementation changes to components.

- A business application owner needs to monitor charts that depict the performance and throughput of transactions.

You define transactions by using the discovery process to identify operation flows. Based on the dependencies revealed by discovery, you can define a transaction to include services whose interactions interest you. You can then modify the default transaction definition to enable monitoring features that provide exactly the level of detail needed about performance and usage. Features include segmenting information by user or host address, instance and property logging, and message content logging.

This section describes this iterative process of defining transactions and the level of detail that is made available as you enable each feature. It includes the following sections:

- What Defines a Transaction?

- Properties

- Auto and Manual Correlation

- Default Transaction Definition

- Additional Features

- Service Level Agreements and Conditions

5.1.1 What Defines a Transaction?

A transaction is defined by the following:

- A primary operation flow. This is specified by providing the *start message*, which triggers the flow. The start message is typically the request message for the initial operation of the flow.

- Additional operation flows if needed, which must be manually correlated to the primary flow.

- The ending message of the transaction. By default this is the response message from the initial operation of the primary operation flow. However, you can change it to model more complex transactions.

You can alter the transaction's view of the operation flows by deleting one or more operations from a flow.

Business Transaction Management allows you to include the same operation in multiple transactions; the only limitation is that the same operation cannot be the starting operation of more than one transaction. If you do include an operation in multiple transactions, Business Transaction Management is able to keep accurate aggregate measurements for each transaction's overall response time, counting the

invocation of the shared operation appropriately as it is called by each of its owning transactions. Business Transaction Management is also able to handle situations where services are replicated, accurately collecting performance information as one replicate or another is used in a failover or load balancing architecture.

A transaction executes many times in a given period; Business Transaction Management tracks the flow of messages included in the transaction and can map these to particular *transaction instance*s.

- A transaction instance starts when an instance of the primary operation flow is started.

- A transaction instance ends when the system sees an instance of the ending message that correlates to the instance's starting message.

In addition to the operation flows it contains, a transaction is also defined by settings that specify the following information:

- The scope of the transaction: how long you expect a single transaction to run. Business Transaction Management uses this value to figure out when to stop looking for messages for a given transaction instance

- How long to retain logged messages that you might want to assemble so that you can view a particular instance of a transaction

- Whether transaction monitoring is enabled and whether free text search is enabled

- Whether segmentation, instance logging, and message content logging are enabled

5.1.2 Properties

In order to allow the user to extract as much information from a message without logging the content of every message, Business Transaction Management offers the use of properties, which are associated with transaction operations. Properties can be associated with parts of the message header or message payload. You can use them in the following ways:

- To provide values for message keys used to correlate operations in a transaction

- To surface the value of a message element without having to log the content of the message

- To search for specific transaction instances

- To define conditions

5.1.3 Auto and Manual Correlation

Once you define the start and end operation of a transaction, by default Business Transaction Management determines the intervening operations based on traffic flow, and it creates message fingerprints to link together (correlate) the operations of a transaction instance. You can also define manual keys as an alternate means of correlating transaction messages.

You will need to manually correlate operations in the following cases:

- To add a secondary operation flow to the transaction. This might be necessary in the case of asynchronous systems like JMS, or desirable if you want to connect two operation flows that share underlying data but are not causally connected. For example, you have one flow defined by an order entry, and another flow defined by the fulfillment of the order. The same order is involved in two otherwise disjoint processes.

- To define a transaction whose end message is not the end message of the operation that starts the flow.

- To define a missing message condition

See "Correlating Messages in a Transaction Using Manual Keys" for additional information on what you need to correlate in each of these cases.

A single transaction definition can contain operations that are correlated automatically and other operations that are correlated manually.

5.1.4 Default Transaction Definition

When you select an operation and choose to create a transaction, Business Transaction Management displays a **Create Transaction Definition** tool with default values already filled in. According to this default definition:

- The start message of the transaction is the request message of the operation you selected; the ending message is the response message of that operation.

- Transaction monitoring is enabled. That is, Business Transaction Management will collect data that enables it to calculate and display the following measurements: the number of started and completed transactions, throughput, average response time, maximum response time, and faults.

- Messages in the transaction are correlated using message fingerprints (calculated by Business Transaction Management).

- Segmentation, instance and property logging, and message content logging are disabled.

The default configuration is sufficient to give you a first broad look at your transaction. It can help you identify slow links, faulty services, and unexpected traffic spikes. It can also show you if overall transaction performance is degrading or not completing on time. You can also create fixed value and baselines service level agreements (SLA's) based on the measurements collected for the transaction.

You can customize the default transaction definition in the following ways:

- Change the operations that are included in a transaction and change the end operation. You might want to do this because the transaction contains one way operations, because you want to limit the transaction to fewer operations than the overall request flow, or because you want to add services in those cases where services are rarely invoked and therefore cannot be found within the discovery period.

- Correlate messages using message keys.

- Turn on features that will give you more detailed information about the transaction.

The following subsections describe the features you can add to your transaction definition.

5.1.5 Additional Features

The following are the features you can add to provide a more detailed picture of transaction performance and message content.

5.1.5.1 Segmentation

You can segment transaction measurements based on host address and by individual consumer. Consumer segmentation might be useful if you have usage contracts with specific customers or if you want to identify customers that make especially heavy or light use of your services.

Segmentation by host address can help you understand the distribution of requests in your network. You can discover whether some network segments or hosts are bearing a disproportionate load of the traffic, or that problematic transactions are all executing on a particular host.

5.1.5.2 Instance and Property Logging

Enabling instance logging allows you to see a list of transaction instances captured in a given time period. You can assemble and inspect a given instance, view any property values for that instance, and create conditions based on these property values.

You can also search on the logged properties and find the related transaction instance.

5.1.5.3 Message Content Logging

Enabling message content logging allows you to view message content for the operations you specify, and you can search for an operation based on the content of its request or response message.

5.1.6 Service Level Agreements and Conditions

The features you enable for a transaction record information for the entire transaction and for select operations. In addition to this continual monitoring, whose results you must analyze to discover performance issues, you can also configure Business Transaction Management to alert you about special situations by using conditions and service level agreements.

- Service level agreements define standards of performance for your transactions based on aggregate measurements. Business Transaction Management then monitors deviations from those standards, and when deviations occur, an alert is issued and displayed in the Management Console.

- Conditions can alert you when an expected message does not arrive, when a specified message property value is encountered, or when a fault occurs. They are tools to help you detect issues in specific transaction instances. When the condition is triggered and satisfied, Business Transaction Management assembles the corresponding transaction instance, allowing you to view its content and perform whatever analysis is needed for troubleshooting or other performance evaluation.

5.2 About Properties

Properties are variables that hold values associated with the request or response phase of an operation. A property value can correspond to a message element, a fragment of an element, the combined value of several elements from the same message, or an expression that uses a message element. You can use a message element from either the message header or body to define a property.

Properties are commonly used to facilitate searches, to surface message elements without having to log message content, to define conditions, and so on.

This section explains the different ways you can define and use properties. It also describes standard properties that are defined by BTM for any operation.

Property definitions are shown in the **Properties** tab for any transaction that includes the service operation for which a property has been defined. Property definitions are also shown in the **Properties** tab for the service operation, endpoint operation, service, and endpoint. However, you can modify or delete a property only from the **Properties** tab for a transaction.

5.2.1 Using Properties

Using properties helps you do the following:

- Surface aspects of a message that you would otherwise need to drill down to see. The use of properties can improve performance by making it possible for you to view values of message elements without having to log message content.

- Define keys that allow you to manually correlate the operations that make up a transaction

- Segment transactions by consumer

- Define a condition that depends on the value of a message property

For example, imagine you have a service element containing a customer ID. If you wanted to know which customers experience exceptions on a transaction that includes the operation, you could create a property to hold the value of the customer ID element. Then, when you inspect performance results, the ID of customers experiencing exceptions on that transaction would be prominently displayed.

Standard properties, described in the next section, are defined by Business Transaction Management; *user-defined properties*, as the name suggests, are defined by the user as described in Section 5.2.3, "User-Defined Properties."

5.2.2 Standard Properties

The properties described in the following table are defined by BTM for any message. Standard properties are defined for each phase of a message: request, response, and fault. You can use the `aggregateCount`, `aggregateFaultCount` and the ECID-based properties in searches. You can use the ECID-based properties for manual keys.

Table 5–1   Standard Properties

Message Phase	Property Name	Description
Request	aggregateCount	The number of calls associated with this `execute` statement. For more information, see Section 4.1.4.1, "Understanding the Display of JDBC Summary Data."
	aggregateFaultCount	The number of faults associated with the given `execute` statement. For more information, see Section 4.1.4.1, "Understanding the Display of JDBC Summary Data."
	arrivalTime	The arrival time of the request message measured as milliseconds elapsed since January 1, 1970, 00:00:00 UT.
	count	The number of times the requested operation was invoked in the current transaction instance. Each request increments the count even if the operation does not execute successfully.

Table 5–1 (Cont.) Standard Properties

Message Phase	Property Name	Description
	`exists`	A boolean expression that is true if the request message has arrived as part of the current transaction instance.
	`interactionId`	For internal use only.
	`requestECID`	The ECID associated with the request phase of the selected operation. For more information, see Section 5.9.2.1, "ECID Basics."
	`sender`	Deprecated. Do not use. This property will be removed from future releases.
		A string identifying the sender of the request message. This is typically the username portion of the credentials that the sending application provides to the application server in which the receiving service is deployed. This property is set only if the application server authenticates users.
Response	`arrivalTime`	The arrival time of the response message measured as milliseconds elapsed since January 1, 1970, 00:00:00 UT.
	`exists`	A boolean expression that is true if the response message has arrived as part of the current transaction instance.
	`interactionId`	For internal use only.
	`responseECID`	The ECID associated with the response phase of the selected operation. For more information, see Section 5.9.2.1, "ECID Basics."
	`responseTime`	The time interval between the arrival of the request message and the arrival of its corresponding response message measured in milliseconds.
	`sender`	Deprecated. Do not use. This property will be removed from future releases.
Fault	`actor`	The value of the fault message's `faultactor` element, which provides information about who caused the fault to happen within the message path.
	`arrivalTime`	The arrival of the fault message measured as milliseconds elapsed since January 1, 1970, 00:00:00 UT.
	`code`	The value of the fault message's `faultcode` element, which provides an algorithmic mechanism for identifying the fault.
	`detail`	The falue of the fault's message's detailcode element, which provides application-specific error information related to the `Body` element.
	`exists`	A Boolean expression that is true ifthe fault message has arrived as part of the current transaction instance.
	responseTime	The time interval between the arrival of the request message and the arrival of its corresponding fault message measured in milliseconds.

Table 5–1   (Cont.) Standard Properties

Message Phase	Property Name	Description
	sender	Deprecated. Do not use. This property will be removed from future releases.
	string	The value of the fault message's faultstring element, which provides a human-readable explanation of the fault and is not intended for algorhithmic processing.

5.2.3 User-Defined Properties

A property can be based on any XPath expression defined on a request or response message, including message content or message headers (for example, SOAP headers, HTTP headers, or JMS message properties).

The simplest type of property holds the value of a single message element. For example, you might have the following message element:

```
OrcerServiceImpl.submit.request.order.orderId
```

You could create a property named Order_ID whose value is the value of this element.

You can also create a property that stores the result of a computation involving one or more message elements. For example, you might multiply an element that holds a price times a constant to compute tax:

```
OrderServiceImpl.submit.request.order.grandTotal *.0825
```

You could store the result in a property named Tax_Due.

In some cases, you might not be able to extract the data you need from messages directly. In these cases, you can create a property based on an XPath expression to extract data from your message and perform computations.

For example, if your message contains an indeterminate number of <item> elements, you could create a property to count the number of <item> elements in the message and store that value.

You might also use an XPath expression to extract data from an XML element that itself contains an encoded XML document.

Finally, you might have a service that is missing a WSDL. Using Business Transaction Management you can log a message from this service and then use the XML view to obtain the element you need to define the desired property.

The following subsections provide additional information about creating, modifying, duplicating, and deleting properties.

5.3 Working with Properties

Properties are variables that hold values associated with the request or response phase of an operation (message). This section explains how you use Business Transaction Management to work with properties. It includes the following sections:

- Defining Properties
- Properties for Services that Have Missing or Incomplete WSDLs
- Viewing Property Definitions
- Viewing Property Values

- Modifying Properties

- Duplicating Property Definitions

- Removing Properties

5.3.1 Defining Properties

The process you use to define a message property varies depending on whether the property corresponds to a single message element, a fragment of an element, a combination of elements, or an expression that includes an element.

You can access the **Create Message Property** tool by selecting a service, operation, or endpoint in any view; then select **Create Message Propertyon** *Name* from the **Create** menu.

Independently of the kind of property you create, you will need to specify the following information (in addition to specifying the message element that supplies the property value):

- A **Name** for the property

- The **Operation** (message) whose content you will use to define the property value

- An optional **Description** explaining the purpose of the property or anything other notes you want to retain about this property.

- The **Property Source**: where the value of the property is to be obtained.

- Whether the property should be mapped to a consumer through a particular attribute. If you want the property to be associated with a particular consumer attribute, check the **Consumer Mapping** check box. By default, the only user attribute defined by the system is **Consumer Name**. You can define other consumer attributes if you need to.

- Whether the value of the property can be accessed only by Business Transaction Management users whose role allows them to view sensitive information. To restrict access, click the **Treat as sensitive** check box.

If you define a new property after messages have been collected, the logged messages will not include the newly defined properties.

The following procedures describe how you can create properties based on a single element, based on an element fragment, or based on more than one message element.

Defining a property based on a single element

1. Select a service, operation, or endpoint in any view.

2. Select **Create Message Property on** *Name* from the **Create** menu.

3. Specify the name of the property. If you do not provide a name, uses the message element name for the property name.

4. **Property Source**: Specify whether the element of interest is part of the message content or the message header.

   If you select **Header**, Business Transaction Management displays a small dialog you must use to specify the Header Name, the phase (request, response, fault), and the data type. Then skip to **Step 8**. Steps 5-8 apply if you selected **Message Content.**

5. Click the **Pick from Message** link to display the message elements.

6. Use the **Message** drop down list to select **Request** or **Response**.

7. Double click the message element to select it.

8. Click the **Consumer Mapping** check box if you want to map this property to a consumer attribute.

9. Select the attribute if there is more than one.

10. Click the **Sensitivity** check box if you want to restrict access to the property value.

11. Click **OK**.

Defining a property based on an arithmetic expression

1. Select a service, operation, or endpoint in any view.

2. Select **Create Message Propertyon** *Name* from the **Create** menu.

3. Specify the name of the property. If you do not provide a name, Business Transaction Management uses the element name for the property name.

4. **Property Source:** Select **Message Content.**

5. Click the **Pick from Message** link to display the message elements.

6. Use the **Message** drop down list to select **Request** or **Response**.

7. Double click the message element to select it.

8. Click the **Convert To XPath** item. The display shows the XPath definition of the element.

9. In the XPath box, create the desired expression. For example, if you want to define a Tax property that computes tax, you could take the original XPath expression for the `Price` element

    ```
    /soapenv:Envelope/soapenv:Body/*[local-name(.)='submit']/^*
    [1]/^tns1:OrderItems/^tns1:Price
    ```

    and define it to compute a tax of 10%; for example:

    ```
    /soapenv:Envelope/soapenv:Body/*[local-name(.)='submit']/^*
    [1]/^tns1:OrderItems/^tns1:Price * .10
    ```

10. Click the **Consumer Mapping** check box if you want to map this property to a consumer attribute.

11. Select the attribute if there is more than one.

12. Click the **Sensitivity** check box if you want to restrict access to the property value.

13. Click **OK**.

Defining a property based on the partial contents of an element

This might be useful if you have nested xml documents and want the property's value to contain a nested element.

1. Select a service, operation, or endpoint in any view.

2. Select **Create Message Propertyon** *Name* from the **Create** menu.

3. Specify the name of the property. If you do not provide a name, Business Transaction Management uses the element name for the property name.

4. **Property Source**: Select **Message Content**.

5. Click the **Pick from Message** link to display the message elements.

6. Use the **Message** drop down list to select **Request** or **Response**.

7. Double click the message element to select it.

8. Click the **Convert To XPath** item. The display shows the XPath definition of the element.

9. Edit the XPath expression to limit the property value to the nested element.

10. Click the **Consumer Mapping** check box if you want to map this property to a consumer attribute.

11. Select the attribute if there is more than one.

12. Click the **Sensitivity** check box if you want to restrict access to the property value.

13. Click **OK**.

Defining a property based on the combined contents of multiple elements of the same message

This might be useful if you need to provide manual keys to correlate messages but no single message element is unique to a service. You can create a property for the unique value that results from combining two message elements; for example, Customer ID and zip code.

The following procedure assumes you are combining two elements. You can extend this case for additional elements.

1. Select a service, operation, or endpoint in any view.

2. Select **Create Message Propertyon** *Name* from the **Create** menu.

3. **Property Source**: Select **Message Content**.

4. Click the **Pick from Message** link to display the message elements.

5. Use the **Message** drop down list to select **Request** or **Response**.

6. Double click the message element to select it.

7. Click the **Convert To XPath** item. The display shows the XPath definition of the element.

8. Copy the XPath definition and push the **Cancel** button to dismiss the **Create Message Property** tool.

9. Select the same operation again.

10. From the **Create** menu, select **Message Property on** *Name.*

11. Specify the name of the property.

12. **Property Source**: Select **Message Content**.

13. Click the **Pick from Message** link to display the message elements.

14. Use the **Message** drop down list to select **Request** or **Response** (the phase has to be the same as for the message from which you copied the first element).

15. Double click the message element to select it.

16. Click the **Convert To XPath** item. The display shows the XPath definition of the element.

17. Paste the XPath definition you have copied from the first element. Depending on the element type, use the appropriate operator to form a valid XPath expression.

18. Click the **Consumer Mapping** checkbox if you want to map this property to a consumer attribute.

19. Select the attribute if there is more than one.

20. Click the **Sensitivity** checkbox if you want to restrict access to the property value.

21. Click **OK**.

5.3.2 Properties for Services that Have Missing or Incomplete WSDLs

Business Transaction Management uses the same model to represent interconnected components no matter what the component type: the model shows services that interact by sending request and response XML messages. The model also assumes that each of the services is described by a WSDL specifying the service's location and its interface. However, not all discovered components can be precisely rendered by this model. If the component is not an actual web service, the corresponding WSDL might be missing or incomplete. In such a case, you will need to take some additional action to create properties for the message elements of this component. Basically, you will log a message containing the element of interest, and then copy the XML into the property definition.

The process is the following:

1. Enable message logging for the transaction operation that contains the element for which you want to create a property.

2. Run traffic through your system to capture at least one transaction instance.

3. Select the transaction of interest in the main area of the console.

4. Open the **Message Log tab** and select the operation for which you have logged messages.

5. Click on the magnifying glass icon to open the **Message Content** inspector.

6. Select **View XML** for either the Request or the Response message, depending which of these contains the element of interest.

7. Copy the XML for the message.

8. Select a service, operation, or endpoint in any view. (The service must contain the operation of interest.)

9. Select **Create Message Property on** *Name* from the **Create** menu.

10. **Property Source**: Select **Message Content**.

11. Click the **Pick from Message** link to display the message elements. Ignore the element tree that is already filled in (if any).

12. Select the operation phase: **Request** or **Response**.

13. Click the **Get XML** button to open the **Paste XML document** window.

14. Paste the text in the window and click OK.

15. The element tree should now contain the message element.

16. Select the element of interest.

17. Click the **Consumer Mapping** check box if you want to map this property to a consumer attribute.

18. Select the attribute if there is more than one.

19. Click the **Sensitivity** check box if you want to restrict access to the property value.

20. Click **OK**.

5.3.3 Viewing Property Definitions

You can view partial property definitions on the **Properties** tab of an operation, endpoint, service, or transaction. The property name, phase, type, and description are displayed. The property's sensitivity and consumer mapped attribute is also displayed.

To see the full definition, click the **Edit** button and select **Modify**.

5.3.4 Viewing Property Values

You can view property values in the following views:

- **Message Log** tab for a selected endpoint, operation, service, or transaction

- **Alerts** tab for a selected endpoint, operation, service, or transaction

- Transactions view, **Instances** tab

- **Transaction Instance** inspector

5.3.5 Modifying Properties

You can change the definition of a property. Note that if you change the XPath definition of an existing property, the runtime will start calculating the new XPath definition based on incoming messages. Previously recorded messages will continue to hold the value calculated based on the old XPath definition.

To change the name of a property, you must create a duplicate, assign it a new name, and then delete the original.

Properties play an important role in transaction definitions, segmentation, conditions, and logging; for this reason, modifying a property definition might have the following consequences:

- If the property is used as a key in a transaction definition, the transaction will be disabled.

- If the property is used as a key in a transaction definition, the transaction will be disabled.

- If the property is used by a condition, the condition will be disabled.

To modify any user-created property

1. Select the **Properties** tab of the operation, endpoint, service, or transaction of interest.

2. Click the **Edit** button and select **Modify**.

3. Modify the property definition. Refer to the instructions for creating a property.

4. Click **OK** to save the new definition.

5.3.6 Duplicating Property Definitions

There are a couple of reasons why you might be interested in duplicating a property definition:

- You might want to use the original property as a starting point for creating another property that needs a similar complex expression or includes some of the content

that is extracted by this property. In this case, you would want to duplicate the property and partly modify its definition.

- You need to rename a property. In this case, you can duplicate the property, assign it the desired name, and delete the original property. Deleting properties can have significant side effects. See Section 5.11, "Troubleshooting Transactions" to learn more.

To duplicate any user-created property

1. Select the **Properties** tab of the operation, endpoint, service, or transaction of interest.

2. Click the **Edit** button and select **Duplicate**.

3. Modify the property definition. Refer to the instructions for creating a property.

4. Change the name of the duplicate if you wish.

5. Click **OK** to save the modified copy.

5.3.7 Removing Properties

Properties play a key role in transaction definitions, segmentation, condition definitions, and logging; for this reason deleting a property might have significant consequences:

- If a property is used as a correlation key, it is considered fundamental to the transaction definition. If one of these properties is deleted, the entire transaction will become disabled and a system alert will be generated.

- If a property used for transaction consumer segmentation is deleted, or modified so that it is no longer marked as "consumer," the transaction segmentation will no longer be captured. If the transaction is marked enabled for segmenting by consumer, a system alert will be generated and consumer segmentation will be disabled.

- If a property used by a condition definition is deleted, the condition will be disabled and a system alert will be generated.

- If a property is deleted which is defined on an operation in a transaction and logged (but not explicitly used for other purposes), the user will no longer be able to see that property and associated content when viewing instances of logged messages.

To delete any user-created property

1. Select the **Properties** tab of the operation, endpoint, service, or transaction of interest.

2. Click the **Edit** button and select **Delete**.

3. Click **OK** to save the new definition.

5.4 Defining, Modifying, and Deleting Transactions

A *transaction* is a sequence of service operations that you want to monitor and manage as one unit. This section explains how you use Business Transaction Management to define transactions, to modify transaction definitions, and to delete transactions.

Keep in mind that the more features you turn on for your transactions, the greater the impact on Business Transaction Management performance, especially with high volume. For example, it is best to restrict message logging to narrow areas of interest

rather than to enable it for all operations in a transaction. The following guidelines indicate the relative performance cost of transaction monitoring options:

- **Low**: core measurements when the start and end message are the same
- **Medium**: segmentation, condition evaluation
- **High**: instance logging and message logging

5.4.1 Defining a Transaction

Defining a transaction involves specifying the following information:

- The operations that participate in the transaction, and the start and end message for the transaction.
- Whether the transaction should be actively monitored or temporarily quiesced.
- Whether you want information to be segmented by ip address or consumer.
- Whether you want to log transaction instances or message content.
- How you want the system to correlate messages: using fingerprints or using manual keys that you define
- How long transaction messages should be held by the system.

The following subsections explain how you specify each aspect of the transaction definition.

5.4.1.1 Selecting Operations and Enabling the Transaction

Once you have defined the operations that participate in a transaction and enabled transaction monitoring, you will be able to view transactions and related services in the Transaction view, create fixed-value and baseline Service Level Agreements (SLA), view summary of link measurements on the Summary tab and other measurements in the appropriate views. For measurements, you will be able to see the number of started and completed transactions, throughput, average response time, maximum response time, and faults.

1. Select any operation that you want to include in transaction. You can select the operation from any view: service graph, dependency diagram, data grid, or message search tool.

2. Select **Transaction** from the **Create** menu. Business Transaction Management brings up the **Transaction** tool, which allows you to define all aspects of the transaction.

3. Select the **Definition** tab if it is not already selected. Note the basic elements required for a definition as shown on this tab: start message, end message, and maximum transaction duration. The **Enable Transaction Monitoring** checkbox is enabled by default. Disabling the box allows you to keep a transaction definition while temporarily turning off monitoring.

   In the upper pane, the tool displays all the operations that are dependent upon the operation you selected. By default, the request message of the left-most operation in the dependency chain is designated to be the start operation and the response message of the left-most operation is designated to be the end message. The tool displays the specified starting operation along with all the service operations in the consequent dependency chain.

   By default message fingerprints are used to correlate messages in a transaction. These are represented in the transaction map with a fingerprint icon. You can

choose to have Business Transaction Management correlate messages using manual keys if you like.

4. By default, the transaction name is based on the starting operation. To assign another name, specify it in the **Transaction Name** text box.

5. If you want to change the start and end message selected by default, click on the **Start message** or **End message** drop down lists and select an alternate start and end message.

    If the start and end message are not the request/response messages of the same operation, you will need to specify keys with common content for Business Transaction Management to measure performance. You will be prompted for the required information. You can specify an ECID value as a common key if you like.

6. To further customize the transaction, use the buttons to the right of the graph to add an operation, connect it to existing operations, to indicate message flow direction, or to remove an operation. You will also need to specify manual keys to connect the operation you have added to an existing operation in your transaction.

7. A new transaction is enabled by default. This means that Business Transaction Management is monitoring transaction performance, collecting measurements, and evaluating conditions if these have been defined for the transaction.

    To disable a transaction, click the **Enable Transaction Monitoring** check box. This will turn off measurement collection (segmented and non-segmented), condition evaluation, instance logging and content logging. However, SLA's applied to the transaction will continue to evaluate and potentially fire unless you explicitly disable them. Disabling a transaction is a way of quiescing transaction monitoring without losing the transaction definition.

8. By default the scope of the transaction (how long it takes for it to complete) is set to one minute. Business Transaction Management uses this value as a guideline in locating transaction messages; it defines the time window within which unique content is expected to appear. You can change this value to a longer duration for **Maximum Transaction Duration**), depending on the characteristics of your transactions. (The one minute value is both the minimum and the default setting.) The intent here is to close out transactions that tare not likely to complete normally, and to free needed resources.

If you decide to change the setting of JDBC Summary mode after creating a transaction, you will need to run more traffic and then update your transaction definition. For details, see Section 4.1.4.2, "Updating Transaction Definitions After Changing Summary Options."

5.4.1.2 Enabling Segmentation

Business Transaction Management allows you to segment transaction measurements based on ip address and by individual consumer. You use the **Segmentation** tab of the **Transaction** tool to enable segmentation. Enabling segmentation allows you to create usage SLAs for the transaction, to see transaction usage by client IP address, and to see transaction measurements by consumer. You can see on the **Profile** tab which segments are enabled and which property is being used as the consumer property.

You can choose to enable either or both of the segmentation options. The image below show what the tab looks like after segmentation has been enabled and a particular property has been mapped to a consumer. Note the consumer icon.

To enable segmentation:

1. Select the **Segmentation** tab of the **Transaction** tool

2. To enable client address (ip) segmentation, click the check box **Enable client address segmentation**.

3. To enable consumer segmentation, click the check box **Enable consumer segmentation**.

Consumer segmentation depends upon defining a property for the transaction's starting message. This property refers to the message element that you want to map to the consumer business object. If you have not already defined this property before you define the transaction, you can do so as shown in steps 4 through 9. If you have defined the property, select it now.

4. Click the **New Property** button. Business Transaction Management displays the **Create Property** tool.

5. Specify a name for the property, and provide a description if desired.

6. **Property Source**: Select the source for the property value: **Message content** or **Header**.

7. Select the **Pick from Message....** link, and select the element you want to map to the consumer business object.

8. **Consumer Mapping:** Click on the check box **Map to consumer through attribute Consumer Name**. You can change your selection later if you wish, but you cannot have more than one property mapped to a consumer object.

9. If you do not want the consumer name displayed exactly because it contains sensitive information, check the **Treat as sensitive** check box.

10. Click **OK**.

5.4.1.3 Enabling Instance and Message Content Logging

Enabling instance logging allows you to see a list of transaction instances captured in a given time period. These are displayed in the **Instances** tab of the **Transaction** view. You can inspect a given instance, view any property values for that instance, and create conditions based on property values. You can also assemble any of these instances.

To enable instance logging:

1. Select the **Logging** tab.

2. Click the check box titled **Enable instance and property logging**.

Enabling message content logging allows you to view message content for the operations you specify. (In this case, Business Transaction Management logs all messages for the specified operations, not just those belonging to this particular transaction flow.) You can get to the message content in different ways: use the **Message Log Search** tool to find an operation based on free text search and then open the related transaction instance, or you can drill through to the message log from the Analysis tab and alerts.

To enable message content logging:

1. Select the **Logging** tab, and click the check box titled **Enable message content logging on selected operations**.

2. Select the operations (messages) whose content you want to log. Logging message content is an expensive operation. It is best to narrow the scope of messages that interest you before you start logging messages.

The example above shows that message content is enabled for the operation `checkCredit` of the service `CreditService`.

5.4.1.4 Correlating Messages in a Transaction Using Manual Keys

By default Business Transaction Management correlates messages in a transaction by computing fingerprints (hash values) based on the content of each message and by pairing incoming messages with outgoing messages all the way to the end of the call chain. Defining manual keys provides an alternate means of correlating messages. Manual correlation is used to add a secondary operation flow to a transaction, to define a transaction whose end message and start message do not belong to the same operation, and to define a missing message condition.

Transaction messages do not have to be correlated by a single method: some messages can be auto-correlated; others can be correlated using manual keys. In order to define a manual key, you must map it to a message property and that property must satisfy certain requirements as explained next.

A manual key is mapped to a message property.

The property can correspond to a single element in a message, to a combination of elements, or to a fragment of an element. You can define the property before you define the manual key

The property you define for the manual key must satisfy two requirements:

- Its value must be the same for the messages you are correlating.

- Its value must be unique for the scope of the transaction (the scope is defined by the value of **Maximum transaction duration**).

For example, in a shopping type application, the order ID is often an excellent choice for a manual key.

You will need to manually correlate operations in the following cases:

- To add a secondary operation flow to the transaction. This might be necessary in the case of asynchronous systems like JMS or desirable if you want to connect two operation flows that share underlying data but are not causally connected. For example, you have one flow defined by an order entry, and another flow defined by the fulfillment of the order. The same order is involved in two otherwise disjoint processes.

  In this case, you will need to figure out which two operations you want to link, and to find a common element for those two operations that you can map to a message property. The process would be as follows: add the operation to the transaction; connect it to an existing operation in your transaction, and indicate the direction in which the request is going. The two operations will then be shown in the **Message Keys** tab, where you can connect them using a manual key.

- To define a transaction whose end message is not the end message of the operation that starts the flow.

  In this case, you will need to find a common element in the starting message and the desired end message that you can map to a message property. When you specify the end message, the system will prompt you for the manual key that will connect the start and end message.

- To define a missing message condition.

  In a missing message condition, an alert is generated if the target message you specify does not arrive within a given period of time after the start message occurs. In this case, you must define a key for some element that is common to the

start and target message. When you define the missing message condition, the system will prompt you for the key that will connect the start and target message.

You use the **Message Keys** tab of the **Create Transaction** tool to define the means by which Business Transaction Management correlates operations when you are adding a secondary flow. Note that if a given operation correlates to more than one operation in a dependency flow, you have the option of creating manual keys for each correlation. You do not have to use the same key to correlate to the three different operations nor do you have to use the same means of correlation.

To create a manual key to correlate messages:

You can define the properties you need before you define the correlations or you can access the property tool from the **Message Keys** tab and define the property at the same time that you are defining the correlation.

1. Select the **Message Keys** tab.

2. Select the start operation for the pair of operations you want to correlate.

3. Click the **Message Fingerprint** item for the **Start** operation of interest to see the key drop down list. Business Transaction Management displays a list of properties for the operation from which you can pick one to use as a manual key.

4. Select the desired property to use as a key.

5. If no property is displayed that will have a unique value on each invocation of the operation, select **New Property** from the drop down list to open the **Property tool** and use that tool to define the property you want to use as a message key.

6. Select the End operation that corresponds to the Start operation for which you have just created the key.

7. Repeat Steps 4 through 5 to select or define a key for the end operation. (Although the property names might be different, the value of the property must be the same, and it must be unique to the transaction instance.)

8. Repeat Steps 4 through 7 for each pair of operations that you want to correlate. (The properties you have created for the manual keys are displayed in the **Profile** tab for the selected transaction, in the **Properties** pane.) When you are done defining the transaction, the graphic that represents the transaction will be changed to show key icons rather than fingerprint icons for all operations that are correlated using a manual key.

5.4.1.5 Defining Storage Settings

The final step in defining a transaction is to define archival settings and to enable the indexing of message content.

Use the **Storage Settings tab** to define these values. The table below shows the default settings and explains the meaning of the settings.

Item	Default value	Description
Retain archived condition instances	30 days	The amount of time Business Transaction Management must retain archived condition instances in the transaction server's database.
		An *archived condition instance* is a transaction instance that is assembled and retained when a condition set on one of the operations belonging to the transaction evaluates to true. (Instance logging must be turned on when you define a condition if it hasn't already been turned on.)

Item	Default value	Description
Retain individual messages	24 hours	The minimum amount of time Business Transaction Management must retain logged messages in the transaction server's database.
Rotate message log	720 minutes	If you have turned message logging on, this specifies the rotation interval for logged messages. Messages that are rotated out are held in the message log database for the amount of time defined for **Retain individual messages**.
Text index message content	On	Creates a text index in the oracle database that speeds up keyword-based searches (at the expense of storage and some initial processing time). If you disable this feature, you can still do content-based searches, but they will be slower.

5.4.2 Modifying a Transaction Definition

Modifying a transaction definition changes the definition of the transaction in the environment. You can modify any part of a transaction definition, but keep in mind that Business Transaction Management does not track definition versions. For this reason changing some aspects of the definition might confuse things. For example, if you add or delete operations to a transaction definition or if you change the keys used to correlate operations, and then you try to assemble a transaction instance that preceded the modified definition, you might get odd results. On the other hand, enabling or disabling the transaction, or changing storage options should not be a problem.

To modify a transaction definition:

1. Select the transaction of interest in the **Explorer > Transactions** summary view.

2. Select **Edit** *TransactionName* from the **Modify** menu.

3. Use the **Modify Transaction tool** to modify the transaction definition. Refer to Section 5.4.1, "Defining a Transaction" for information about definition elements.

4. Click **OK**.

Business Transaction Management offers you a short cut you can use to enable or disable a transaction without having to edit its definition.

To enable or disable a transaction:

1. Select the transaction of interest in the **Explorer > Transactions** summary view.

2. Select **Enable** *TransactionName* or **Disable** *TransactionName* from the **Modify** menu. The appropriate action will be available, depending on the current status of the selected transaction.

5.4.3 Deleting a Transaction Definition

Deleting a transaction definition removes the definition from the environment. No condition alerts defined for this transaction are triggered. The system deletes all existing instances of this transaction.

To remove a transaction:

1. Select the transaction of interest in the **Explorer > Transactions** summary view.

2. Select **Delete** *TransactionName* from the **Modify** menu.

3. Confirm the deletion.

5.5 Viewing and Monitoring Transaction Performance

After a transaction is defined and if monitoring is enabled for the transaction, Business Transaction Management applies the default transaction measurement policy. In addition, if instance logging or message content logging is enabled for the transaction, it starts logging the appropriate information.

As traffic flows through the services that participate in a transaction, Business Transaction Management displays core measurements for the selected transaction in the Main area. This includes the number of started and completed transactions, the throughput, and the average response time. In the **Tabs** area it displays more detailed information about transaction performance and its definition. This includes the following:

- A **Summary** tab that presents summary performance information for the transaction and a map of the operations it includes.

- An **Analysis** tab that presents a graphic view of transaction performance over time as well as summary information for conditions, performance segmented by consumer usage or client address, and violation alerts. This tab also gives you the option of creating your own custom chart and breakdown.

- An **Alerts** tab that lists the alerts triggered for this transaction in a given period.

- An **Instances** tab. If you have instance logging enabled, you can use this tab to assemble an instance and view performance information for the instance.

- A **Message Log** tab. If you have message logging enabled, you can use this tab to view the contents of any request or response message for a logged operation.

- An **SLA Compliance** tab that shows the status and definition of service level agreements currently associated with this transaction.

- **Definitional** tabs. These tabs contain information about the transaction definition and the policies, conditions, and properties associated with the transaction.

This section explains how you use transactions attributes to filter your transaction view, how you view summary and analysis performance information for a transaction, and how you view and respond to alerts.

See Section 6.3, "About Instruments" for a detailed discussion of the instruments used to measure transaction performance.

5.5.1 Filtering the Main View of Transactions

By default all transactions are shown in the Main area when you select **Transactions** from the Navigator. You can have Business Transaction Management display a subset of currently defined transactions by using the Filter link or drop-down list. Using filters allows you to customize your view and to focus on the characteristics of interest.

5.5.2 Summary and Analysis of Performance Information

Business Transaction Management uses the **Summary** and **Analysis** tabs of the **Transaction** view to present summary and detailed analysis of transaction performance. This section describes these views.

5.5.2.1 Summary Tab

The **Summary** tab uses four panes and a grid view to present performance information in a **Transaction Summary Dashboard**. These are mostly self-explanatory, except for the **Delay Analysis** pane, which is described in greater detail below.

The **Transaction Summary Dashboard** includes the following:

- A **Status** pane showing the overall compliance for the transaction.

- A **Measurement and Baselines** pane showing the number of started and completed transactions, average response times and maximum response times. If you have defined baselines, these are shown as gray lines in the picture.

- A map of the transaction showing average response times for each transaction link. Place the cursor over each service icon to get detailed performance information for that service. The thickness of the arrows indicates throughput.

- The **Delay analysis** pane, which you can use in conjunction with the map pane, shows graphically what proportion of the overall response time is spent in each hop (link) of the transaction.

  Each colored area of the grid corresponds to a transaction link. Clicking in a colored region highlights its corresponding link in the map and displays the percentage of the response time taken up by that hop.

  At the bottom of this pane, a graph shows the average and maximum response times and the started transactions. Clicking in the pane displays a vertical red line that shows how the colored proportions correspond to message traffic flows.

- A grid view showing the logical and physical operations that make up the transaction, and the following instruments for each: violation alerts, average response time, maximum response time, throughput, and faults.

5.5.2.2 Analysis Tab

The **Analysis** tab of the **Transaction** view allows you to see the measurements in the Summary tab graphically rendered as trends across time. The **Analysis** tab offers six views of transaction performance data. These are described in the table below.

View	Description
Performance	Provides two color-coded charts: one that tracks started transactions, completed transactions, and conditions; the other tracks average response time versus maximum response time.
Conditions	Lists information about the conditions that have been triggered in a given time period: the name of the condition that was met, the endpoint where the condition alert was triggered, and the number of condition alerts triggered.
Consumer usage	If you have chosen to segment by consumer: Lists service consumers and the number of started and completed transactions in which they were involved and the average and maximum response time for each consumer.
Breakdown by Client Address	If you have chosen to segment by ip address: Lists the hosts for the services involved in the transaction, the number of transactions started and completed on each, and the average and maximum response time for each host.
Custom charting	Displays performance results for custom charts. Lets you set up a customized chart and table with instruments of your choosing.
	Click **Choose Instruments** and select the instruments you want displayed in the chart and table. You can select multiple instruments.

View	Description
Custom breakdown	Defines the custom performance information to measure and the segment on behalf of which measurement is done.
	Click **Choose Instruments** and select the instruments you want displayed in the table. You can select multiple instruments. Click **Choose Segments** and select how you want to segment the measurements. You can select multiple segments.

5.5.3 Viewing and Responding to Alerts

Alerts are shown on the **Alerts** tab for the selected transaction. You can use the **Filter** button in that tab to determine the range of time for which alerts should be displayed.

The system automatically generates alerts when one of the following happens:

- An SLA enforcement policy issues a failure, warning, or compliance alert.

- A condition evaluates to true

You can respond to alerts triggered by a condition by examining the assembled transaction instance and taking appropriate action.

5.6 Top 10 Transactions Dashboards

The **Top 10 Transactions** dashboard enables you to quickly identify and assess the health of the most stressed components in your system.

To display the dashboard, choose **Dashboards** > **Top 10 Transactions**

The dashboard provides four tables. Each table is based on a particular instrument and lists the transactions with the ten highest measurements for that instrument (except for uptime, which is lowest).

The default evaluation period for the data displayed for transactions is seven days. To change the evaluation period, click the Time Period control at the top of the display. You can change the period to the last day, hour, or 10 minutes.

The **Top 10 Transactions** dashboard provides tables that list the 10 transaction types with the following characteristics:

- highest number of started transactions (**Most Started** table)

- highest number of completed transactions (**Most Completed** table)

- slowest average response time (**Slowest Avg Response Time** table)

- highest number of condition alerts (**Most Condition Alerts** table)

Each table provides numeric instrument values as well as charts. Hover the cursor over a chart to view detailed information for a particular time segment.

For in-depth information and analysis, double-click a transaction or service to display the Tabs area, and select the **Analysis** tab.

5.7 Viewing Transaction Instances

A transaction usually executes many times in a given period. If you have enabled transaction instance logging or if you have enabled fault monitoring, Business Transaction Management tracks the flow of messages included in the transaction and maps these to particular *transaction instance*s. It assembles the messages for a transaction instance in the following cases:

- when an alert is generated as a result of a fault, or a condition being met

- when you explicitly ask for assembly

Once a transaction instance is assembled, you can use the **Instances** tab to view detailed performance information for that instance. You can also use the **Message Log** tab to search for messages containing particular property values. To view the content of a message, you must also enable message content logging.

This section explains how you do the following:

- View a summary of all transaction instances and search for a subset of these

- Assemble and inspect an instance

5.7.1 Viewing Captured Transaction Instances

The **Instances** tab in the Transactions **Tabs** area allows you to view captured transaction instances.

The ID column of the table displays both instances that have been assembled (these have an ID value assigned) and instances that have not been assembled (these have an **Assemble** button).

Information for each instance shows when it was captured, what the overall response time for the transaction instance was, and values for properties if you have created these.

The **Show instances** filtering control allows you to list instances that have occurred in a set time period or to show only assembled instances.

Which instances you choose to assemble depends on what interests you. For example, you might want to assemble an instance with an unusually slow response time; or you might want to assemble an instance with an unexpected property value.

If you are capturing a very large set of messages, you might want to use the **Message Log** tab to search for a smaller set of messages, based on property values, and then assemble one or more of these.

5.7.2 Assembling Instances and Using the Instance Inspector

You can assemble an instance by clicking the Instance Inspector icon (magnifying glass) in the **Instances** tab:

Business Transaction Management brings up a **Transaction Instance Inspector**. This view consists of three parts: identifying information for the transaction instance at the top, a map view in the middle, and a grid view at the bottom showing all the operations included in the transaction instance.

The top part of the inspector shows the name of the transaction, the time the assembled instance started executing, its ID, the number of message exchanges, the total messages exchanged, and the response time between the starting and ending message. Any warnings or faults will also be displayed in this area.

In the instance map that follows, you can view the entire transaction instance, with the response time given for each request/response link. Move the cursor over the operation name to display the service type, the endpoint name, and the host name and port. You have the option to hide uncalled operations and to hide JDBC calls. Hiding JDBC calls can greatly simplify the call graph for complicated transactions with many operations that call the database.

At the bottom of the inspector window, a grid view shows you detailed information for each message included in the transaction instance. The view includes columns to show property values if you have defined these. The information displayed might be affected by the setting of JDBC summary mode; for more information, see Section 4.1.4.1, "Understanding the Display of JDBC Summary Data."

Clicking on the magnifying glass (tear-off control) for any operation, opens a **Message Content** inspector window and displays the contents of the selected message if you have enabled message content logging for that operation.

5.8 Viewing Logged Messages and Properties

You can use the **Message Log** tab for a transaction to view the following information:

- If instance logging is enabled, you can view information about each message logged in a specified time period as well as the value of any property associated with a message. You can also use the **Message Log Search** tool to search for a message or messages that contain property values of interest or based on the ECID value associated with the message.

- If message content logging is enabled, you can view information about each message logged in a specified time period as well as its content. In this case, in addition to searching for messages based on property and ECID values, you can also search based on the content of any message element (free text search).

Business Transaction Management logs message content or instance and property values only if you have done the following:

- You have enabled monitoring for the transaction

- You have enabled the appropriate type of logging for the transaction (instance or message)

- For message logging, you have selected one or more operations

Logged information is stored according to storage settings that you define when you create the transaction.

This section explains how you view information about messages, search message logs for messages, and enable or disable logging.

5.8.1 Viewing Information About Logged Messages and Properties

The **Message Log** tab for a given transaction displays information using a grid view that displays a list of messages, showing the arrival time of the request message, the service that includes the selected operation, the location of the endpoint that implements the service, the operation (message), and the type of operation. If there are any properties associated with the operation, their values are shown in additional columns whose title is the property name.

If you have message content logging enabled, double clicking on any message shows you the contents of the message.

The set of messages shown in the grid varies depending on the setting of the filters shown at the top of the tab. These allow you to see

- All operations or specific operations chosen from a drop down list

- Any response, only successful operations, only failures

- Messages that arrived within a time interval denoted by the last specified time period, since a certain time, or between two given times.

You can use these controls to narrow the selection of messages shown in the grid. After you change filter settings, click **Search** again to repopulate the grid.

You can further narrow your search by using the **Message Search tool** accessed from the **Choose Content...** link. This allows you to search for messages based on their property values or, if message content is enabled, based on message content. This tool is described in the next section.

5.8.2 Using the Message Log Search Tool

The **Message Log Search** tool enables you to further refine the scope and the criteria of your search (in addition to the strictures defined by the view filters). To bring up this tool, choose **Search Message Log** from the **View** menu. This allows you to find messages belonging to any monitored transaction for which logging is enabled.

You can also find messages belonging to the currently selected transaction, by clicking the **Choose Content...** link from the **Message Log** tab of the transaction. However, this more restricted search does not give you the additional option of searching for related transactions when you obtain the search results.

The **Message Log Search** tool has four main areas you can use to define search criteria: one area for searches using user-defined properties, an area for ECID-based searches, one area for searches related to the properties `aggregateCount` and `aggregateFaultCount,` and one labeled Free text search. You use controls in these areas to search for a set of messages that match you criteria. As you specify criteria, a search expression is constructed in the text box at the top of the tool. To clear the box and start over, press the **Clear** button.

For additional information about using Oracle query language to formulate your query, look at the following:

```
http://download.oracle.com/docs/cd/B28359_01/text.111/b28304/cqoper.htm#BABBJGFJ
```

When you are done defining the expression to be used in the search, click the **OK** button. Then click the **Search** button to repopulate the grid according to your newly defined search criteria.

The following sections provide additional information for each type of search; however, note that you are not restricted to a single type: you can use them in any combination. For example, you can search for messages that have a given property and ECID value.

5.8.2.1 ECID-Based Search

If a message (operation) has been assigned a request or response ECID, you can search for those messages that have been assigned a specific ECID value. For information on the meaning and assignment of ECIDs, see Section 5.9, "Accessing Other Diagnostic Tools."

To do an ECID-based search, select the **matching** criteria in the ECID field and enter or paste the ECID value in the ECID entry field. You can copy the ECID value from anywhere it is displayed. If you are copying from a column, expand the column to make sure you have the full string value. The Message Log Search tool, displays all operations with the specified ECID in all transaction instances in a grid view.

Figure 5–1   ECID-Based Search

If you want to look at a specific transaction instance that contains a given ECID, select the operation of interest in the grid view of the Message Log Search tool and select **View > Related Transaction > target_Transaction** in the Message Log Search tool. The instance of interest is displayed in the **Instance Inspector** window.

5.8.2.2 Property-Based Search

The properties that have been defined for the transactions in question are shown in the **Message property search** area. Use the drop-down lists to specify your property-based search criteria.

- For a numeric value, the drop down list allows you to search for messages whose properties are less than, equal to, greater than, not equal to, and so on.

- For a string value, the drop-down list provides a similar set of matching options.

5.8.2.3 Free Text Search

To use this search option, you must have enabled message content logging.

Enter a string you want to search for in the box labeled **Message contains the text**. Do not enclose the string in quotation marks unless you're looking for a string enclosed in quotation marks.

5.8.3   Enabling and Disabling Message Logging and Instance Logging

Do the following to enable or disable message logging and instance logging:

1.   Select the transaction of interest.

2.   Select **Modify > Edit** *TransactionName.*

3.   In the ensuing dialog, select the Logging tab.

4.   Use the check boxes shown to enable or disable message content logging and instance logging.

5.   If enabling message logging, select the specific operations you want to log.

5.9 Accessing Other Diagnostic Tools

In addition to the metrics it provides about transactions, services, endpoints, and operations, Business Transaction Management allows you to view additional

information about the JVM context within which a request executes and to see more detailed information about the request as it executes across distributed JVMs.

The additional views are the following:

- **The JVMD view** allows you to view the details of an executing Java Virtual Machine (JVM) process for the period within which a given operation executes. You can see stack frames for executing threads, thread state information, aggregate information about the frequency and cost of method execution, information regarding the holding of Java and DB locks, and details about the objects in the Java heap. JVMD also stores historical data for each JVM it monitors so you can view data relating to things that have happened in the past and get a sense for historical trends.

- **The Request Instance Diagnostics view** allows you to trace the path of a request in a WebLogic domain and to generate a report of all the metrics associated with a particular instance of the request.

The following sections describe the information you can get from these views as they relate to transaction operations and explain how you can access these views from the Business Transaction Management console.

For information on how to configure Business Transaction Management to access these views, see Section 5.9.3, "Enabling Access to the JVMD and RID Views."

5.9.1 Using the JVMD View

Using JVMD to access additional information can be useful in situations like the following:

- You are viewing summary information for a transaction and see that a particular operation is unusually slow or faulty. You want to get more information about the execution of that step, and you initiate a drill through to JVMD from this point.

- You are viewing the details of a logged transaction instance and you notice that one of the steps in the transaction did not behave as expected. You would like more details about the execution of this step from JVMD, and you initiate a drill through to JVMD from this point.

- You are viewing a collection of logged messages in the message log search tool and you notice a message that appears abnormal or unusual. You would like to get more details of the handling of that message using JVMD, and you initiate a drill through to JVMD from this point.

You can access the JVMD console from the following entry points:

- **Message Log tab** for a service, endpoint, logical operation, physical operation, or transaction. Right-click on row and select **Drilldown to JVMD** from the drop-list.

- **Transaction Instance Inspector.** Right click on an operation (either in the graph or grid view) and select **Drilldown to JVMD** from the drop-list. When a step is executed repetitively, the first endpoint on which it was executed is selected.

- **Message Search Log tool** Right-click on a message row and select **Drilldown to JVMD** from the drop-list.

In each of these cases, a new window is displayed showing the JVMD view. In the multi-VM case, JVMD shows a VM group target and aggregate information for that group.

For information on how to enable access to the JVMD page, see Section 5.9.3, "Enabling Access to the JVMD and RID Views."

5.9.2 Using the Request Instance Diagnostic View

To view information about an operation (message) in the Request Instance Diagnostic view, that message must be assigned an Execution Context Identifier (ECID). This section explains what an ECID is, how ECIDs are assigned, how they map to transactions, and how you can access and use the Request Instance Diagnostic view from the Business Transaction Management console.

Data is displayed in the Request Instance Diagnostics view only for operations executing in a WebLogic (10.3.x) domain with JRF 11.1.1.1.0 or higher. (The installation of SOA 11g, OSB 11g, and Fusion Applications already includes the JRF.)

For information on how to enable access to the Request Instance Diagnostics page, see Section 5.9.3, "Enabling Access to the JVMD and RID Views."

ECID information for a message request or response is displayed in the Business Transaction Management console even if you have not installed a JVMD agent or configured a connection to the EM console; however, without having satisfied these pre-requisites, you will not be able to open the Request Instance Diagnostic view for a given ECID.

You can use the **Message Log Search** tool to find all messages that have a given ECID. For more information, see Section 5.8.2, "Using the Message Log Search Tool."

5.9.2.1 ECID Basics

An ECID is an identifier for tracking a request, for components into the Oracle technology stack. It is usually generated by the outer-most Oracle component handling the request and may be propagated by the Oracle components handling that request, even crossing server boundaries. The creation and propagation of ECIDs enable the sharing of context and of diagnostic data between components.

Transaction operations (messages) that have been handled by Execution Context aware components represent steps in a request that were assigned an ECID shared by all the request steps. If Business Transaction Management observes that a message has an associated ECID, it assigns the ECID as an intrinsic property of the message request and response, and it notes the value of the ECID in the message header. The ECID value for a given operation and phase is stored in the standard properties `requestECID` and `responseECID`.

Because transactions might span multiple Oracle execution contexts, it is possible for different messages in a transaction to have different ECIDs. It is also possible for the same ECID to be shared by operations in different transactions. The next section describes how ECIDs are mapped to transaction operations (messages).

In addition to using ECIDs to drill down to the JVMD or RID views, ECIDs are also useful in the following situations:

- You want to define a transaction where the start and end message are not the same. In this case, you can use a common ECID value to correlate the start and end message. This enables BTM to measure performance for such a transaction.

- You want to manually correlate two disconnected operations using ECID as the shared key value.

- You want to search for a message that has a specific ECID value.

5.9.2.2 The Correspondence between ECIDs and Transaction Operations

Depending on how a transaction has been defined and deployed, the messages it contains might have one or more ECIDs. It's important to understand the mapping between ECIDs and transaction messages. This section describes two possibilities.

- Multiple transaction definitions; one ECID

  Consider a call chain A -> B ->C ->D ->. If I define transaction X to include ABC, and transaction Y to include CDE, the messages in an instance of X and an instance of Y that resulted from a single invocation will have the same ECID.

- Multiple ECIDs; one transaction.

  If transaction X includes the call chain A -> B ->C, and B executes on a non-Oracle platform, the message that corresponds to operation B will have no ECID, and the messages for A and C will have different ECIDs.

- For a given operation, the ECID for a request may not be the same as the ECID for a response.

In short, seeing multiple ECIDs for a single transaction, or the inverse, does not mean that anything is wrong; it can provide valuable information about operation flows and invocation boundaries.

5.9.2.3 Accessing the Request Instance Diagnostics View

The Request Instance Diagnostics view, which provides information about a given ECID, is accessible from the following locations:

- **Transaction Summary tab**. Right-click on an operation (in the graph or grid view) and select **Request Instance Diagnostics** from the drop-list.

- **Message Log tab** for a service, endpoint, logical operation, physical operation, or transaction. Right-click on row and select **Request Instance Diagnostics** from the drop-list.

- **Transaction Instance Inspector.** Right click on an operation (either in the graph or grid view) and select **Request Instance Diagnostics** from the drop-list.

- **Message Search Log tool** Right-click on a message row and select **Request Instance Diagnostics** from the drop-list.

It is possible that the Request Instance Diagnostics page displays no information for a given ECID. This can be because the default sample rate for the JVM monitoring tool is longer than the step duration, and the specified ECID executed outside of the sampled period.

The Request Instance Diagnostics page displays a list of the JVMs through which request steps with the specified ECID executed. For each JVM the following details are displayed:

- One or more target JVMs where the request executed.

  JVMs can show up more than once with the same ECID if the RID (relationship ID) is different for a request step. (The RID is a sequence of digits that represents the relationship of the current step's execution to the execution of the request as a whole.)

- The start time and duration of the request.

- The request name, the individual step in the request; for example, jsp, EJB, or DB.

- The CPU and memory use by the JVM.

- Garbage collection statistics (GC Major/Minor).

The bar graph displayed at the bottom of the Request Instance Diagnostics view shows the thread state in each JVM snapshot taken within the duration of the request. Each color represents a different thread state; a color key is shown on the right.

You can click on any bar in the bar graph to display a pane with detailed information for the given thread and the thread stack.

The **Request Aggregate** tab does not contain information relevant to transactions.

You can access the Request Instance diagnostics page in different ways. For example, you notice that a particular operation is taking an unusual amount of time and you want more information about its runtime context. If an ECID is supplied for the operation, you can drill in to the Request Instance Diagnostics page and get additional information. Alternately, you obtain an ECID from an external source and you want to see whether messages bearing this ECID participate in any transaction. You would use the Message Log Search tool to search for all messages associated with this ECID. Having found the messages, you can see the corresponding services and operations, view related instances, and find the transaction instance that included the message of interest.

5.9.3 Enabling Access to the JVMD and RID Views

> **Note:** You can only access JVM Diagnostics from BTM if the machines where JVMD is installed are known to the Enterprise Manager instance by their fully qualified domain names (FQDN). If a machine has been discovered/registered by the EM in its short form (for example `server1` instead of `server1.oracle.com`) then you will not be able to access JVM Diagnostics for that specific server. Access to JVM Diagnostics for machines that have an FQDN will continue to work.

To enable access to the JVM Diagnostics and Request Instance Diagnostics views, you must do the following:

- Install the JVMD agent on all nodes hosting your monitored services.

- Deploy the JVMD manager in EM.

- Install a Business Transaction Management observer on all nodes hosting your monitored services. (This would already be the case if you are able to observe traffic.)

- For RID only: enable instance logging. To access message content, you must also enable message content logging.

- Configure Business Transaction Management to access the EM console as explained next.

> **Note:** ECID information is displayed in the Business Transaction Management console even if you have not installed a JVMD agent or configured a connection to the EM console; however, without having satisfied these pre-requisites, you will not be able to open the Request Instance Diagnostic view for a given ECID.

Connecting to the EM Console

The following procedure explains how you connect to the EM console. This is required to enable drill down to the JVM Diagnostics page and to the Request Instance Diagnostics page.

1. Select **System Services** from the Navigator.

2. Find and select the **AP_Sphere_Service** entry.

3. Select **Admin > Edit Setup Data for AP_Sphere_Service**.

4. Click the **Edit XML** button.

5. Scroll to the bottom of the XML document and find the **emgcURL** element.

6. Edit this XML element so that it contains the base URL for the EM console. For example, if you have an EM console running at `https://adc2101158:4473/em`, you would enter `https://adc2101158:4473`, and the edited entry would look like this:

   ```
   <pfx6:emgcURL>https://adc2101158:4473</pfx6:emgcURL>
   ```

   Note that the namespace prefix might be other than `pfx6`; use whatever value appears in the XML text.

7. Click the **Apply** button to save changes.

You will have to supply login information for the JVM Diagnostics console when you access it. Single sign on is not supported.

5.10 Getting Information About a Transaction

Once you have defined a transaction, you can use detail area tabs to get information about policies, conditions, and properties applied to a transaction. You can also display the transaction's profile. In some cases, you can modify the definition or status of elements shown in a tab.

5.10.1 Getting Profile Information

Use the **Profile** tab of the **Transaction** view to see a map of the transaction and to see its definition.

The **Profile** tab provides essentially the same information you would see if you displayed the transaction definition by choosing **Edit** *TxName* from the **Modify** menu. Selecting the **Profile** tab instead, makes it easy to switch to other tabs or views, prevents your making unintentional changes, and provides three additional pieces of information:

- the date the definition was last modified,

- any user attributes defined for the transaction

- the transaction identifier, which is sometimes needed to identify the transaction in CLI commands.

5.10.2 Getting Policy Information

Use the **Policies** tab to view information about policies associated with a transaction.

By default, the tab shows information about applied policies. You can use the filter control to view changed policies, disabled policies, pending policies, rejected policies and unapplied policies.

The name of the applied policy is shown in a tree view in the **Name** column. Expanding the policy node shows the following information:

- **Policy Status Details** lists any issues arising from the application of the policy.

- **Monitored Object Type** specifies the targets to which the policy is applied.

- **Location** specifies the address of a target endpoint

- **Management Intermediary** specifies the Business Transaction Management agent that is applying the policy.

Double clicking the policy name in the **Policy** tab, opens a new window that you can use to view alert, profile, and target information for the selected policy.

5.10.3 Getting and Changing Condition Definitions

Use the **Condition** tab for the **Transaction** view, to display the conditions you have defined for a transaction. This tab allows you to do the following:

- View and change the status of fault monitoring: enabled or disabled.

- View condition definitions and status

- Use the **Edit** drop down list, shown for each condition, to modify the condition definition, to duplicate the condition, or to delete the condition.

5.10.4 Getting Property Information

Use the **Properties** tab for the **Transaction** view, to display a list of all the properties defined for messages included in a given transaction. In addition to listing the properties, the tab shows information about the following:

- The service and operation for which the property is defined

- The phase (request/response) of the operation

- The data type of the property value

- Whether the value is deemed sensitive

- Whether it is mapped to a consumer (denoted by a human icon on the left) and what consumer mapped attribute it is associated with.

- A description if you have supplied one when you created the property.

An **Edit** drop down list for each property listed allows you to modify, duplicate, or delete the property definition.

5.11 Troubleshooting Transactions

This section offers information to help you resolve problems in working with transactions.

If performance monitoring is not working as expected for transactions, the two most common issues to consider are the following:

- Has there been any traffic flowing through your services in the period of time for which you are expecting results? Business Transaction Management cannot measure the performance of inactive services.

- Has the feature upon which measurement or analysis depends been enabled? *Feature* refers to consumer segmentation, instance logging, message content logging, and most fundamentally, transaction monitoring. Check the **Profile** tab for the transaction to make sure that the desired features are enabled.

If monitoring is enabled for the transaction, if traffic is flowing during the period for which you seek results, and if the required features are activated, you should be seeing complete and correct data. If not, read through the following discussion for possible issues and resolutions.

> **Note:** Remember to check **Alerts in the last hour > System Alerts** for information that might pertain to your problem.

You cannot create a transaction

Business Transaction Management is unable to validate your transaction definition and cannot create the defined transaction. Possible causes include the following:

- The name of an operation in the transaction contains a special character. Business Transaction Management can process names with periods in them, but other special characters might pose a problem.

- You selected a start operation that is used by another transaction.

- You chose an end operation that differs from the start operation, but did not specify manual keys to correlate the start/end messages.

- There are disconnected flows in your transaction definition.

- The start and end operations are different and use different message log databases.

Your transaction has been disabled

You have not explicitly disabled monitoring for the transaction, but no measurements are gathered, and logging and condition processing are disabled. When you check the status shown in the **Profile** tab, you find that the status for a transaction is disabled or system disabled.

This can happen for a number of reasons:

- You have modified or deleted a property that was used as a key in the transaction definition.

- You have deleted a service or operation that was included in the transaction definition.

- You have disabled a service that was included in the transaction definition.

- For transactions where the start operation is not the same as the end operation, one or more of the start/end operation endpoints no longer shares the same message log database.

- You delete a property used for consumer segmentation, or user edits a consumer property used for consumer segmentation and removes consumer mapping information.

Performance measurements are incomplete or inaccurate

This can happen for a number of reasons:

- The value you specified for **Maximum transaction duration** is too short for your transaction.

- You have defined two or more transactions that share an operation. Total response time for each of the transactions will be accurate, but the link measurements to the shared operation will show the sum of the invocations from each caller. In the following diagram, showing transaction ABC and YBC, response time for the BC link will be a sum.
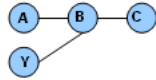
- The transaction has not finished executing.

- Business Transaction Management uses the start and end operations to define response time. It is assumed that these operations are called only once within the interval defined for **Maximum transaction duration**. It is also assumed that the content that appears during that interval is unique. If these requirements are not satisfied, performance measurements generated for the transaction might not be accurate.

Segmented information is missing

Consumer segmentation is enabled but information for the consumer is not collected.

- Check to see whether a property used for consumer segmentation has been deleted or modified so that it is no longer mapped to a consumer.

- If you have imported the transaction definition from a previous release, consumer mapping information is not imported. You will have to modify the transaction definition and recreate the mapping information.

Logged instances or messages do not contain the expected information

For logged instances, check to see whether the property that is associated with the expected information has been deleted.

It is a requirement that all messages for one transaction be logged to the same database. Sometimes, it is not possible to satisfy this requirement because the transaction crosses a firewall. In this case, you might need the assistance of Business Transaction Management support.

For logged message problems, you might need to check log policy settings and report what you find to Business Transaction Management support.

A condition has been disabled

Check to see whether a property used in the condition definition has been modified or deleted.

You disable instance logging and there is at least one enabled condition for the transaction that depends on it.

A condition has not triggered an alert

The Transaction server was down when a condition needed to be evaluated; check the status of your servers.

A missing message condition is erroneously triggered

A missing message condition is triggered, but the message exists and has been collected. This can happen when monitor processing is especially slow and the missing message condition is evaluated after the transaction's time to live interval has expired. A rare situation.

5.11.1 Viewing Log Policy Settings

Based on the storage settings for the transaction, Business Transaction Management creates a log policy for each transaction and refers to this policy when monitoring the transaction and storing logged messages. Log policy settings are partly derived from the information you provide when you define a transaction. The following table shows the correspondence between log policy settings and transaction definition runtime settings.

Log policy fields	Transaction definition runtime settings	Description
Guaranteed storage time	Retain individual messages Default: 24 hours	The amount of time Business Transaction Management must retain unassembled messages in the transaction server's database. These messages are thus available for assembly.
Rotation interval (minutes)	Rotate message log Default: 720 minutes	Specifies the rotation interval for the log policy that backs the transaction. Messages that are rotated out, are held for the amount of time defined for **Retain individual messages**.
Text index message content	Text index message content	Creates a text index in the oracle database that enables faster keyword-based searches (free text search) at the expense of additional storage and initial processing.

The log policy is named **System Generated Log Policy for** *TransactionName*. The only time you might need to view this policy is when troubleshooting logging problems.

To view log policy settings for a transaction

1. In the Navigator, select **Administration > System Policies**.

2. In the Main area, select the **System Generated Log Policy for***TransactionName* entry.

3. Open the tab area if it is not already open and select the **Profile** tab. Settings for the transaction's log policy are displayed.

6

Monitoring Services

This chapter describes how you can use Business Transaction Management to monitor services. It explains the meaning of the instruments used for monitoring, and it describes the **Top 10 Services** dashboard. It includes the following sections:

- Ways of Monitoring Services
- Top 10 Services Dashboard
- About Instruments

6.1 Ways of Monitoring Services

This section lists the ways you can monitor services. The list orders these monitoring tasks from the most general to the most specific, and explains, for each task, how you navigate to the view where you can perform the monitoring.

To monitor:

- Overall status of services, that is total number of services that are up/down and with SLA warnings/failures

  Navigate to **Dashboards** > **Operational Health Summary** > **Services**

- Most stressed services

  Navigate to **Dashboards** > **Top 10 Services**

- Current summary information across all services, endpoints, and operations (includes up/down status, SLA compliance, and performance)

  1. Navigate to **Explorer** > **Services to Endpoints** or navigate to **Explorer** > **Services to Operations** and look at the Main area.

     Services To Endpoints provides a physical view, letting you drill down from services to endpoints and then physical operations. **Services To Operations** provides a logical view, letting you drill down from services to logical operations.

  2. Select the service or endpoint of interest in the Main area.

  3. Select the **Summary** tab.

- Detailed current performance and usage for a specific service, endpoint, or operation

  Navigate to *specific service, endpoint, or operation* > **Analysis** tab

- Detailed current SLA compliance for a specific service, endpoint, or operation

  Navigate to *specific service, endpoint, or operation* and click the **Compliance** tab

- Recent history of issues affecting a specific service, endpoint, or operation

  Navigate to *specific service, endpoint, or operation* and click the **Alerts** tab

- Logged messages

  Navigate to *specific service, endpoint, or operation* and click the **Message Log** tab. Message logging is available only if a service endpoint is part of a transaction and if message logging is enabled for the transaction.

6.2 Top 10 Services Dashboard

The **Top 10 Services** dashboard enables you to quickly identify and assess the health of the most stressed services in your system.

To display the top ten services, Choose **Dashboards** > **Top 10 Services**.

The dashboard provides four tables. Each table is based on a particular instrument and lists the services with the ten highest measurements for that instrument (except for uptime, which is lowest).

The default evaluation period for the data displayed for services is seven days. To change the evaluation period, click the **Time Period** control at the top of the display. You can change the period to the last day, hour, or 10 minutes.

The **Top 10 Services** dashboard provides tables that list the 10 services with the following characteristics:

- highest throughput (**Most Load** table)

- lowest uptime (**Uptime Issues** table)

- slowest average response time (**Slowest Avg Response Time** table)

- highest number of vaults (**Most Faults** table)

Each table provides numeric instrument values as well as charts. Hover the cursor over a chart to view detailed information for a particular time segment.

For in-depth information and analysis, double-click a transaction or service to display the Tabs area, and select the **Analysis** tab.

6.3 About Instruments

Business Transaction Management uses a variety of instruments to measure the performance and usage characteristics of your business transactions and underlying services and operations. These instruments are displayed in various parts of the Management Console for interactive monitoring. You can also use most of these instruments as a basis for defining service-level agreements (SLA).

The period over which these instruments operate is either the *evaluation period*, in the case of an SLA, or the *display period*, in the case of interactive monitoring in the Management Console. The following descriptions use the term *period* to mean the evaluation period and/or display period, depending on the context in which the instrument is used. Some instruments, for example current compliance status, provide a current value only.

6.3.1 Transaction Instruments

The following instruments are available for monitoring transactions.

Average Response Time

The average amount of time a transaction requires to complete. For each instance of the transaction, the instrument measures the time from when the instance's start message is observed until its end message is observed. The instrument keeps a running average of the response time across all instances observed during the period. All completed instances are counted in the response time, regardless of whether condition alerts occurred.

If no transactions are observed during the period, the instrument value is set to -. Response time is measured in milliseconds.

Maximum Response Time

The maximum amount of time a transaction requires to complete. The instrument records the single highest response time from all instances of the transaction observed during the period.

Completed Transactions

The number of instances of a transaction that complete during the period. An instance is considered to have completed when both its start and end messages have been observed, regardless of whether condition alerts occurred. However, if the end message is defined as being in the response phase (for example, submit.response) and the end operation faults, the end message will not exist and the instance will, therefore, not be counted.

Completed Transaction Rate

The number of instances of a transaction that complete per hour during the period. This instrument derives its measurements from the completed transactions instrument.

Started Transactions

The number of instances of a transaction that start during the period. An instance is considered to have started when its start message is observed.

Started Transaction Rate

The number of instances of a transaction that start per hour during the period. This instrument derives its measurements from the started transactions instrument.

Condition Alerts

The number of condition alerts generated on the transaction during the period.

Condition Alert Rate

The number of condition alerts generated on the transaction per hour during the period. This instrument derives its measurements from the transaction condition alerts instrument.

Current Compliance Status

The current compliance status for the transaction.

Violation Alerts

The number of SLA violations or warnings caused by a transaction during the period.

6.3.2 Service and Operation Instruments

The following instruments are available for monitoring services, endpoints, and operations.

Average Response Time (services, endpoints, and operations)

The average amount of time a service or operation requires to respond to a request. For each request, the instrument measures the time from when the service receives the request until it sends a corresponding response to the client. The instrument keeps a running average of the response time across all messages received during the period.

Only successfully processed requests are counted in the response time; the response times for faults are not figured into this measurement. The response time is measured individually for each operation. The response time for a service is the average response time of all of its operations. This average is weighted according to the number of messages processed by each operation.

If no requests are observed during the period, the value of the instrument is set to -. Response time is measured in milliseconds.

Maximum Response Time (services, endpoints, and operations)

The maximum amount of time a service or operation requires to respond to a request. The instrument records the single highest response time for all requests received during the period.

Link Average Response Time

The average response time to outbound requests. For example, imagine a hypothetical `orderService` that receives a request from some client, and as a result sends a request to a `creditCheckService`. In this case, `orderService` is acting as a client to `creditCheckService`. The response time is measured from the point of view of the service that is acting as a client. In other words, it measures the time from when the client service sends the request until it receives the response, meaning that network latency, if it exists, is included in the response time.

Only successfully processed requests are counted in the response time; the response times for faults are not figured into this measurement. If no requests are observed during the period, the value of the instrument is set to -. Response time is measured in milliseconds.

Traffic

The number of requests that a service or operation receives during the period. The traffic count equals the throughput plus the fault count. Traffic count is measured individually for each operation. Traffic count for a service is the total traffic count of all of its operations.

Throughput

The number of requests that a service or operation successfully receives, processes, and responds to during the period (in other words, the number of responses). A message that generates a fault is not counted by the throughput instrument. Throughput is measured individually for each operation. Throughput for a service is the total throughput of all of its operations.

Throughput Rate

The number of successfully handled requests per hour during the period. This instrument derives its measurements from the throughput instrument.

Link Throughput

The number of outbound requests to another service that are successfully received, processed, and responded to during the period (in other words, the number of inbound responses; see the link average response time instrument for an explanation of service-to-service calls).

Faults

The number of faults generated by a service or operation during the period. Fault count is measured individually for each operation. The overall fault count for a service is the total fault count of all its operations.

Fault Rate

The average number of faults generated per hour over the period. This instrument derives its measurements from the faults instrument.

Fault Percentage

The percentage of messages that cause faults during the period. This instrument derives its measurements from the faults and traffic instruments.

Link Faults

The number of faults generated by outbound requests to another service during the period (see the link average response time instrument for an explanation of service-to-service calls).

Current Compliance Status

The current compliance health for the selected object.

Violation Alerts

The number of SLA violations or warnings caused by a service or operation during the period.

Violation Alerts Percentage

The percentage of time that a service or operation is in a state of SLA violation or warning during the period.

Failure Alerts

Count of failure violations for the specified period.

Warning Alerts

Count of warning violations for the specified period.

Uptime

The percentage of time that an endpoint's container responds successfully to a periodic ping message. See the `configureAlivenessCheck` command for details on how you can specify the method to be used for aliveness checking.

7

Working with Consumers

This chapter explains how you can present performance results to reflect consumers' use of your services. It explains how you set up consumer segmentation, how you create consumers, how you map messages to consumers, and how you view and manage consumer-based performance information. It includes the following sections:

- About Consumers
- Mapping Messages to Consumers
- Monitoring Consumers
- Adding and Deleting Consumers
- Editing Consumer Names
- Customizing the Consumer Label

7.1 About Consumers

In addition to presenting performance results based on the performance of individual transactions, services, and endpoints, Business Transaction Management allows you to view performance results in terms that are meaningful to your business. You can see transaction performance segmented by business criteria: your suppliers, your users, your re-sellers, or your customers. And you can see this information even in those cases when the object of interest participates in a series of different transactions. To this end, Business Transaction Management provides a *consumer business object* that you can define to segment performance data using criteria that is important to you. You can then monitor your business objects to make sure that your objectives are being met. For example,

- You can view custom graphs showing performance segmented by a given consumer.
- You can have Business Transaction Management generate alerts when the performance of your system for a given consumer does not meet the targets you define.

Consumers refer to Business Transaction Management consumer business objects; a *consumer* is an instance of the Consumer business object you have defined for your environment. The default name "Consumer" implies that this business object could be the initial client of a transaction; but it could also be the country or city where an order originates, a company name, and so on. Throughout this section, we will use the default name "consumer," with the understanding that this can refer to the wider range of criteria just mentioned.

To make the actual segmentation criteria more clear, the Business Transaction Management administrator can customize the consumer label that appears on fields, buttons, and links in the Management Console. For example, your administrator could change the label from "Consumer" to "Department", "Supplier", or any other descriptive term.

The following sections explain how you set up consumers:

- Setting up Consumer Segmentation
- Creating Consumers
- Changing the Default Consumer Name

7.1.1 Setting up Consumer Segmentation

In the simplest case, to obtain performance segmentation by consumer, you must do the following:

1. On the start operation for a transaction, create a property whose value is the request message element that uniquely identifies the consumer. It could be an ID number, a social security number, a name, or an email address. If the consumer is not a person, it could be a company name, a code identifying the company, and so on. (The information you choose to supply for the property value can be based on a single message element, an xpath expression, or content in the message header.)

2. Map this property to the consumer attribute on the consumer business object. This mapping process is called *mapping messages to consumer*.

3. Enable consumer segmentation in your transaction definition.

The figure below shows a simple mapping between a property called `email` (associated with the `request.submit.order.customerEmail` element of the starting message of a transaction) and the default attribute of the business object, `Consumer Name`.
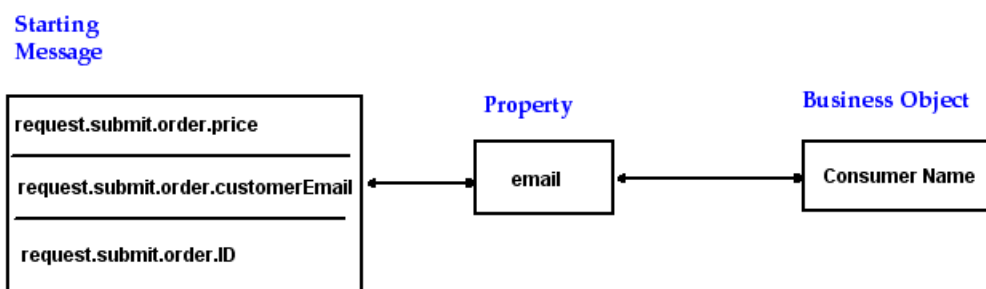
If consumer segmentation is enabled for the transaction, once traffic starts to flow, Business Transaction Management inspects the value of the `email` property of this transaction as part of its measurement gathering. For this example, let's assume the property value is `123hop@gmail.com`:

- If it finds an existing consumer whose name is 123hop@gmail.com, it records measurements for that consumer.

- If it does not find a consumer of that name, it creates the consumer, assigns it the value of the property as its name, and records measurements for it.

The business object that Business Transaction Management provides by default is adequate for the simple case where you are monitoring one specific kind of consumer and you are satisfied to use the value of the property as the consumer name. In reality,

you might need to handle more complicated situations. To do that, you can customize the definition of the consumer business object by adding custom attributes. Using these attributes, you can teach Business Transaction Management to recognize the same consumer though the consumer is identified by different properties.

For example, what happens if we want to monitor several transactions, and in each transaction the consumer of interest is identified in a different way? In one transaction, the consumer is identified by email address, and in the other, by a social security number.

The following figure shows two transactions, Tx1 and Tx2. Tx1 has a `customerEmail` element mapped to the `email` property; Tx2 has a `custID` element mapped to a `SSNumber` property. You want Business Transaction Management to monitor both transactions and to recognize that the same consumer is involved whether that consumer is identified by the `email` property or by the `SSNumber` property. To do this, you must add two custom attributes to the Consumer business object and map each property to its corresponding attribute.
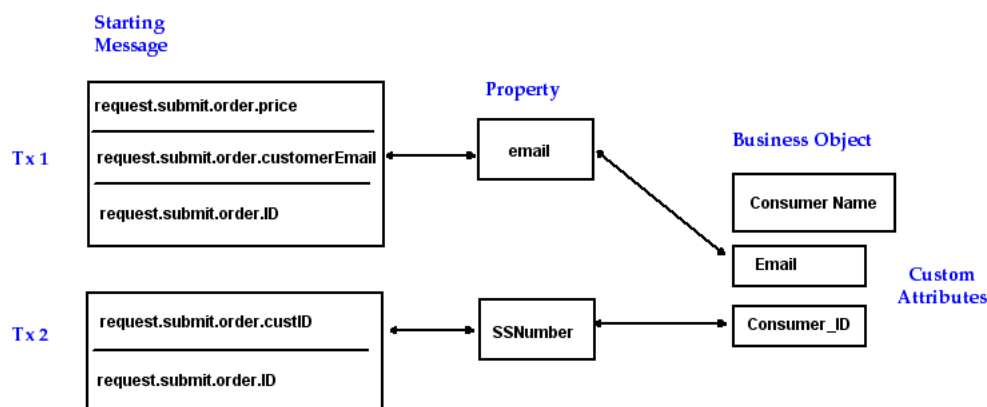
For the sake of this example, let us assume that all consumers involved in these transactions have already been created. Once traffic starts to flow, Business Transaction Management inspects the value of each of the properties shown as it observes messages. If it encounters either an email address or social security number associated with an existing consumer, it records measurements for that consumer. This makes it possible for you to see performance information for a given consumer that is an aggregate of performance across all the transactions in which the consumer participates.

In this more complicated case, setting up consumer segmentation would involve the following:

1. Add custom attributes to the consumer business object that will correspond to the properties you will create in Step 2.

2. For each transaction you want to segment by consumer: Create a property whose value is the request message element that uniquely identifies the consumer.

3. Map each property to its corresponding business object custom attribute.

4. Enable consumer segmentation in your transaction definition.

(You can do steps 2-4 all at once when you define consumer segmentation for a transaction.)

7.1.2 Creating Consumers

Business Transaction Management creates consumers in one or more of the following ways:

- If you have set up consumer segmentation, Business Transaction Management automatically creates a consumer whenever it encounters a message for a consumer that does not already exist.

- According to the information you provide when you choose **Create > Consumer** and supply values for the consumer name and attributes.

- According to the information you provide using the `importBusinessObject` command.

If you create consumers manually, as with the last two options, consumer monitoring will only start after you set up consumer segmentation.

7.1.3 Changing the Default Consumer Name

By default, discovered consumers are named according to the value of the message property to which you map them. For example, if you use a property that holds an email address, the name of the consumer displayed in the Management Console is the email address of the consumer. If the value of the property is the user's full name, that name is displayed in the Management Console. You can edit the name that will be displayed for a consumer by editing the consumer profile. If you see a combination of formats used to display user names, this might result from the fact that some were created manually (and assigned an actual name) and others were added automatically and assigned the value of the property used to map messages to consumers.

If you decide that you want to change the default consumer name, please note that you must add an attribute to the business object that corresponds to the property used for segmentation and that you must then map the property to that attribute. In the following illustration, an `Email` custom attribute is mapped to the `email` property. If you then change the default consumer name to an actual name (by editing the consumer profile), Business Transaction Management will continue to gather measurements for this consumer when it sees the message and evaluates the `email` property, but it will display the name you supply, in the main pane of the **Consumers** view and in the `Name` field of the **Profile** tab for the consumer.
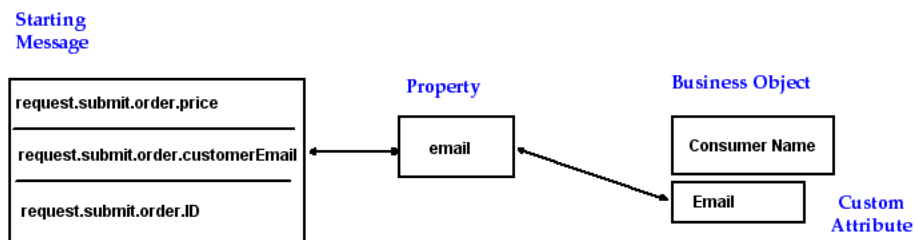
7.2 Mapping Messages to Consumers

Consumers are the default Business Transaction Management business objects that allow you to segment transaction performance based on some desired criteria. *Mapping messages to consumers* is the means by which you teach the system what criteria to use to segment performance information. The mapping process involves your creating a property that identifies the consumer and then mapping that property to a consumer business object. Please read Section 7.1, "About Consumers" for a

detailed discussion before you proceed.

Before you map messages to consumers, take a moment and decide the following:

- What properties do you need to create?

  The consumer you choose must be uniquely identified by information contained in the starting message of a transaction: for example, an ID number or an email address. This information can be contained within a message element or header.

- What message element or xpath expression will you use to define each property? (For the sake of clarity, the remainder of this procedure refers to this element or header as *Consumer_ID*.)

- Do you need to customize the consumer business object by adding custom attributes? You might need to do this if you do not want Business Transaction Management to use the default name for the consumer or if you are tracking consumers across different transactions and the consumer is represented in different ways in each transaction.

The basic steps to map messages to consumers are as follows:

1. Customize the consumer business object if needed to provide additional mapping options for your consumers.

2. When creating the property, use one of the mapping options described below to map a property to a consumer.

After you complete this mapping, enable segmentation, and send traffic, you can then view transaction performance information segmented by consumers from the **Explorer > Consumers** view, or from the **Analysis** tab of the **Transactions** view.

7.2.1 Customizing the Consumer Business Object

The consumer business object is seen throughout your environment. Each consumer created manually or automatically is an instance of the consumer business object. You can see the structure of this object in the Consumer's **Profile** tab. Section 7.1.1, "Setting up Consumer Segmentation" explains why you might need to add attributes to the consumer business object.

To add attributes to the consumer business object:

1. Select **Admin > Edit Data Model Attributes**. The **Edit Data Model Attributes** dialog box opens.

2. Scroll down to the **Business Object Custom Attributes** section and click **add attribute**.

3. Specify a **Name** and **Type** for your attribute.

4. You can optionally specify a display name for your attribute in the **Form Label** field. This name will be used to identify the attribute in the Consumer's **Profile** tab.

   The default display name is the value of the **Name** field.

5. You can optionally specify a name for querying on your attribute in the **Query Label** field. You can use this name in forming query expressions. The default query name is the value of the **Name** field.

6. Repeat Steps 2-5 to add more attributes.

7. Click **Apply**.

7.2.2 Mapping Options

You can map a message to a consumer either when you create a property or when you create a transaction.

Do the following to create a message property to hold the value of *Consumer_ID* and map the property to an attribute on the consumer object.

1. Select the starting operation of the transaction to be segmented.

2. Select **Create > Message Property on** *operation_name*.

The **Message Property** dialog box opens.

1. Specify a **Name** for your property.

2. Specify the **Property Source**: Message Content or Header, and specify *Consumer_ID*.If you specify **Header**, a dialog is opened that allows to specify the phase of the message, the header name, and the data type. If you specify **Content**, use the **Pick from Message** link to select the desired message element.

3. In the **Consumer Mapping** section, select the **Map to consumer through attribute** checkbox and choose the consumer attribute that you want to map it to, either the default **Consumer Name** attribute or an attribute that you created when you customized the consumer business object.

4. Click **OK**.

7.3 Monitoring Consumers

This topic lists the ways you can monitor consumers. The list orders these monitoring tasks from least to most detailed. Each listed task directs you to where you can perform the monitoring and provides a link to more detailed information.

In general, there are two views that offer transaction performance information segmented by consumer:

- **Explorer > Consumers view: Analysis, Alerts, SLA Compliance** tabs

- **Transactions view: Consumer Usage pane in the Analysis** tab,

If you have defined a consumer that participates in different transactions, the **Analysis** tab and the information presented in the main view for that consumer will display values that aggregate performance measurements for all transactions that involve the consumer.

Before you can monitor consumers, you must map messages to consumers and enable segmentation by consumer. See Section 7.2, "Mapping Messages to Consumers" for more information.

To get information about a consumer definition, select the consumer in the **Explorer > Consumers** view, and select the **Profile** tab. The tab shows the name of the consumer, its ID (useful in some CLI commands), any description you entered when you created the consumer, and any attributes you have created for the consumer business object.

To monitor consumers:

- Total number of consumers with SLA violations

  Navigate to **Dashboards** > **Operational Health Summary** and look for **Consumers** information.

- Transaction performance and usage summaries per consumer

Navigate to the **Analysis** tab, **Consumer Usage** panel in the **Transactions** view, and select a transaction.

- Detailed current transaction performance and usage for a specific consumer.

   1. Navigate to the **Explorer > Consumers** view.

   2. Select a consumer in the main pane.

   3. Select the **Analysis** tab.

   4. Look at **Transaction Performance** for a graphic rendition of performance over time; look at **Transaction Usage** for a grid view.

- Segmentation by consumer and by ip address

   1. Navigate to the **Explorer > Consumers** view.

   2. Select a consumer in the main pane.

   3. Select the **Analysis** tab. Look at **Breakdown by Client Address**.

- Violation alerts by consumer

   1. Navigate to the **Explorer > Consumers** view.

   2. Select a consumer in the main pane.

   3. Select the **Analysis** tab. Look at **Violation Alerts**.

- Detailed current SLA compliance for a specific consumer.

   1. Navigate to the **Explorer > Consumers** view.

   2. Select a consumer in the main pane.

   3. Select the **SLA Compliance** tab.

- Recent history of SLA violations for a specific consumer

   1. Navigate to the **Explorer > Consumers** view.

   2. Select a consumer in the main pane.

   3. Select the **Alerts** tab.

7.4 Adding and Deleting Consumers

Business Transaction Management creates consumers in one or more of the following ways:

- If you have set up consumer segmentation, Business Transaction Management automatically creates a consumer whenever it encounters a message for a consumer that does not already exist.

- According to the information you provide using the `importBusinessObjects` command.

- According to the information you provide when you choose **Create > Consumer** and supply values for the consumer name and attributes.

This section explains this last option: how you use the management console to create consumers. It also explains how you delete consumers.

To add consumers using the management console

1. Choose **Create > Consumer**.

The **Create Consumer** dialog box opens. The contents shown depend on how you have defined the consumer business object. Any custom attributes you have defined are shown in the **Attributes** area.

2. Enter a descriptive name and notes.

    The value of the **Name** field is used as the display name for the consumer in the Management Console.

    You can set up filtering on the value of the **Notes** field.

3. Supply the value of any attributes defined for the consumer business object in the **Attributes** field.

4. Click **Apply**.

To delete consumers using the management console:

1. Select the consumer you want to delete in the **Explorer > Consumers** view.

2. Choose **Modify > Delete** *consumer_name*.

    The **Delete Consumer** dialog box opens.

3. Click **Delete**.

7.5 Editing Consumer Names

As consumers are discovered and added to the Business Transaction Management sphere, they are named according to the value of the message property to which you mapped them. For example, if you mapped them to a message property that holds an email address, the name of the consumer displayed in the Management Console is the email address of the consumer.

You can edit the name of a consumer by editing the consumer profile.

You should do this only if you created a custom attribute on the Consumer object and mapped the message property to that custom attribute. Otherwise, message traffic will no longer be segmented for that consumer. See Section 7.2, "Mapping Messages to Consumers" for more information.

For example, in the following consumer profile, you could edit the **Name** field to provide the name of the consumer whose email address is listed. Whatever you enter in the name field becomes the format and content used to display the consumer in the console. But make sure that the property used for segmentation is mapped to the **User ID** attribute, which remains the email address.

Another reason you might edit the consumer profile is to add notes. You can use the value of the **Notes** field to set filtering.

To edit a consumer profile:

1. Select the consumer whose profile you want to edit from the **Explorer > Consumers** view.

2. Choose **Modify > Edit Profile for** *consumer_name*.

   The **Edit Profile for** *consumer_name* dialog box opens. Make your edits.

3. Click **Apply**.

7.6 Customizing the Consumer Label

RequiredBusiness Transaction Management application role: **btmadmin**.

You can customize the text of labels in the Management Console that by default use the terms "Consumer" or "Consumers". For example, you might want to monitor service levels and usage per department, rather than per individual. In this case, it might make sense to change the labels in the Management Console to use the term *Department* instead of *Consumer*. You should specify both the singular and plural forms of your term.

Your modified label will be used in the Navigator, in the **Analysis** pane names, in the Filter tool, and so on.

To modify the Consumer label:

1. Choose **Admin > Edit Data Model Attributes**.

   The **Edit Data Model Attributes** dialog box opens.

2. Scroll to the bottom and find the **Business Object Domains** section.

3. Type the singular form of your new label into the **Name** field.

4. Type the plural form of your new label into the **Plural Label** field.

5. Edit the **Description** field as appropriate.

6. Click **Apply**.

7. Refresh your web browser to see the effect of your change by selecting **Refresh** from the **View** menu.

8

Configuring and Viewing Alerts

Business Transaction Management alerts you when defined performance standards are violated or when conditions you specify are satisfied. This chapter explains how you define and work with service level agreements and conditions. These are powerful tools that enable you to evaluate performance at critical junctures.

This chapter contains the following sections:

- About Alerts
- About Service-Level Agreements
- Defining Service Level Agreements
- Working with SLA Policies
- Monitoring SLA compliance
- SLA Compliance Tab
- Scheduling Service or Endpoint Downtime
- About Conditions
- Working with Conditions
- Viewing Condition Alert Information

8.1 About Alerts

In addition to ongoing monitoring of the performance of services, endpoints, and transactions, you can have Business Transaction Management alert you when specific events occur.

Business Transaction Management issues several types of alerts:

- **Service level agreement alerts** (also called violation alerts and SLA alerts) are issued when a deviation occurs from the standards of performance you have defined for an endpoint, service, or transaction. These standards might be based on fixed values or historically determined baselines.

  You define standards of performance using one or more service level agreement policies that you then apply to an endpoint, service operation, or transaction.

  SLA violation alerts can be queried, filtered, and viewed on the **Alerts** tab of the related service, endpoint, operation, or transaction instance. SLA alerts in the last hour are displayed using the **Alerts in the Last Hour > SLA Alerts** view (from the Navigator). Alerts are also displayed on the operational summary dashboard.

In this case, when performance deviates from the **Warning Value** threshold set in an SLA policy, a warning alert is issued, and when it deviates from the **Failure Value** threshold set in an SLA policy, a failure alert is issued. The Warning Value is optional.

- **Condition alerts** are issued when a condition is satisfied.

  A condition is created on a transaction and is based on evaluating messages and message content per transaction instance.

  Condition alerts can be queried, filtered, and viewed on the **Alerts** tab of the related transaction instance. Alerts issued in the last 24 hours can also be filtered and viewed using the **Alerts in the Last Hour > Condition Alerts** view (from the Navigator). The dashboards include various condition alert counts.

  You might define conditions to enable fault monitoring on transactions, to test for specific property values, or to report a missing message.

- **System alerts** provide information about the health of the monitoring infrastructure. Business Transaction Management issues system alerts in situations such as the occurrence of a fatal error, a container going down, the rejection of a policy, deletion of an attribute that is in use, and so forth.

  To check for system alerts, do one of the following:

  - Select **Alerts in the Last Hour > System Alerts** in the Navigator.

  - Select **Administration > System Policies** in the Navigator and then click the **Alerts** tab.

8.2 About Service-Level Agreements

A service level agreement (SLA) policy lets you set standards of performance for your business applications and then monitor deviations from those standards. When deviations occur, an alert is issued and displayed in the Management Console.

Using SLA's you can get information about events like the following:

- When the overall fault rate for a transaction is above 1%

- When the response time for a service is 15% below its historic norm

- When the maximum response time for the consumer of a transaction is greater than one second

You can apply SLA policies to many different objects: transactions, services, endpoints, individual operations, and consumers. You can also apply one SLA policy to any number of objects in your system, for example, to all transactions, or to any subset of the objects mentioned. Not all SLA policies are appropriate to all objects.

When you define an SLA you must specify the following information

- Its type: fixed, baseline, or usage

- What objectives the performance of a given object must meet

- The period during which the SLA is enforced

- The criteria to be used in selecting the objects to which the SLA is applied

If you need to shut down an endpoint without affecting the state of your SLAs, you can schedule a downtime for the endpoint before shutting it down. Instrument measurements continue to be collected and displayed during scheduled downtimes, but these measurements are not used for SLA evaluations.

8.2.1 Performance Objectives

When you create an SLA policy, you specify some number of performance objectives, where each objective is based on one of the monitoring instruments. For example, you might specify a performance objective based on the average response time instrument. You also specify a valid range, what would trigger a warning value, and what would trigger a failure value. The Warning Value is optional.

8.2.2 Evaluation Period

Business Transaction Management evaluates the performance of your system against your SLAs on a periodic basis. When you create an SLA, you specify:

- length of the time period evaluated

- whether the period is fixed or rolling

- when the evaluation occurs

Fixed periods begin and end at defined times and the evaluation occurs only once, at the end of the period. You set the evaluation timing by specifying an exact time for the evaluation to occur, which is also the end of the period. For example, you can set the fixed period to be one month, and start the evaluation on the second day of the month at noon.

Consecutive fixed periods do not overlap.

Rolling periods begin and end relative to the evaluation timing. You set the evaluation timing by specifying an evaluation frequency. At each evaluation, the end of the period is the time of evaluation, and the beginning is the length-of-the-period earlier. For example, you can define a rolling ten minute period with evaluation occurring every two minutes.

Consecutive rolling periods *do* overlap.

8.3 Defining Service Level Agreements

You define a service level agreement (SLA) by creating an SLA policy. Depending on the type of SLA policy you choose, you might have to perform a set of tasks before applying the policy. The three types include the following:

- **Fixed Value** - This is the most basic SLA. It evaluates performance across all transaction, service, endpoint, and/or operation usage. It cannot be used for consumer usage. It does not take historical performance into account.

- **Baseline** - This SLA lets you specify performance objectives relative to historical performance measurements. It evaluates performance across all transaction, service, endpoint, and/or operation usage. It cannot be used for consumer usage.

- **Usage** - This SLA evaluates performance on a per-consumer basis. Alerts are issued only for consumers for whom performance deviates from the performance objectives, and you can monitor performance for individual consumers. This type of SLA can be used with transactions.

8.3.1 Defining a Fixed-Value SLA

1. Decide whether you want to create an SLA for a service or for a transaction:

   To create an SLA for services, choose **Create > SLAPolicy > Service SLA**, and then choose the **fixed value** type from the submenu.

To create an SLA for transactions, choose **Create > SLAPolicy > Transaction SLA**, and then choose the **fixed value** type from the submenu.

The SLA policy tool opens.

2. Specify the evaluation period.

3. Select the objectives: click **Choose Instruments**, and click the checkbox for each instrument you want to use for setting up performance objectives, and click **OK**.

4. Click **Choose Operations**, pick the operations on which you would like to set up performance objectives, and click **OK**.

The SLA policy tool should now display one performance objective for each instrument that you picked.

(In order to choose operations, you must open the tool by first selecting an object and then choosing **Create > SLAPolicy for** *selected_object*.)

5. Specify warning and failure values for your performance objectives.

If the objective is for an operation, the values pertain to only that operation. If the objective is for a service or endpoint, the values pertain to the total or average for all operations on the service or endpoint (total if the instrument is a counter, such as Traffic; average if the instrument is an averaging instrument, such as Average Response Time).

6. Use the Criteria section to choose the objects to which the policy should apply.

7. Preview the objects in the **Criteria** section.

8. Click **Apply**.

8.3.2 Defining a Baseline SLA

Before setting your baselines, you can delete previously set baselines by running `retrieveObjectData` with the `-delete` option set.

Before you can define a baseline SLA, you must do the following to set the baseline:

1. Run traffic and identify a time period that provides a good selection of baseline data.

2. Export the baseline data: Run the `retrieveObjectData` CLI command to retrieve the performance history that you want to use as a baseline for your SLA.

3. Using the output file of the `retrieveObjectData` CLI command as an input file, run the `setBaseLines` CLI command to set the baselines for your SLA.

To define a baseline SLA

1. Decide whether you want to create an SLA for a service or for a transaction:

To create an SLA for transactions, choose **Create > SLAPolicy > Transaction SLA**, and then choose the **baseline** type from the submenu.

To create an SLA for services, choose **Create > SLAPolicy > Service SLA**, and then choose the **baseline** type from the submenu.

The SLA policy tool opens.

2. Specify the evaluation period.

3. Select the objectives: click **Choose Instruments**, and click the checkbox for each instrument you want to use for setting up performance objectives, and click **OK**

The SLA policy tool should now display one performance objective for each instrument that you picked.

4. Specify warning and failure values for your performance objectives.

   If the objective is for an operation, the values pertain to only that operation. If the objective is for a service or endpoint, the values pertain to the total or average for all operations on the service or endpoint (total if the instrument is a counter, such as Traffic; average if the instrument is an averaging instrument, such as Average Response Time).

5. Use the **Criteria** section to choose the objects to which the policy should apply.

6. Preview the objects in the **Criteria** section.

7. Click **Apply**.

8.3.3 Defining a Usage SLA

Before you can define a usage SLA you must map messages to consumers and enable segmentation for the given transaction.

1. To create a usage SLA for transactions, choose **Create > SLAPolicy > Transaction SLA**, and then choose the **usage** type from the submenu.

   The SLA policy tool opens.

2. Specify which consumers the policy should apply to:

   a. By default, the policy applies to all consumers. To restrict the policy to a subset of consumers, click **all consumers in the system** and choose **a subset of consumers**.

   b. Then click the **Filter** button that appears and use the **Set Filters** dialog box to specify your subset of consumers.

3. Specify the evaluation period.

4. Select the objectives: click **Choose Instruments**, and click the checkbox for each instrument you want to use for setting up performance objectives, and click **OK**.

   The SLA policy tool should now display one performance objective for each instrument that you picked.

5. Click **Choose Operations**, pick the operations on which you would like to set up performance objectives, and click **OK**.

   In order to choose operations, you must open the tool by first selecting an object and then choosing **Create > SLAPolicy for** *selected_object.*

   If the objective is for an operation, the values pertain to only that operation. If the objective is for a service or endpoint, the values pertain to the total or average for all operations on the service or endpoint (total if the instrument is a counter, such as Traffic; average if the instrument is an averaging instrument, such as Average Response Time).

6. Specify warning and failure values for your performance objectives.

7. Use the Criteria section to choose the objects to which the policy should apply.

8. Preview the objects in the **Criteria** section.

9. Click **Apply**.

8.4 Working with SLA Policies

After you define an SLA policy, you might need to view it, modify it, disable it, duplicate it, or delete it. This section explains how you accomplish these tasks.

To View an SLA Policy Definition

1. Select **Explorer > SLA Policies** tab.

2. Select the **Profile** tab.

To View which Policies Are Applied to an Endpoint, Service, or Transaction

1. Select the object of interest in the main area.

2. Select the **Policies** tab.

To View All the Objects to which a Policy Is Applied

1. Select the policy of interest in the SLA Policies view.

2. Select the **Targets** tab.

To View the Status of a Policy

The status of a policy is either displayed in the upper right hand corner of its **Profile** tab, or it is displayed in the **Policy Status** column of the **Policies** tab if you select to view an individual target. Status values include `applied`, `pending`, `rejected`, `disabled`, or `deleted`.

To Modify an SLA Policy

1. Select **Explorer > SLA Policies**.

2. Select the policy of interest in the main area.

3. Select **Modify > Edit definition for <policy>**.

4. Make the needed changes and click **Apply**.

To Disable a Policy

You can disable policies in one of two ways; the second method is an abbreviated version of the first.

- Modify the policy definition to disable the policy: Select **Modify > Edit Definition.**Uncheck the **Enabled** check box in the policy definition (and then click **Apply**).

- Select the policy in the main pane and select **Modify > Disable** *policyName*.

To Duplicate a Policy

Duplicating policies might be useful in those cases where you have defined criteria or settings that are complex and do not want to recreate the work to define a policy that is very similar to the one you have already defined.

1. Select the policy in the main pane.

2. Choose **Duplicate Policy** from the **Modify** menu.

3. By default the policy duplicate is given the name of the original with a _1 extension. You can provide your own name if you like.

4. Modify any settings or criteria.

5. Click **Apply**.

To Delete a Policy

To delete a policy, select the policy in the main pane and choose **Delete** *<Policy>* from the **Modify** menu.

8.5 Monitoring SLA compliance

This section lists the ways you can monitor compliance of SLAs. Monitoring tasks are ordered from high-level to low-level (top to bottom). Navigation directions are given for each task.

To monitor:

- Overall current SLA compliance

  Navigate to **Dashboards** > **Operational Health Summary**

- Recent history of all SLA violation alerts. By default, data for the last hour is shown, but you can use the filter to see a longer period.

  Navigate to **Alerts in the Last Hour** > **SLA Alerts**.

- Overall current SLA compliance for a specific object (for example, a transaction named `PlaceOrder`)

  Navigate to *object type* > the **Compliance Status** icon for the specific object in the main area.

  For example, select **Transactions** in the Navigator and look at the **Compliance Status** icon for a transaction in the main area. The green circles indicate that the object is currently in compliance.

- Detailed current SLA compliance for a specific object (for example, a transaction named `PlaceOrder`)

  Navigate to *specific object* > **SLA Compliance** tab.

  For example, select **Transactions** in the Navigator, select the transaction of interest in the main area, and click the **SLA Compliance** tab.

- Recent history of SLA violation alerts for a specific object

  Navigate to *specific object* > **Alerts** tab.

  You can also use the **Analysis** tab for a particular object and look at the **Violation Alerts** panel to see violations triggered during a particular period.

8.6 SLA Compliance Tab

The **SLA Compliance** tab displays the current state of SLA compliance for the selected object, either a transaction, service, endpoint, operation, or consumer. Because some types of information are relevant to only certain types of objects, the layout of the Compliance tab can vary from object to object.

The SLA Compliance tab has two subtabs, **Threshold Compliance** and **Baselines**.

- The **Threshold Compliance** subtab provides real-time monitoring of the selected object.

- The **Baselines** subtab displays historical baseline values for the object that you can use as a reference point. Data is shown only if you have defined baselines for the selected object. (For more information, see Section 8.3.2, "Defining a Baseline SLA.")

There are no baselines for consumers. The information that is displayed on the **Threshold Compliance** subtab for other object types is displayed directly on the **SLA Compliance** tab for consumers. This information is described below.

The **Threshold Compliance** subtab uses a grid view. Each row represents one performance objective.

The columns provide various types of static information that identify and define the objectives. Also provided are two dynamic columns with real-time monitoring values. These are the **Value** and the **Current Status** columns.

The **Current Status** column can have three possible values:

- A green circle indicates that the object is in SLA compliance.

- A yellow triangle indicates that the warning threshold for the objective is currently in violation.

- A red diamond indicates that the failure threshold for the objective is currently in violation.

The **Value** column displays the current value of the instrument on which the objective is based

Click the magnifying glass next to a value to pop up a chart showing the instrument's recent history, for example:
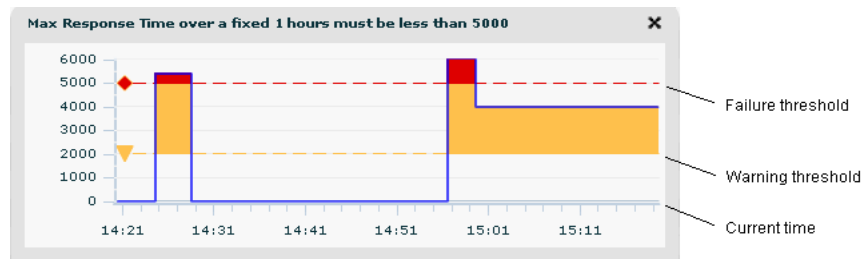
The chart displays a time period equal to the SLA's evaluation period. The right-hand edge is the current time. In this example, values above the yellow line have crossed the warning threshold. Those above the red line have crossed the failure threshold. (If you set up an objective whose value must remain above a specified threshold, then colored warning and failures areas are displayed below the threshold lines.)

Objectives are listed per consumer only if the objective is part of a usage SLA. For objectives that are not part of a usage SLA, the Consumer column is empty.

8.7 Scheduling Service or Endpoint Downtime

If you need to shut down a service without affecting the state of your SLAs, you should schedule a downtime for the service before shutting it down. Instrument measurements continue to be collected and displayed during scheduled downtimes, but these measurements are not used for SLA evaluations.

In order to schedule downtime of a service, you must first create a downtime schedule and then add the schedule to each of the service's endpoints.

This section also explains how to view a schedule, how to remove a schedule from and endpoint, and how to delete a downtime schedule from all endpoints.

To Create a Downtime Schedule

1. Choose **Create > Schedule** to open the **Create Schedule** tool.

2. Give your schedule a descriptive name.

3. Set the **Schedule Type** to **Custom**.

4. Click the first drop list and choose the recurrence frequency for the downtime. For example, the following schedule chooses a weekly downtime, specifies the day, the starting time, the duration of the down period, and the starting date.

Custom Schedule

☒ Weekly ▾ on Monday ▾ at [11]:[43] PM ▾ for a duration of [1] hours ▾ starting 09-08-2011
[add new period]

The **pick a day of month** options let you schedule by a numeric day of the month, for example, on the 1st day of each month.

The **pick a day of week** options let you schedule by the name of a day, for example, on the 1st Monday of each month.

5. Use the remaining fields and drop lists to specify the time the downtime begins, its duration, and the date of its first occurrence.

6. To schedule additional downtimes to the schedule, click **add new period**.

7. Click **Apply**.

To Add a Downtime Schedule to an Endpoint

1. Select the endpoint.

2. Choose **Modify > Set Endpoint Downtime Schedule**.

   The **Set Endpoint Downtime Schedule** tool opens.

3. Choose your downtime schedule from the **Choose schedule** drop-list.

4. Click **Apply**.

To View Scheduled Downtimes for an Endpoint

1. Select the endpoint

2. Open the **Downtimes** tab.

   The name of the schedule is shown, followed by a list of all events specified for that schedule.

To View All Schedules

1. Select **Explorer > Schedules** from the Navigator.

2. Currently defined schedules are listed in the main pane.

3. The **Profile** tab shows the definition of any schedule selected in the main pane.

To Remove a Downtime Schedule from one Endpoint

1. Select the endpoint.

2. Choose **Modify > Set Endpoint Downtime Schedule**.

   The **Set Endpoint Downtime Schedule** tool opens.

3. Choose **No Downtimes** from the **Choose schedule** drop-list.

4. Click **Apply**.

To Delete a Downtime Schedule from All Endpoints

1. In the Navigator, select **Schedules.**

2. In the summary area, select the schedule you want to delete.

3. Choose **Modify > Delete** *schedule_name*.

4. Click **Delete**.

8.8 About Conditions

You can associate one or more conditions with a transaction. A *condition* is an expression that Business Transaction Management evaluates against a transaction instance. You might use conditions to have Business Transaction Management let you know about the following kinds of events:

- An order larger than $10,000 is processed.

- A customer's fault rate is greater than 1%.

- A transaction instance has exceeded its expected completion time.

You can define three types of conditions for a transaction:

- **Content-based conditions**, which issue an alert when the criteria you specify is met. The criteria might specify a property value and/or a fault for a particular operation.

- **Missing message conditions**, which issue an alert if the message you specify is not seen by the system in the specified time period.

- **Fault monitoring conditions**, which issue an alert when a fault occurs on any operation in a transaction. These type of conditions are the equivalent of defining a content-based condition for any fault on any operation in your transaction. Fault monitoring conditions are disabled by default.

When a condition evaluates to true, Business Transaction Management does the following:

- Creates a condition alert based on the severity specified in the condition definition

- Tracks the number of occurrences of the condition

- Assembles and logs the instance of the transaction that evaluated to 'true'.

- Keeps the logged transaction instance for the length of time specified for the transaction's "Retain archived condition instances" value in the **Storage Settings** tab. By default, this is 30 days.

Conditions are evaluated on a per-instance basis; therefore, instance logging must be enabled. If you define a condition for a transaction and you have not enabled instance logging, Business Transaction Management automatically turns it on for that transaction.

When you define a condition, you also assign a *severity level* to the alert that will be issued when the condition evaluates to true. Severity levels include info, warning, and failure.

Section 8.9, "Working with Conditions" explains how you define conditions, view condition information, modify a condition, and delete a condition.

8.8.1 Defining and Evaluating Conditions

You must observe the following limitations when defining content-based conditions.

- Condition definitions can include multiple clauses, but all clauses must reference the same message (`operation.request`, `operation.reply` or `operation.fault`).

- A content-based condition can test for a property value, a fault value, or both. If both, the property must be defined on the fault phase of the operation.

- Only message elements that are referenced by means of properties can be used in a condition definition, with the exception of fault codes.

The system evaluates a content-based condition when the message for which it is defined is seen. If the condition evaluates to true, Business Transaction Management will correlate back to the transaction's start message to make sure that the instance that has satisfied the condition has been called from within the transaction. If so, the resulting instance is assembled and saved for the time duration specified on the transaction's storage settings **Retain archived condition instances**.

The system begins to evaluate a missing-message condition when it sees the starting message for the transaction. It then looks for the message for the period specified in the condition definition. If it does not see the message in that period, the missing message condition evaluates to true.

Once you have defined a condition on a transaction, it is listed in the transaction's **Conditions** tab.

8.8.2 Interpreting Condition Alerts

Alerts are commonly thought to signal that something bad has happened, for example, that an error has occurred. However Business Transaction Management uses condition alerts to signal the occurrence of any kind of message pattern. So, depending on the condition definition, an alert could signal a positive event: a customer has doubled an order, a process has attained a certain level of efficiency, sales in a particular region have exceeded the norm, and so on. You can use severity settings to signal whether the alert denote a positive event (info level) or a negative one (warning or failure level).

Once an alert is issued, you have several ways to deal with the situation. You can:

- view the logged instance

- notify interested parties by email

Condition alerts that have occurred can be queried, filtered, and viewed on the **Alerts** tab of the related transaction instance. Alerts created in the last 24 hours for all objects can also be filtered and viewed using the Condition Alerts view (from the Navigator bar). The dashboards also include various alert counts.

8.9 Working with Conditions

This section explains how you complete the following tasks in working with conditions:

- Enabling and Disabling Fault Monitoring

- Defining a Condition

- Enabling and Disabling Conditions

- Viewing Condition Information

- Modifying a Condition Definition

- Duplicating a Condition Definition

- Deleting a Condition

Section 8.10, "Viewing Condition Alert Information," provides information on viewing alert counts and alerts that are generated when a condition is met.

8.9.1 Enabling and Disabling Fault Monitoring

When you create a transaction, a fault monitoring condition is automatically created on the transaction and is disabled.

To Check Fault Monitoring Status

1. Select the transaction of interest.

2. Click the **Conditions** tab. Fault Monitoring state is shown.

3. You can click the **Edit** button to enable fault monitoring, or you can use the **Modify** menu as described next.

To Enable Fault Monitoring

1. Select the transaction for which you want to enable fault monitoring.

2. Select **Modify > Enable/Disable Fault Monitoring for** *TransactionName*. Business Transaction Management displays the Edit Fault Monitoring tool for the selected object.

3. Use the **Severity** drop down list to select the severity level you want to associate with fault-generated alerts for this object. Business Transaction Management shows the fault monitoring is enabled for the service on the **Conditions** tab for the service, and also shows the severity status for fault monitoring.

4. To disable, you can click the **Edit** button in the **Conditions** tab and deselect the enable check box in the resulting dialog. (You can also use the **Enable/Disable Monitoring** item from the **Modify** menu.

8.9.2 Defining a Condition

How you define a condition differs, depending on whether it's a content-based condition or a missing message condition. Both options are described below.

To Define a Content-Based Condition

- Select the transaction for which you want to define a condition.

- Select **Create > Transaction Condition > Content Condition for** *TransactionName*. Business Transaction Management displays the **Create Condition** tool dialog.

- Specify the name of the condition in the **Name** text box.

- Select a value for the severity you want to associate with this condition from the **Severity** drop-down list.

  Associating a severity level with a condition allows you to filter your views of alerts based on this attribute.

- Use the **Enabled** check box to enable or disable the condition.

- Optional: Use the **Description** text field to provide a description that helps you understand the nature or use of this condition without having to read through the boolean expression.

■ Use the buttons and the items in the **Criteria** area to create the boolean expression that will be used when the condition is evaluated. Three buttons are displayed from which you can select the elements that will compose your condition. (All clauses must belong to the same message (operation phase).

The **Property** button is displayed only if you have defined properties for the selected object.) The table below describes the use of the buttons. Once you click a button to select an element, a row is added to the **Criteria** used to define the condition. For each row, select an operator, and a value to test.

When you are done, click **Apply**.

Buttons	Description
New Property	If you need to define a property to base the condition on and you have not done so already, you can use the New Property button to create one.
Property	Select an existing property.
Fault	Select the operation whose status interests you, and specify whether you are looking for a particular code or any fault in the Criteria section.

To Define a Missing-Message Condition

To define a missing message condition, you must specify the message whose arrival time interests you and how much time must elapse before its absence raises an alert. The absent message can be a request, a response, or any fault.

1. Select the transaction for which you want to define a condition.

2. Select **Create > Transaction Condition > Missing Message Condition for** *TransactionName*. Business Transaction Management displays the Create condition tool dialog.

3. Specify the name of the condition in the **Name** text box.

4. Select a value for the severity you want to associate with this condition from the **Severity** drop-down list. Associating a severity level with a condition allows you to filter your views of alerts based on this attribute.

5. Use the **Enabled** check box to enable or disable the condition.

6. Optional: Use the **Description** text field to provide a description that helps you understand the nature or use of this condition without having to read through the boolean expression.

7. Use the drop down lists in the **Criteria** area to create the boolean expression that Business Transaction Management will evaluate when the condition is evaluated. Three drop down lists are displayed from which you can select the elements that will compose your condition.

Drop-down list	Description
Operation name	Select an operation from the drop down list.
Operation phase	Select request, response, or fault from the drop down list

Drop-down list	Description
Evaluation period	Select the period of evaluation: either **before the transaction's maximum duration has expired** or **Use timer**.
	If you select **Use timer**, you must specify the time period in milliseconds.
	In either case, Business Transaction Management will start timing on a per instance basis when the transaction's start message arrives.

8.9.3 Enabling and Disabling Conditions

You can enable and disable conditions individually or multiple conditions at once.

To Enable or Disable an Individual Condition

1. Select the transaction for which this condition is defined.

2. Click the **Conditions** tab in the detail area.

3. Click the Edit button for the condition you want to enable or disable.

4. Select **Modify** from the drop down list.

5. In the ensuing dialog click the Enabled/Disabled check box to change the status of the condition.

6. Click **Apply**.

To Enable or Disable Selected Conditions

1. Select the transaction whose conditions you want to enable or disable.

2. Select **Modify > Enable/Disable Conditions for TransactionName**.

3. In the ensuing dialog click the **Enable All** or **Disable All** buttons to change the status of the conditions or use the radio buttons to select the conditions of interest.

4. Click **Apply**.

8.9.4 Viewing Condition Information

1. Select the transaction of interest. from the **Transactions** view.

2. Click the **Conditions** tab to see all defined conditions. An icon indicates the severity level of the condition: green for Information, yellow for Warning, red for Failure.

8.9.5 Modifying a Condition Definition

1. Select the transaction of interest.

2. Click the **Conditions** tab in the detail area.

3. Select **Edit** from the **Edit** drop-down list for the condition.

4. Use the Condition tool to modify the definition.

8.9.6 Duplicating a Condition Definition

1. Select the transaction of interest.

2. Click the **Conditions** tab.

3. Select **Duplicate** from the **Edit** drop-down list for the condition.

4. Modify the definition if needed.

5. Click **Apply**.

8.9.7 Deleting a Condition

1. Select the transaction of interest.

2. Click the **Conditions** tab.

3. From the drop-down **Edit** menu for the condition of interest, select **Delete**.

8.10 Viewing Condition Alert Information

When a condition evaluates to true, the system tracks its occurrence and automatically creates a condition alert. This section explains the various ways Business Transaction Management displays count and condition alert information, and it explains how you can modify the amount of time that alerts are stored.

You cannot manually delete alerts.

8.10.1 Viewing Count Information

You can view count information condition alerts as follows:

- In the **Operational Health Summary** dashboard (counts for each type of alert).
- In the **Top 10 Transactions** dashboard.
- In the **Alerts** tab of a transaction.
- In the **Performance** pane of the **Analysis** tab for the transaction.
- In the **Conditions** pane of the **Analysis** tab for the transaction.

You can also edit the summary view (for a transaction) to show a count of condition alerts and the current condition alert rate by using the column chooser and selecting the **Condition alert** and **Condition alert rate** instruments. The **Condition alert** instrument specifies the condition occurrences in the specified period. The **Condition alert rate** instrument specifies the number of condition occurrences per hour in the given period.

8.10.2 Viewing Alert and Detail information

You can view condition alerts in the following ways:

- on the **Alerts** tab of the related service, endpoint, operation, or transaction instance. Detailed information for the selected alert is shown in a separate window if you click the looking glass icon or double click the alert
- in the **Condition Alerts** view from the navigator for condition alerts that occurred in the last hour. Detailed information for the selected alert is shown in the **Info** tab in the tab area.

Condition alert detail includes the following information:

- the name and severity of the condition and the expression whose evaluation triggered the alert.
- summary information: The name of the transaction, the time it was captured (Start Time), its ID, the number of message exchanges and the total messages, the

response time for the transaction, and any conditions alerts raised while the transaction executed (with severity flags shown).

■ a graph of the transaction, showing the operation for which the condition expression evaluated to true with a red broken arrow. To display the request and response message for each operation, click the inspector icon for the desired operation in the list below the graph.

8.10.3 Modifying Alert Expiration Time

By default logged information for alerts expires after 1 year (31536000 seconds).

You can change the alert expiration time as follows:

1. Select **Administration > System Services** from the Navigator.

2. Select the **AP_Notifier_Service** in the summary pane.

3. Select **Edit Setup Data for AP_Notifier_Service** from the **Admin** menu.

4. Modify the **Logged notification expiration time** field to the desired number (of seconds).

5. Click **Apply**.

9

Creating Reports

This chapter explains how you create reports based on the performance information gathered by Business Transaction Management.

It contains the following sections:

- Overview
- Generating Reports

9.1 Overview

Business Transaction Management provides the following elements to help you create reports:

- the `retrieveObjectData` command, which produces the data that you want to use in your report. You specify the data you are interested in using command parameters. For example, you could extract data about fault count by service, maximum response time by transaction, condition alert rate by transaction, and so on.

- templates that you can use with the Oracle BI Publisher (11 g) application to create well-formatted reports in HTML, PDF, RTF, Excel, PowerPoint, MHTML, and Data formats.

- sample `retrieveObjectData` commands that you can use for the type of report you want to generate.

- sample `retrieveObjectData` output files that you can use to experiment with your reporting tool without having to generate data.

Material to help you create reports is found in the following directory:

BTM_Install_Directory\samples\reports

The directory includes two sub-directories:

- **templates**: this directory includes templates you can use only with the template tools provided by the Oracle BI Publisher (11 g) application. It also includes the `reports.xls` file, an Excel spreadsheet that shows, for each template, the appropriate `retrieveObjectData` command that will provide the output data to be combined with the template to generate the report.

  For example, the `retrieveObjectData` command to be used with the `AvgResponseTimeByService` template is

  ```
  btmcli retrieveObjectData -s sphere -l sphereLogin -type service
       -attributeNames  friendlyName averageResponseTime -inTheLast 24h
       -intervalSize 1h -format xmltable > AvgResponseTimeByService.xml
  ```

Or, the `retrieveObjectData` command to be used with the `FaultCountByService` template is

```
btmcli retrieveObjectData -s sphere -l sphereLogin -type service
    -attributeNames friendlyName faultCount -inTheLast 24h -intervalSize 1h
    -format xmltable > FaultCountByService.xml
```

- **sampleData**: this directory contains sample `retrieveObjectData` output files that you can use to see what reports might look like without having to generate data.

9.2 Generating Reports

Using the sample templates and command lines for reference, you can learn to create the output data and the templates of your choice to generate reports. You might also need to use the `listInstruments` command to determine what attribute names and segment names to use for the `retrieveObjectData` command.

For information on how to use Business Transaction Management templates when working with BI Publisher, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

10

Commands and Scripts

This chapter introduces the Business Transaction Management command line interface (CLI), which you can use to execute commands as well as scripts. It explains how you use the CLI, how you get help and version information, how you can execute commands securely, and it describes the system variables you can set to use with commands. It includes the following sections:

- About the Command Line Interface (CLI)
- Getting Help and Version Information
- Executing CLI Commands and Scripts
- Security Options in Accessing CLI Commands
- Command Summary

  Environment Variables
- Attribute Names

In addition to commands, Business Transaction Management also provides several utilities. These are described in Chapter 11, "Utilities."

10.1 About the Command Line Interface (CLI)

The Business Transaction Management Command Line Interface (CLI) allows you to issue management commands from the command line or through the use of scripts (in bash, perl, ruby, DOS). You can use the CLI from the Unix and DOS command lines, and any other environments that work with your Business Transaction Management installation.

CLI commands allow you to configure the system, manage DNS aliases, resolve replication issues, register services, manage metadata, control monitoring, apply policies, migrate data, or generate data needed for reporting.

The CLI executable (btmcli.bat on Windows and btmcli.sh on Unix-like systems) is located in the *BTM_Install_Directory*/tools directory. On Windows platforms, the CLI uses the JRE bundled with Business Transaction Management. On Unix-like systems, you must specify a JRE to use by doing one of the following:

- set the JAVA_HOME environment variable and remove the *BTMi_nstall_dir/_server*/jre directory
- replace *BTM_install_dir*/server/jre with a link to a valid JRE directory

There are several command line utilities for performing additional actions. See Section 11.1, "About Command Line Utilities" for more information.

10.1.1 Getting Help and Version Information

To get a list of all CLI commands with a brief help summary, enter the following command:

```
C: > btmcli help
```

To get detailed help for an individual command, enter a command like the following:

```
C: > btmcli help commandName
```

To get version information, use the following command:

```
C: > btmcli -v
```

10.1.2 Executing CLI Commands and Scripts

You can execute a Business Transaction Management CLI command or script in any command shell using the following syntax:

btmcli *commandName commandFlags*

To execute a single command, specify the command and its parameters at the command prompt. For example:

```
C: btmcli encryptPassword -password "abracadabra"
```

To execute a script, specify the name of the script at the command prompt. For example:

```
C: > myBTMScript.cmd
```

- The script can contain any Business Transaction Management CLI commands.

- The name of your script file should have an extension that is appropriate for your operating system (.cmd, .bat, .sh).

- Script comments should be formatted as appropriate for your operating system.

- The script can make use of the environment variables described in Section 10.1.5, "Environment Variables."

10.1.3 Security Options in Accessing CLI Commands

Nearly all CLI commands require that you provide a credential in the form of `username:password` to execute the command.

You can provide this credential in one of two ways:

- Using the `-l` flag to the command.

- Setting the `AP_USER_LOGIN` environment variable to the value of the credential. In this case, you do not have to specify the `-l` flag.

Whether you use the -l flag or the environment variable, you can specify the credential in one of two ways:

- by providing the user name and password directly when you enter the command, or by setting the `AP_USER_LOGIN` environment variable to that *username:password* value.

- by creating a `username:password` credential in a credential store, and passing the credential name to the command or by setting the `AP_USER_LOGIN` environment variable to the value of that credential name.

The following examples use the btmcli `configure` command to illustrate each case:

1. In the following command, the -l flag is used to specify a user name and password.

```
btmcli configure -i myConfigureFile -s http://uitst3:8080/btmcentral/sphere/
-l joanna:abracadabra
```

2. In the following command, no `-l` flag is specified because the `AP_USER_LOGIN` environment variable has been set to `joanna:abracadabra`.

```
btmcli configure -i myConfigureFile -s http://uitst3:8080/btmcentral/sphere
```

3. In the following command, the credential `JoannaCred` has been defined by the `credStoreTool` command as follows:

```
btmcli credStoreTool -createCred JoannaCred -credType up
-credValue joanna:abracadabra
```

The `configure` command will then look like this:

```
btmcli configure -i myConfigureFile -s http://uitst3:8080/btmcentral/sphere/
-l JoannaCred
```

4. In the following command, the credential `JoannaCred` has been defined by the `credStoreTool` command as follows:

```
btmcli credStoreTool -createCred JoannaCred -credType up
-credValue joanna:abracadabra
```

The `AP_USER_LOGIN` environment variable has then been set to `JoannaCred=joanna:abracadabra`. The `configure` command will then look like this:

```
btmcli configure -i myConfigureFile -s http://uitst3:8080/btmcentral/sphere/
```

10.1.4 Command Summary

The following table lists CLI commands in alphabetical order and provides a summary of the command action.

Command Name	Description
addBaseAddressAlias	Add the specified alias for the entry point with the given base address. If another entry point with this alias as its base address is already known to the sphere, the two entry points and their contents are merged, and duplicate WSDLs and endpoints are removed.
addNodeAlias	Add network aliases to an already registered node.
addPathAlias	Use the addPathAlias command to add an alias representing an alternative URL path that can be used to access the specified WSDL or endpoint. Specifying this alias can help prevent duplicate artifacts from being created during discovery.
configure	Read a configuration xml file and use the information to configure the installation associated with the given sphere URL.
configureAlivenessCheck	Specify the preferred method for the sphere to check the aliveness of endpoints, rather than using the default ping.

Command Name	Description
createOrUpdateGenre	Specifies the name of a new service or endpoint genre, and identifies a custom icon to be used in representing the new genre.
createOrUpdatePolicy	Create or update the specified policy directly in the sphere.
createSettingsDocument	Use the `createSettingsDocument` command to create a settings document for input to the createOrUpdatePolicy command.
credStoreTool	Creates, obtains, or deletes the specified type of credential from the credential store.
deleteAll	Deletes discovered objects along with related artifacts such as transactions, properties, registered services, and containers. Also deletes historical data associated with deleted items.
encryptPassword	Convert the specified text string using the Business Transaction Management encryption engine. The resulting cipher text appears in the command output.
exportBusinessObjects	Create a settings document for input to the `createOrUpdatePolicy` command.
exportMessages	Export messages for an endpoint for a specified period of time. Exporting messages can be useful in testing and debugging. You have the option of exporting messages to a file; whether you do or not, output is always sent to the Console.
exportPolicies	Export the specified policies either to stdout, or to the named output file. Policies can be exported by name, type, or by means of a query that defines a set of policies.
exportPolicyTemplates	Export the specified policy templates either to stdout, or to the named output file. Policies templates can be exported by name, type, or by means of a query that defines a set of policies.
exportProfile	Export the profile values for the specified object(s) either to stdout, or to the named output file. Profiles can be exported by name or id.
exportSchedules	Export schedules from the target sphere. You can output the schedules to a file or to stdout. Schedules can be exported by name or by means of a query that defines a set of policies.
exportTransactionDefns	Export the specified transaction definitions. If you do not specify an output file, the definition is sent to stdout.
getSetupData	Write the setup file for the specified service to standard output. You can use this setup file as input for the putSetupData command.
importBusinessObjects	Import one or more of the specified business objects described by the specified XML file into the target sphere. By default, this command replaces all existing objects that are older than the imported objects.
importPolicies	Import the policies contained in a previously created export file. The policies to be imported will be taken from the file specified by the -i parameter, or from stdin, if no such file is specified.
importPolicyTemplates	Import the policy templates contained in a previously created export file. The policy templates to be imported will be taken from the file provided by the -i parameter or from stdin if no such file is specified.
importProfile	Import a previously exported object profile into the system. Note that importing a profile does not cause new objects to be created, it only updates the profile attributes of existing objects.

Command Name	Description
importSchedules	Import previously exported schedules to the target sphere. By default, this command replaces all existing schedules that are older than imported schedules.
importTransactionDefns	Import previously exported transaction definitions to the specified sphere. By default, this command replaces all existing definitions that are older than imported definitions.
listInstruments	List the instruments defined by the policy templates in the system, as well as the attribute names and segments for each. You can use this information as arguments for the `retrieveObjectData` command.
listNodeAliases	List all the aliases for a single network node, or for all known network nodes.
mergeServices	Merge two versions of the specified service. This command removes the source version and moves its endpoints to the target version. After the merge, the target version contains all the endpoints from both versions. If the source version has any profile attributes or message properties that you want to retain for the merged versions, you must recreate these on the target version.
moveEndpoints	Move one or more endpoints from the source version to the target version of the specified service.
moveMeasurements	Move the measurements collected for a service or endpoint that has been deleted from the sphere to another endpoint or service that is its logical successor.
putSetupData	Read the setup file from a named file or from standard input and save it as the setup data for the specified service.
registerDevice	Register a management device with the sphere. Currently, hardware and software load balancers are the only supported device type, and the F5 BigIP is the only fully-supported load-balancer. If you register multiple devices, you can use the `setDefaultLoadBalancer` command to set the default device.
registerExternalContainer	Group the endpoints of one or more entry points (not part of an observed container) into their own external container. This is only necessary for manually-registered endpoints.
registerMonitor	Register a monitor agent with Business Transaction Management.
removeBaseAddressAlias	Remove the specified alias to the entry point with the given base address. The alias must be defined locally for this entry point only. Aliases defined at the network node level must be removed using the `removeNodeAlias` command.
removeDuplicateEndpoint	Remove the specified duplicate endpoint. Typically, you would use this command when the system discovers two endpoints with the same path but different host names due to a DNS alias that was not known to the system. Executing this command also adds the missing alias to make sure that future discoveries do not result in recreating the duplicate endpoint.
removeNode	Remove a network node from the sphere.
removeNodeAlias	Remove aliases from a registered node.

Command Name	Description
removePathAlias	Remove an alias representing an alternative URL path that can be used to access the specified WSDL or endpoint. Any future registration or discovery will again treat the two paths as distinct.
removePolicy	Remove an applied policy and optionally wait for the removal of the policy to complete.
removePolicyTemplate	Removes the specified policy template from the system.
renameEndpoint	Change the specified endpoint's friendly name in the sphere.
resetSphereUrl	Make sure the sphere URL for container services currently registered with this sphere matches the current sphere URL. You might need to use this command if the base address of the sphere changes.
retrieveObjectData	Retrieve profile and measurement data for one or more objects. You can use this data as input to a reporting tool.
sendEventNotification	Create and send an event notification using the Notifier Service. How the notification is handled is determined by the currently active subscriptions.
sendToNotifier	Send commands to the Notifier Service for processing. The commands are specified in the XML input file. The command returns the response document returned by the Notifier Service.
setBaselines	Set baselines for managed objects. The input document format is the same as the output of the `retrieveObjectData` command.
setDefaultLoadBalancer	Set the default device used to model routing entry points discovered from observed message traffic. In a simple environment with only one known load-balancer, that device is automatically used as the default. If you register additional devices, use this command to set which device should be used to model routing entry points.
showService	Show the structure of the service specified by the friendly name, qualified name, or URL of any endpoint for this service. You can also specify a single version of the service (-version) or the display of more detailed information (-verbose).
unmonitorEndpoint	Stop monitoring the specified endpoint with the monitor agent where it is registered. Once monitoring stops, no performance measurements are recorded, no messages are logged, and no transactions are traced.
unregister	Unregister the service, WSDL, or endpoint from the sphere.
unregisterContainer	Unregisters the specified container.
unregisterMonitor	Unregister the specified monitor agent from the sphere. Use this command only when the monitor has been taken offline permanently. Monitors must be offline long enough for the system to mark them down before you can unregister them.
updateProfileData	Update profile attribute data for the specified set of objects. This command only updates existing objects; if it cannot find the specified objects, it returns an error.

10.1.5 Environment Variables

The following table describes environment variables that you can define for use with the Business Transaction Management commands in the place of commonly used command flags.

Environment Variable	Flag Name Equivalen	Flag Long Name Equivalent	Description
AP_SPHERE_URL	-s	-sphereUrl	The URL for the sphere. `http://hostname:port/apcentral/sphere/` .NET example: `http://host:port/apcentral/sphere/soa.asmx`
AP_USER_LOGIN	-l	-userLogin	The log-in credentials for the sphere, in one of the following formats: `username:password` credName=myCredentialName Typically these would be the credentials of a user with administrative privileges. See Section 12.3.1, "Authentication and Role Mapping" for more information. You can encrypt passwords for use with the CLI using the `encryptPassword` command.

10.1.6 Attribute Names

A number of CLI commands require that you pass an attribute name as a flag value. This section lists valid attribute names for different object types.

Endpoint Attributes

Name	Name	Name
aliases	friendlyName	operationsNote
binding	gateway	qualifiedName
businessContact	genericNote	relativeLocation
businessNote	genre	service
contact	interface	serviceQName
container	isAlive	relativeLocation
defaultCredentials	lastIsAliveChangeDate	service
dependsOnEndpoint	lastManagementChangeDate	serviceQName
deployment	lastManagementChangeIdentity	serviceDescriptionURL
description	lastUpdateIdentity	supportContact
endpointAppProtocol	lastUpdateMarker	supportNote
endpointSoftwareType	location	targetEndpoint
endpointSource	managingIntermediaries	technicalContact
endpointTransportProtocol	note	technicalNote
endpointType	operations	testingNote
firstRegistryDate	operationsContact	uddiBusinessKey
--	--	uddiKey

Operation Attributes

Name
dependsOn
operationName
operationType
propertyOverrides

Interface Attributes

Name	Name	Name
businessContact	name	supportNote
businessNote	namespace	technicalContact
classifications	note	technicalNote
contact	operations	testingNote
description	operationsContact	uddiBusinessKey
friendlyName	operationsNote	uddiKey
genericNote	owningEndpoint	--
lastUpdateIdentity	sourceType	--
lastUpdateMarker	supportContact	--

Interface Operation Attributes

Name
classification
documentation
operationName
operationType
properties
sampleMessageNotes

Service Attributes

Name	Name	Name
alive	lastIsAliveChangeDate	serviceSource
applicationGroup	lastManagementChangeDate	serviceType
businessContact	lastManagementChangeIdentity	supportContact
businessNote	lastUpdateIdentity	supportNote
category	lastUpdateMarker	technicalContact
contact	managedDescriptionURL	technicalNote
deploymentNames	note	testingContact

Name	Name	Name
description	operations	testingNote
documentation	operationsContact	uddiBusinessKey
endpoints	operationsNote	uddiKey
firstRegistryDate	organization	version
friendlyName	originalDescriptionURL	--
lastDeployDate	primaryDeploymentName	--
lastDeployIdentity	qualifiedName	--

Transaction Attributes

The only transaction attribute is `friendlyName`.

Container Attributes

Name	Name	Name
administratorUI	lastIsAliveChangeDate	phaseInLifeCycle
baseAddress	lastDeployIdentity	supportContact
businessContact	lastDiscoveryDate	supportNote
businessNote	lastManagementChangeDate	technicalContact
connectionLimit	lastManagementChangeIdentity	technicalNote
contact	lastUpdateIdentity	testingContact
containerInfo	lastUpdateMarker	testingNote
firstRegistryDate	note	uddiBusinessKey
friendlyName	operationsContact	uddiKey
genericNote	operationsNote	--
hostName	osName	--
implementationType	osVersion	--
ipAddress	owner	--
isAlive	performanceRating	--

Deployment Attributes

Name	Name	Name
businessContact	isAlive	supportContact
businessNote	lastDeploymentDate	supportNote
contact	lastDeploymentIdentity	technicalContact
container	lastIsAliveChangeDate	technicalNote
deploymentCategory	lastUpdateIdentity	testingContact
deploymentStatus	lastUpdateMarker	testingNote
deploymentType	note	uddiBusinessKey

Name	Name	Name
description	operationsContact	uddiKey
firstRegistryDate	operationsNote	--
friendlyName	pluginAgentPossible	--
genericNote	pluginAgentPossibleText	--

Contact Identity Attributes

Name
emailAddress
informationURL
name
phoneNumber
usage

Notes Type Attributes

Includes two attribute names: `theNote` and `usage`.

Description Attributes

Includes two attribute names: `contacts` and `notes`.

10.2 addBaseAddressAlias

Use the `addBaseAddressAlias` command to add the specified alias for the entry point with the given base address.

If another entry point with this alias as its base address is already known to the sphere, the two entry points and their contents are merged and duplicate WSDLs and endpoints removed.

The `addBaseAddressAlias` command accomplishes the same thing as the command `removeDuplicateEndpoint` when that command is given two endpoints that differ only in their base address.

Using the `-force` option creates an entry point and its alias before any endpoints corresponding to that entry point are discovered or registered, thus preventing duplicates from being (mistakenly) created in the first place.

Unlike the aliases created by the `addNodeAlias` command, aliases created by the `addBaseAddressAlias` command or the `removeDuplicateEndpoint` command apply to a single entry point. If the same alias is added at the network node level, either automatically by message observation or discovery, or manually by the `addNodeAlias` command, it will supercede the locally-defined alias for the entry point, and it will not be possible to delete that alias using the `removeBaseAddressAlias` command.

You can preview the results of this command before committing to the changes it defines. After previewing, you must run the command again without the `-P` option to actually make the changes.

Use the `removeBaseAddressAlias` command to remove an alias created with the `addBaseAddressAlias` command.

Section 12.7, "Resolving Discovery Issues" describes the use of this command in resolving discovery issues.

Command Syntax

```
btmcli addBaseAddressAlias -baseAddress address -alias address [-P] [-f]
                          -s sphereURL -l username:password
```

Name	Long Name	Description
-baseAddress	--	The base address of an existing entry point in the sphere. Use the format *scheme*://*host*:*port*/
-alias	--	The base address to use as an alias for the above entry point.
-P	-preview	Preview the result of the command, but do not actually make any changes to the model.
-f	-force	Create an entry point with the given base address if it does not yet exist. Use this option to prime the system with aliases before duplicate endpoints are discovered.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the encryptPassword command.

Examples

The following example shows the addBaseAddressAlias command.

```
btmcli addBaseAddressAlias -baseAddress http://jbujes:8080
        -alias http://1.2.3.4:8080 -s http://localhost:8080/btmcentral/sphere/
        -l admin:abracadabra
```

10.3 addNodeAlias

Use the addNodeAlias command to add the specified alias to the network node identified by the canonical name. If the node does not exist, the alias is not added unless you also specify the -force option, in which case the node is created as well.

Use the removeNodeAlias command to delete an alias created with the addNodeAlias command.

See Section 12.7, "Resolving Discovery Issues" for a discussion of the use of this command in resolving discovery issues.

Command Syntax

```
btmcli addNodeAlias -n nodeName -alias aliasName [-f]
                    -s sphereURL -l username:password
```

Name	Long Name	Description
-n	-name	The canonical name of the network node. Typically a fully-qualified domain name. For example, jbujes-myPC.edgility.com
-alias	-alias	An alias for the network node. This can be a domain name or a static IP address.
-f	--	Force the alias to be created (as well as its canonical network node) even if the network node is unknown.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the encryptPassword command.

Examples

The following example shows the addNodeAlias command.

```
btmcli addNodeAlias -n jbujesPC.edgility.com -alias Athena
      -s http://localhost:8080/btmcentral/sphere/ -l admin:abracadabra
```

10.4 addPathAlias

Use the addPathAlias command to add an alias representing an alternative URL path that can be used to access the specified WSDL or endpoint. Specifying this alias can help prevent duplicate artifacts from being created during discovery.

Normally, path aliases are added automatically in one of two ways:

- The register command detects an HTTP redirect and adds the original URL as an alias to the service descriptor created with the redirected URL.

- The user uses the removeDuplicateEndpoint command to remove an unwanted duplicate created due to message traffic being observed at a non-canonical URL for an existing endpoint.

In the second case, adding the alias when the original endpoint is registered or discovered will prevent the duplicate from ever being created. So, you might add the addPathAlias command to a script used to recreate an environment that was observed in testing to have this problem.

Use the removePathAlias command to delete an alias you created with the addPathAlias command.

See also addNodeAlias for host name aliases and addBaseAddressAlias for other aliases in the authority part of the WSDL's URL.

See the section "Resolving Discovery Issues" for a discussion of the use of this command.

Command Syntax

```
btmcli addPathAlias -s wsdlUrl | -e endpointUrl -alias pathAlias
                    -s sphereUrl -l username:password
```

Name	Long Name	Description
-w	-wsdlUrl	The full URL of the WSDL (service descriptor) for which you want to add an alias.
-e	-endpointUrl	The full URL of the endpoint(s) for which to add the alias. If multiple endpoints are matched, they will all be given the alias.
-alias	--	The alternate path to map to the same descriptor or endpoint. This should be just the path portion of the URL, starting with "/".
-s	-sphereUrl	The URL of the sphere.
		http://*hostname*:*port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `addPathAlias`command.

```
btmcli addPathAlias -e http://jbujes:7001/bookmart/credit/CreditService
        -alias /bookmart/amex/CreditService
        -s http://jbujes:8080/btmcetnral/sphere/ -l admin:abracadabra
```

10.5 configure

Use the `configure` command to configure Business Transaction Management as an alternative to using the browser-based Configuration Wizard for initial configuration. You use the `configure` command in conjunction with an input file that contains specifications for various Business Transaction Management settings.

Before executing the `configure` command, you must decide whether you want to store information in an embedded database, or in an Oracle database. If you want to store information in an Oracle database, you will need to install and set up the database before running the `configure` command.

The sections following the syntax description provide additional information about the configuration process and the input file.

Command Syntax

```
btmcli configure -i inputFile -sphereUrl -l username:password
```

Name	Long Name	Description
-i	-inputFile	The path to the configuration file to use to configure a Business Transaction Management installation.
-s	-sphereUrl	The URL of the sphere you want to configure. http://hostname:port/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `configure` command.

```
btmcli configure -i myConfigFile.xml -s http://localhost:8080/btmcentral/sphere/
                 -l admin:abracadabra
```

10.5.1 How the configure Command Works

The `configure` command checks every Business Transaction Management service for its current configuration state. If all services are in a configured state, the command exits and informs you that Business Transaction Management is configured. If there is a mix of configured and unconfigured services, the `configure` command attempts to configure the unconfigured services. If an error or exception occurs, Business Transaction Management logs the error and the configuration continues. When the `configure` command completes, it returns a message that describes all successful and unsuccessful service configurations.

If all Business Transaction Management services are configured successfully, the command returns a "configured" message.

You can run the `configure` command again following a configuration attempt where an error occurs, and the system will attempt to configure the service(s) that returned an error during the previous configuration attempt. The `configure` command does not try to reconfigure a service that reports a successful configuration.

If you would like to reconfigure a service that has been configured successfully, you can run the `getSetupData` and `putSetupData` commands.

Any reconfiguration of Business Transaction Management services requires a restart of the server.

10.5.2 The Configuration File

When you install and configure Business Transaction Management using the browser-based configuration wizard, a configuration file called `essentialConfiguration.xml` is created. (See Section 12.4, "Backing up and Restoring Business Transaction Management," for information on its location on different platforms.) You can use this file as a starting point to initialize a new Business Transaction Management system; you will need to change the contents of the file to specify your desired settings. This then becomes the input file to the `configure`

command. The preference names in the configuration file, and their values, are both case sensitive.

Unlike the Configuration Wizard, the `configure` command does not allow you to stop midway through the configuration and then resume configuration at a later time. The `configure` command requires a complete configuration file -- if required parameters are missing from the file, the `configure` command returns an error.

Password Encryption

To supply an encrypted password for the configuration file, run the `encryptPassword` command and then copy and paste the resulting value into the configuration file.

If you use a clear text password in the configuration file, Business Transaction Management encrypts the password when it stores the configuration information in the setup data of the appropriate system services.

Configuration File Parameters

The following table lists the parameters to use in the configure command configuration file.

Alias Preferences

You can specify alias values for the nodes running Business Transaction Management central services: sphere, performance, and transaction monitoring. In each case the preference value is a comma-separated list of alternative DNS names/IP addresses that can be used to access services on the respective server.

For example, if the sphere URL is `http://server1:8080/apcentral/sphere/`, but the machine `server1` also has the DNS alias `spherehost`, you can set `sphereAliases` to `spherehost` to reflect that. If the machine's primary IP address is 10.10.22.44, but the sphere's container is listening on all lP addresses and the machine has a second network interface running on 10.10.22.47, you could set `sphereAliases` to `spherehost`, 10.10.22.47 to reflect both these aliases.

If all Business Transaction Management central services are running in the same container (the demo case), the `performanceAliases` and `transactionAliases` are ignored. If the servers are running in different containers but on the same machine as the sphere, the aliases are applied to the same node definition. So, in effect, the performance and transaction aliases are added to those for the sphere, rather than tracked separately.

Note: All parameters are `type=string`, except for the `acceptLicenseTerms` parameter, which is `type=boolean`.

Parameter Name	Required	Description
sphereUrl	YES	URL of the sphere you are configuring. The sphere is deployed in btmMain.
		The value should be the same value provided in the -s flag for the configure command.
		http://hostname:port/btmcentral/sphere/

Parameter Name	Required	Description
databaseChoice	YES	Indicates the type of database to use for the Business Transaction Management databases. Use one of the following: ■ embedded (Java platforms only) ■ oracle If you specify embedded, ignore all subsequent database settings.
nodeAliases	No	Aliases for the node running the sphere. See discussion of Alias preferences, above.
performance.nodeAliases	No	Aliases for the node running the performance server (btmperformance.war). See discussion of Alias preferences, above.
transaction.nodeAliases	No	Aliases for the node running the transaction server (btmexmrt.war). See discussion of Alias preferences, above.
deploymentUsername, deploymentPassword	YES	Some Business Transaction Management services communicate with the application server to perform discovery; these services require a set of administrative credentials mapped to an administrative user for the application server This is required for all application servers except Tomcat
agentServiceDB_ databaseUrl, agentServiceDB_provider, agentServiceDB_ userName, agentServiceDB_password	NO	**Agent message log database. By default, all agents in a container log to this database. You can override the behavior after initial configuration. See** Section 12.2.2, "Setting up the Message Log Database." This is required if databaseChoice is Embedded.
sphereServiceDB_ databaseUrl, sphereServiceDB_provider, sphereServiceDB_ userName, sphereServiceDB_ password	NO	Sphere database settings. Required if databaseChoice != Embedded
exmServiceDB_ databaseUrl, exmServiceDB_provider, exmServiceDB_userName, exmServiceDB_password	Required if databaseChoice != Embedded	Exception management component database settings. Required if databaseChoice != Embedded
performanceDB_ databaseUrl, performanceDB_provider, performanceDB_ userName, performanceDB_password	Required if databaseChoice != Embedded	Performance component (SLM) database settings. Required if databaseChoice != Embedded

Parameter Name	Required	Description
performanceUrl	YES	The URL to use to contact the performance (SLM) component of Business Transaction Management (this component is deployed in btmPerformanceServer). The performance component must be running at the time of configuration. btmPerformanceServer is typically deployed in a separate container than the sphere. When you specify an performanceURL on a different node that the sphere, Business Transaction Management also registers the remote container with the sphere.
transactionURL	NO	The URL to use to contact the transaction component of Business Transaction Management (this component is deployed in btmTransactionServer). The transaction component must be running at the time of configuration. btmTransactionServer is typically deployed in a separate container than the sphere. When you specify an transactionURL on a different node that the sphere, Business Transaction Management also registers the remote container with the sphere.

Parameter Name	Required	Description
base_locationType, base_directory, base_maxDuration, base_maxSize, base_numFiles	Optional	By default, trace log files are stored in the directory BTMStorageRoot/deployment_name/system_service/logdir BTMStorageRoot varies with the application server; for more information, see Section 12.4, "Backing up and Restoring Business Transaction Management." If you wish to store trace log files in a different location, use the parameters described below. Note: These parameters will not affect the trace log file settings for the following system services: aqms, notifier, and domain. ■ base_locationType - This required parameter must have a value of "default" or "custom"; if the value is "custom", then the base_directory parameter must have a value. ■ base_directory - The directory where Business Transaction Management should create log files. This parameter must have a value if base_locationType = custom. Do not use this preference if base_locationType = default. Note: If you change the directory location using the above parameters, you are required to also include the base_maxDuration, base_maxSize and base_numFiles parameter settings. ■ base_maxDuration - log file age in hours. Once a log file reaches this age, it rotates. (When you configure Business Transaction Management using the Management Console, the default setting is 168.) ■ base_maxSize - size in kilobytes for all log files together. Once the log files become larger than this setting, log file rotation occurs. (When you configure Business Transaction Management using the Management Console, the default setting is 50000.) ■ base_numFiles - number of concurrent log files that can exist. (When you configure Business Transaction Management using the Management Console, the default setting is 7.) base_maxSize / base_numFiles gives the maximum size for each log file. When this size is reached for a particular file, it rotates.

10.6 configureAlivenessCheck

Use the `configureAlivenessCheck` command to specify whether the sphere checks the aliveness of containers or endpoints and the preferred method for doing so.

Command Syntax

```
btmcli configureAlivenessCheck -baseAddress baseAddress | -e epUrl1 [[epUrl2]...]
                               -method standard | skip | getwsdl |geturl | ping
                               [-alternateUrl alternateUrl]
                               [-endpointDreds username:password]
                               [-pingHostPort host:port]
                               -s sphereUrl -l username:password
```

Name	Long Name	Description
-baseAddress	--	The base address for which to configure aliveness check. The effect of this flag depends on the setting of the -method flag:
		If -method is set to standard, geturl, or ping, the base address is tested only once and all endpoints under this base address inherit its aliveness value.
		If -method is set to getwsdl, each endpoint with the specified base address has its WSDL fetched.
		If -method is set to skip, no aliveness checking is performed for this base address, but endpoints with this base address might be checked individually if specified with the -e or -alternateUrl flags.
-e	-endpointUrl	One or more endpoint URLs for which to configure aliveness checking. All endpoints with the given URLs are configured identically.
-method	--	The method to be used for aliveness checking. Choose one of the following: standard: open and close a socket to the given host:port
		skip: do not check aliveness
		getwsdl: do an HTTP GET on the endpoint's WSDL
		geturl: do an HTTP GET on the given url
		ping: do a TCP ping on the host and port specified with the -pingHostPort flag.
		See below for more details.
-alternateUrl	--	The URL to use in place of the base address or endpoint URL. This flag is required when the geturl method is used.
-endpointCreds	--	Authentication credentials (username:password) to use when fetching the alternate URL for an endpoint. It is ignored for any other configuration.
-pingHostPort	--	The host and port to ping in place of the actual base address of the endpoint URL. This flag is required only when the method ping is specified.
-s	-sphereUrl	The URL of the sphere.
		http://*hostname*:*port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on the various options available for furnishing login credentials.
		You can encrypt passwords using the encryptPassword command.

Choose the method that best suits the endpoint you are monitoring. For example, for WebLogic JAX_RPC or JAX-WS services, fetching the generated WSDL will succeed if the container is up and the service is deployed and running, so it is as good an aliveness check as actually calling the service without the inconvenience of having a sample message to send that is both valid and harmless. The advantage of the getwsdl

method compared to the arbitrary http fetch (`geturl`) is that you can specify it at the base address level and get the effect of configuring the aliveness check for every endpoint in that base address to fetch its own wsdl without running a separate `configureAlivenessCheck` command for each of them. However, there are possible disadvantages:

- loss of efficiency (WSDL may be large and generating it resource-intensive)

- inappropriate for endpoints registered from a copy of the WSDL that was hand-created and/or copied from another server, since you are then checking the aliveness of that HTTP server instead of the endpoint

- inappropriate for observer-discovered endpoints whose WSDL is saved as text in the sphere, since you are then checking the aliveness of the sphere.

Examples

The following example shows the `configureAlivenessCheck` command with the `getwsdl` method for performing the aliveness check.

```
btmcli configureAlivenessCheck -baseAddress http://jbujesPC:8080 -method getwsdl
                -s http://localhost:8080/btmcentral/sphere/ -l admin:abracadabra
```

10.7 createOrUpdateGenre

Use the `createOrUpdateGenre` command to create or update the definition of the indicated service or endpoint genre in the sphere. The genre object provides the information needed to properly display service and endpoint type information such as friendly name and icon.

In general, this command will be used when an observer is released off-cycle. At such times, explicit instructions will be provided about the `createOrUpdateGenre` command to use to teach your current system how to recognize the new genre and how to display services or endpoints of that type.

Command Syntax

```
btmcli createOrUpdateGenre -n name -id id
                           -t service | endpoint [iconUrl iconUrl]
                           -overwrite
                           -sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The user visible or friendly name of the genre. By convention this is often the local name of the qualified name used as the genre's id (when the id is given as a QName). The value specified will be used as the string to display in the Type field for services and endpoints.; for example, WEB_SERVICE, JAXRPC, DATABASE.
-id	--	The unique identifier for the genre. By convention this is often given as a qualified name with a namespace of `urn:com.amberpoint.servicegenres`. This value must match the genre identifier returned by the observer when discovering services and/or endpoints. For example: `urn:com.amberpoint.servicegenres:JAXRPC`
-t	-type	Indicates whether this genre represents a service type or an endpoint type.

Name	Long Name	Description
-iconUrl	--	The URL where the system can find the icon used to represent the genre in the management console.
		There will be two icon files for any genre. The file names will be grey*Type*Icon.*ext* and grey*Type*OperationIcon.*ext*, where the value of *ext* specifies the type of the icon file: gif, bmp, jpeg, or png. The value of *Type* is either Endpoint of Service.
		■ A URL with the prefix asc:// indicates that the icon should be found inside the views/images/genre/urlPath directory of the btmui application.
		■ Any other URL will be used directly as the base of the icon file URL.
-overwrite	--	A switch that specifies whether it is permissible to replace any existing genre of the same id. You must specify this switch when doing an update.
-s	-sphereUrl	The URL of the sphere where the service whose setup data you want to get is registered. http://hostname:port/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_ URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `createOrUpdateGenre` command used to create the endpoint type `MyGenre`.

```
btmcli createOrUpdateGenre -name MyGenre
                           -id urn:com.aberpoint.servicegenres:MyGenre
                           -s http://localhost:8080/btmcentral/sphre/
                           -l admin:abracadabra
```

10.8 createOrUpdatePolicy

Use the `createOrUpdatePolicy` command to create or update the specified policy directly in the sphere.

Creating or updating a policy might involve the following three steps:

1. Generate the template settings document using the `createSettingsDocument` command.

2. Edit the generated document to provide the specific values required for the policy being created.

3. Create the policy using the `createOrUpdatePolicy` command, providing the settings document created in Step 2.

To create a new policy, you need to supply only a template name, type, and name. If the policy requires settings, you must also provide a settings document. You can provide additional information to more precisely identify the source template to specify the initial state of the policy, but these are not required.

To update a policy, you must provide a name, type, the `-overwrite` flag, and any arguments for elements you want to update. Any values you do not specify remain unchanged. Because the policy name and version identify the policy, you cannot use this command to change either. Instead, create a new policy with the desired name and/or version.

Command Syntax

```
btmcli createOrUpdatePolicy [-templateName templateName]
                            [-t type][-subType subType] [-templateVersion version]
                            -n name [-V version] [-desc description]
                            [-settings settingsDocument]
                            [-criteria criterion1 [criterion2]...]
                            [-enable] | [-disable]
                            [-overwrite] [-f]
                            [-s sphereUrl] -l username:password
```

Name	Long Name	Description
-templateName	--	The name of the template on which to base the new policy.
		This parameter is ignored when performing an update.
-t	-type	The policy type for the new policy.
		This parameter is also used to determine if the specified policy already exists.
-subType	--	The policy sub type for the new policy.
		This parameter is also used to determine if the specified policy already exists.
-templateVersion	--	The version of the template on which to base the created policy.
		This parameter is ignored when performing an update.
-n	-name	The name of the new policy.
		This parameter is also used to determine if the specified policy already exists.
-V	-version	The version of the new policy.
		This parameter is also used to determine if the specified policy already exists.
-desc	-description	A description of the policy.
-settings	-settingsDocument	The policy's settings document, created with the `createSettingsDocument` command.
-criteria	--	A list of application criteria for the policy. The form of each criterion is described below.
-enable	--	Marks the policy as enabled.
-disable	--	Marks the policy as disabled.

Name	Long Name	Description
-overwrite	--	Indicates that if the policy already exists, it should be updated with the specified information.
-f	-force	Update the policy even if it is marked internally as "cannot be updated."
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password*. This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Application Criteria

When creating or modifying a policy, you can specify a set of criteria to be used in determining the monitored objects to which the policy can be applied. The format of each criterion in the set is as follows:

monitoredObjectType [;*query* [,{UNION | EXCLUDE}]]+

For detailed information on how to specify criteria visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Example

The following example shows the `createOrUpdatePolicy` command used to enable the enforcement of a pre-existing policy, MyPolicy.

```
btmcli createOrUpdatePolicy -name MyPolicy -enable
                            -s http://localhost:8080/btmcentral/sphere/
                            -l admin:abracadabra
```

10.9 createSettingsDocument

Use the `createSettingsDocument` command to create a settings document for input to the `createOrUpdatePolicy` command.

A settings document is an XML file. The source of the settings document can be a policy template or an existing policy.

- If you specify a policy template, the command returns the default settings document for the policy template.

- If you specify a policy, the command returns a settings document with the settings for the current policy instance. You can edit the values in the settings document before submitting it to the createOrUpdatePolicy command.

To determine the type and subtype of a policy or policy template, select the appropriate object in the Navigator. The type and subtype for each policy or policy template is displayed in the summary pane.

Command Syntax

```
btmcli createSettingsDocument -n name [-t type] [-subType subType[ [-V version]
                              -fromPolicy [-o outputFile]
                              -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The name of the template or policy whose settings document you want to use.
-t	-type	The type of the policy template or policy whose settings document you want to use.
-subType	--	The subtype of the policy template or policy whose settings document you want to use.
-V	-version	The version of the policy template or policy whose settings document you want to use.
-fromPolicy	--	Indicates that the settings document should be constructed using a pre-existing policy instance and not a template.
-o	-outputFile	The file where the settings document is written. If this is not specified, the settings document is sent to standard output.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `createSettingsDocument` command used to create the settings document `MySettingsDoc`.

```
btmcli createSettingsDocument -namemySecurePolicy -o mySettingsDoc
                  -s http://localhost:8080/btmcentral/sphere/
                  -l admin:abracadabra
```

10.10 credStoreTool

Use the `credStoreTool` command to create, obtain, or delete a credential from the credential store. This command allows you to create three different types of credentials:

- credentials based on a username:password format.

  You use these credentials to access btmcli commands. Nearly all commands require a user name and password.

- credentials based on a trusted issuer and secret

Business Transaction Management components use this type of credential to establish trust relationships between them.

- binary credentials (AES keys for the System Default Encryption Key).

   An AES encryption key is used to encrypt sensitive data that is transmitted from one Business Transaction Management component to another or when that sensitive data is stored in a database or on disk.

Before you can use this command, you must install and configure the Oracle Java Platform Security classes as described in the *Business Transaction Management Installation Guide*.

The syntax of the `credStoreTool` command varies depending on the type of credential you are working with. The sections that follow provide syntax and parameter information for each kind of credential.

In all cases, the commands allow you to create a credential, to obtain a credential if you need to copy it to other servers, and to delete a credential.

10.10.1 Using user:password Credentials

This option is more secure than specifying the user name and password on the command line itself or in a script. For additional information in how you use this credential to access btmcli commands, see Section 10.1.3, "Security Options in Accessing CLI Commands."

Command Syntax

```
btmcli credStoreTool -createCred credName [-credType up]
                     [credValue username:password]

btmcli credStoreTool -getCred credName [-credType up] [-showPwd]

btmcli credStoreTool -deleteCred credName [-credType up]
```

Name	Description
-createCred	Specifies the name of the credential to be created.
-getCred	Specifies the name of the credential to be obtained.
-deleteCred	Specifies the name of the credential to be deleted.
-credType	The kind of credential to be created, obtained, or deleted. The default is up.
-credValue	If you do not specify this option, you will be prompted for a user name and password. The password entered will be masked with asterisks.
-showPwd	For the `getCred` option, asks that the user name and password be displayed.

10.10.2 Using Trusted Issuer and Secret Credentials

Business Transaction Management components use this type of credentials to establish a trust relationship. When a component receives a request for a service, before it acts, it checks that it came from one of its trusted cohorts.

Command Syntax

```
btmcli credStoreTool -createCred credName credType is [-credValue issuer:secret]
```

```
btmcli credStoreTool -getCred credName -credType is [-showSecret]

btmcli credStoreTool -deleteCred credName -cretType is
```

Name	Description
-createCred	Specifies the name of the credential to be created.
-getCred	Specifies the name of the credential to be obtained.
-deleteCred	Specifies the name of the credential to be deleted.
-credType	The kind of credential to be created, obtained, or deleted.
-credValue	If you do not specify this option, you will be prompted for an issuer and secret. The secret entered will be masked with asterisks.
-showSecret	For the getCred option, asks that the issuer and secret be displayed.

10.10.3 Using Binary Credentials

An AES encryption key is used to encrypt sensitive data that is transmitted from one Business Transaction Management component to another or when it stored in a database or on disk.

Command Syntax

```
btmcli credStoreTool -createCred credName
          -credType bin {credValue Base64-encoded-bytes | -genKey AlgName:KeySize}

btmcli credStoreTool -getCred credName credType bin [-showSecret]

btmcli credStoreTool -deleteCred credName -credType bin
```

Name	Description
-createCred	Specifies the name of the credential to be created.
-getCred	Specifies the name of the credential to be obtained.
-deleteCred	Specifies the name of the credential to be deleted.
-credType	The kind of credential to be created, obtained, or deleted.
-credValue	Specify a set of bytes (base-64 encoded). These bytes might or might not represent a valid encryption key.
-genKey	*AlgName* refers to the JCE (Java Cryptographic Extension) reserved algorithm name. Currently only AES is supported. The *KeySize* is the size of the key that you want to generate. Different algorithms have different allowable key sizes. For AES, these are 128, 192, and 256, which refer to bits (not bytes). 128 is recommended because this is supported in all of the underlying platform's cryptographic implementations.
-showSecret	For the getCred option, asks that the size (in bytes) of the binary credential be displayed along with the base-64 encoded bytes themselves. For example, 16 bytes long ... Base-64 = [qvw1wEOxprSeJf2TbtuK5w==] If you do not specify this parameter, the bytes will not be displayed.

10.11 deleteAll

Use the `deleteAll` command to delete objects already discovered along with related artifacts such as transactions, properties, registered services, devices, and containers.

This is useful when initial discovery configuration results in the discovery of too many objects or of objects that are not of interest. Using this command allows you to start over without having to reinstall the system or to manually remove all observed entities and related artifacts.

> **Caution:**   The `deleteAll` command gives you a clean slate and should be used judiciously to avoid unwanted loss of data, which includes historical data related to observed objects. It is best used when you start working with BTM and are fine tuning your discovery scheme. It should never be used in a production environment.

Run the command with the `-preview` option to determine what objects will be deleted. Run the command again without the `-preview` option to actually delete these objects.

After you use this command, you can adjust your discovery configuration and then run more traffic to produce the desired results.

Command Syntax

`deleteAll [-f][-P]-s sphereUrl -l username:password`

Name	Long Name	Description
-f	-force	Forces the deletion of any existing transaction definitions. To retain definitions, you should export them before using the `deleteAll` command, and reimport them after you re-define your discovery configuration. See the discussion below for more information.
-P	-preview	Shows the effect of the command by displaying a list of objects to be deleted, without executing the command.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Discussion

When you start working with BTM, figuring out a good discovery configuration might take several tries. The `deleteAll` command makes it easy to start over.

There are a number of cases in which discovery results might prove overwhelming or uninteresting. Generally this happens when you accept default probe settings, which enable the monitoring of SOA based applications or OSB environments.

- SOA-based applications depend upon a large number of internal interconnected EJB components.

- OSB WebLogic application server hosts a large number of servlets which, in turn, invoke EJB's or other components.

Equally, WEB_APP and EJB probes are also enabled by default which, after a short time, results in discovering more than what you expected..

Making all these components visible is likely to produce too much information. You want to scale back the picture to the elements of interest.

Other factors might also contribute to producing confusing observation results:

- You install an observer where it does not belong. Services and endpoints are discovered that you are not interested in.

- It is not always obvious how to optimize the use of BTM monitors until you get to know how BTM works. You might decide to add a monitor and to reassign different probes to different monitors. Objects discovered before and after this change are both now visible.

- You update deployments, which results in seemingly redundant service or endpoint registrations.

In sum, you want to start over completely without having to reinstall the system or to manually remove all observed entities and related artifacts. The deleteAll command allows you to do just this.

The deleteAll command deletes data about the following:

- non-system containers

- user services and service interfaces

  Both user-registered endpoints and discovered endpoints are deleted.

- devices

- dependencies

- properties

- transaction definitions (using the -f option)

- observers and their probes (not the installed bits, just their representation in the Sphere)

- persisted runtime data related to the items listed above

  This includes measurements and captured message data.

To ensure that you do not lose transaction definitions that you might need later, the deleteAll command requires that you use the -force option to force their deletion. If you want to use these definitions in the future, you should export the transactions and then re-import them after you have redefined your discovery configuration. If you have used properties to enable manual correlation, BTM will re-create these properties to specify those dependencies.

The deleteAll command does not delete data about the following:

- system and user policies.

  These will be reapplied to newly discovered and registered entities. If you created policies that no longer apply to anything newly discovered, you will need to remove them.

- consumers

- schedules

- alerts

- system configuration

Examples

The following example show how you use the `deleteAll` command to delete all discovered and associated artifacts from your system. Sample output follows the command:

```
btmcli deleteAll

Warning: This command will delete your discovered / registered items in BTM.
Continue? (y/n) [n]: y
Deleted Services:
     OrderService
     ShippingService
     bookmartClient
     PurchasingDB
     CreditService
     WarehouseService

Deleted Containers:
    jmeredit-lap:9200

Deleted Observer Probes:
    Web Application
    Servlet
    Web Service
    JAX-RPC
    Database
    JDBC
    JMS Message Consumer
    JMS
```

10.12 encryptPassword

Use the `encryptPassword` command to convert the specified text string using the Business Transaction Management encryption engine and to write the resulting encrypted password to standard output.

You can then copy and paste the output string to provide credentials to commands that require them.

Command Syntax

```
btmcli encryptPassword -password aPlainTextPswd
```

Name	Long Name	Description
-password	--	A text string that you want to encrypt using the Business Transaction Management encryption engine. Each encrypted output of the same text string provides a unique ciphertext string.

Example

```
btmcli encryptPasword -password "abracadabra"
```

10.13 exportBusinessObjects

The `exportBusinessObjects` command exports one or more of the specified business objects from the target sphere. The command outputs an XML document that you can use to import the business objects into another target sphere.

You can specify the objects to be exported using the `-query` flag or the `-exportAll` flag.

Command Syntax

```
btmcli exportBusinessObjects [-exportAll] | [-query filterQuery ]
                             [-o outputFile ]
                              -s sphereUrl -l username:password
```

Name	Long Name	Description
-exportAll	--	Export all objects of the appropriate type.
-query	-filterQuery	Export only those objects that match the criteria expressed by the *filterQuery* expression.
		Use the Filter tool to construct the query, and then copy and paste the query into the command line.
-o	-outputfile	The name of the XML file where the data is written. If you omit this flag, the data is written to standard output.
-s	-sphereUrl	The URL of the sphere.
		http://*hostname:port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Examples

The following example shows the `exportBusinessObject` command used to output data to `MyConsumers`.

```
btmcli exportBusinessObject -exportAll -o MyConsumers
                  -s http://localhost:8080/btmcentral/sphere/
                  -l admin:password
```

10.14 exportMessages

Use the `exportMessages` command to export (request and reply) messages for an endpoint for the period during which logging has been turned on. Exporting messages can be useful in testing and debugging.

You have the option of exporting messages to an XML file. If you do not specify a file, output is sent to the stdout.

IMPORTANT: Message logging must be turned on for this command to work. See Section 5.4, "Defining, Modifying, and Deleting Transactions," for information about message logging.

Command Syntax

```
btmcli exportMessages -name name | -e endpointUrl
                    [-bn bindingName]
                    -baseDate date -baseTime time -duration minutes
                    [-outputFile outputFile]
                    [-max numberOfMessages]
                    -sphereUrl sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The friendly name of the endpoint for which messages are exported. See the discussion below.
-e	-endpointURL	The endpoint URL. If this is a web service, you can supply the actual URL to reach the endpoint. If this is another type of component, look in the Profile tab in the management console to determine the endpoint's URL. See also the discussion below.
-bn	-bindingName	The binding for the endpoint (for example, SOAP/HTTP). Use this parameter to disambiguate endpoint URLs with multiple bindings.
-baseDate	--	The start timestamp (date) in the format *yyyy-MM-dd* when messages are to be exported.
-baseTime	--	The start timestamp (time) in the format *hh:mm:ss* when messages are to be exported.
-duration	--	The time in minutes, starting from the base time during which logged messages are exported. Messages might be exported for a shorter time if the number of messages exported during the stated interval exceeds the value specified with the -max option.
-o	-outputFile	The path name of the file where messages are to be exported. The command fails if the output file already exists. It is recommended that you specify an absolute path. If you specify a filename only, the file is created in the directory from where the host application server is launched. IMPORTANT: If you specify a path, the command will work only if the process hosting Business Transaction Management has write access to the specified directory.
-max	--	The maximum number of messages to export. If you don't specify a value, all messages will be exported for the specified interval.
-s	-sphereURL	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.

Name	Long Name	Description
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

You can specify the endpoint for which messages should be exported in the following ways:

- Use the -e option if the endpoint URL is uniquely associated with a single endpoint.

  The -e option is also useful to identify an endpoint that belongs to a replicated service; in this case, the endpoint name would not be unique, but the endpoint URL would.

- Use the -n option to specify the friendly name of the endpoint. This is the preferred option if the specified name can uniquely identify the endpoint

Examples

The following command exports messages for the service `MyShippingService`.

```
btmcli exportMessages -name MyShippingService
                      -baseDate 2010-3-15 -baseTime 12:15:30 -duration 10
                      -s http://localhost:8080/btmcentral/sphere/ -l admin:passme
```

10.15 exportPolicies

Use the `exportPolicies` command to export the selected policies either to stdout or to the given output file. You specify the policy to export in one of two ways:

- The name of the policy - you can further qualify this with type, sub-type and/or version.

- The type of the policy - you can further qualify this with sub-type and/or version.

  To determine the type and subtype of a policy, select Policies in the Navigator. The type and subtype for each policy is displayed in the summary pane.

After exporting policies, you can use the `importPolicies` command to import them to another sphere.

Not all files can be exported. Data that you cannot migrate between sphere environments includes the following items:

- Logging policy data collected by your transaction management database.

  You cannot migrate information that has been collected and written to either file or database storage through the application of logging policies on your managed services.

- Instance data for exceptional conditions that have occurred in your transactions.

  You cannot migrate information that has been collected by the transaction management component of AMS.

Command Syntax

```
btmcli exportPolicies
          [[-n PName] [-t PType] [-subType PSubType] [-v Version]|[-exportAll]]
          [-o fileName]
          -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The name of the policy to export.
-t	-type	The type of the policy to export.
-subType	--	The sub-type of the policy to export.
-v	-version	The version of the policy to export.
-exportAll	--	Specifies that all policies should be exported.
-o	-outputFile	Specifies a file into which the results of the export will be placed.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following command exports all policies known to the specified sphere to the file `MyPolicies`.

```
btmcli exportPolicies -exportAll -o C:\Home|MyPolicies
                   -s http://localhost:8080/btmcentral/sphere/
                   -l admin abracadabra
```

10.16 exportPolicyTemplates

Use the `exportPolicyTemplates` command to export the selected policy templates either to stdout or to the given output file. You specify the policy template to export in one of two ways:

■ The name of the policy template - you can further qualify this with type, sub-type and/or version information.

■ The type of the policy template - you can further qualify this type with sub-type and/or version.

To determine the type and subtype of a policy template, select **Policy Templates** in the Navigator. The type and subtype for each policy template are displayed in the summary pane.

Command Syntax

```
btmcli exportPolicyTemplates
```

```
[[-n Name][-t Type] [-subType subType][-v TmplVersion]]| -exportAll
[-o fileName]
-s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The name of the policy template to export.
-t	-type	The type of the policy template to export.
-subType	--	The sub-type of the policy template to export.
-v	-version	The version of the policy template to export.
-exportAll	--	Specifies that all policy templates should be exported.
-o	-outputFile	Specifies a file into which the exported data is placed.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following command exports all the policy templates to the output file
`MyPolicyTemplates`

```
btmcli exportPolicyTemplates -exportAll -o C:\Home\MyPolicyTemplates
                             -s http://localhost:8080/btmcentral/sphere/
                             -l admin:abracadabra
```

10.17 exportProfile

Use the `exportProfile` command to export the profile values for the specified object(s). The resulting profile information is either written to the specified file, or to stdout if no file is provided. Specify the object(s) to be exported by specifying their type and then either their name or id.

You can use the `importProfile` command to import the resulting XML file into another sphere.

Command Syntax

```
btmcli exportProfile -t objectType
                    [-n name] [-v version] | [-id id]
                    [-o fileName]
                    -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The name of the object whose profile you want to export.
-t	-type	The type of the object whose profile you want to export. Type can be one of the following: deployment, service, interface, endpoint, container, DEPLOYMENT, SERVICE, INTERFACE, ENDPOINT, or CONTAINER.
-V	-version	The version of the service object whose profile you want to export.
-id	--	The sphere ID of the object whose profile data you want to export.
-o	-outputFile	Specifies a file into which the results of the export will be placed.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the encryptPassword command.

Example

The following command exports profile information for all containers.

```
btmcli exportProfile -t CONTAINER -o C:\Home|myContainers
                    -s http://localhost:8080/btmcentral/sphere/ -l admin:pass
```

10.18 exportSchedules

Use the exportSchedules command to export schedules from the target sphere. You can output the schedules to a file or to stdout.

You can specify all schedules, a specific schedule, or schedules of a certain type.

After export, you can use the importSchedules command to import the schedules to a different sphere.

Command Syntax

```
btmcli exportSchedules -n Name | -query filterQuery | -exportAll
                    [-o fileName]
                    -s sphereUrl sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The name of the schedule to export.
-query	-filterQuery	An AQL query specifying the set of schedules to export. Use the Filter tool to construct the query, and then copy and paste the query into the command line.
-exportAll	--	Specifies that all schedules should be exported.

Name	Long Name	Description
-o	-outputFile	Specifies a file into which the results of the export will be placed. If this flag is not used, output goes to stdout.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following command exports all schedules known to the specified sphere to the file `MySchedules`.

```
btmcli exportSchedules -exportAll -o C:\Home\MySchedules
                       -s http://localhost/btmcentral/sphere/ -l admin:pass
```

10.19 exportTransactionDefns

Use the `exportTransactionDefns` command to export transaction definitions known to the target sphere. If you do not specify an output file, the definition is sent to stdout.

You can import exported definitions using the `importTransactionDefns` command.

Command Syntax

```
btmcli exportTransactionDefns [-n Name] [-o fileName]
                              -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The name of the transaction to export. If you do not provide this value, all existing transaction definitions are exported.
-o	-outputFile	Specifies a file into which the results of the export will be placed. If you do not provide this value, the output is sent to stdout.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following command exports all transaction definitions to the file `MyTransactions`.

```
btmcli exportTransactionDefns -o C:\Home\MyTransactions
                    -s http://localhost:8080/btmcentral/sphere/
                    -l admin:abracadabra
```

10.20 getSetupData

Use the `getSetupData` command to write the setup file for a system service to standard output.

You can redirect output to a file, modify the file, and then use that file as input to the `putSetupData` command to save the service setup data.

You can also get the setup data from one service and transfer it to another. This command does not verify that the target service has been initially configured.

See Section 12.4, "Backing up and Restoring Business Transaction Management," for a discussion of the use of this command in backing up the system.

Command Syntax

```
btmcli getSetupData -e endpointUrl -s sphereUrl -l username:password
```

Name	Long Name	Description
-e	-endpointUrl	The location (URL) for the system service.
-s	-sphereUrl	The URL of the sphere where the service whose setup data you want to get is registered. http://hostname:port/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password*. This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `getSetupData` command.

```
btmcli getSetupData -e http://localhost:8080/apcentral/policyManager
                    -s http://localhost:8080/btmcentral/sphere/
                    -l admin:abracadabra
```

10.21 importBusinessObjects

The `importBusinessObjects` command imports one or more of the specified business objects described by the specified XML file into the target sphere. Currently, the only business object defined is the consumer business object, which is the means used to segment transaction information by consumer.

By default, this command replaces all existing objects that are older than the imported objects. You might use this command to migrate data from one environment to another or to populate your environment with a set of consumers.

You provide the data to import using the -inputFile argument or standard input. The objects you import must have first been exported using the exportBusinessObjects command.

If you are using this command to populate your environment with a set of consumers, the trickiest part is to create a valid input file. Here's a way to do this without having to guess at how to format your input file properly:

1.   If needed, customize the business object by adding custom attributes.

2.   Use the management console to create a consumer.

3.   Use the exportBusinessObject command, specifying the name of an output file. Data about the consumer you created in Step 2 will be exported to an XML file.

4.   Examine the output XML file. Using the format shown in the file, expand the file to add more consumers.

     See Section 7.1, "About Consumers," for information on working with consumers.

Command Syntax

```
btmcli importBusinessObjects [-i inputFile]
                   [-replace ALL | all | NEWER | newer | NONE | cleaned | CLEANED]
                   [-v] -s sphereUrl -l username:password
```

Name	Long Name	Description
-i	-inputFile	The name of the file containing the data to be imported.
		The input file must contain at least one business object to import.
-replace	-replacementLevel	Determines the outcome when an imported object matches an existing object in the system:
		ALL: replace all existing objects
		NEWER: replace existing objects that are older than imported objects. (Default)
		CLEANED: remove all existing business objects and then import the new business objects.
		NONE: do not replace any existing objects.
-v	-verbose	Display more detailed information about command execution.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the encryptPassword command.

Example

The following example shows the importBusinessObject command used to import data from the file MyConsumers.

```
btmcli importBusinessObject -inputFile MyConsumers -replace ALL
                    -s http://localhost:8080/btmcentral/sphere/
                    -l admin:abracadabra
```

10.22 importPolicies

Use the importPolicies command to import previously exported policies.

The imported policies are taken from the file to which they were exported (-inputFile parameter), or from stdin if no such file is specified.

Not all files can be imported. Data that you cannot migrate between sphere environments includes the following:

- Logging policy data collected by your transaction management database.

  You cannot migrate information that has been collected and written to either file or database storage through the application of logging policies on your managed services.

- Instance data for exceptional conditions that have occurred in your transactions.

  You cannot migrate information that has been collected by the transaction management component of Business Transaction Management.

By default, this command replaces all existing policies that are older than the imported policies.

Command Syntax

```
btmcli importPolicies [-i inputFile]
                    [-replace ALL | all | NEWER | newer | NONE | none]
                    [-v] -s sphereUrl -l username:password
```

Name	Long Name	Description
-i	-inputFile	The name of the file containing the data to be imported.
-replace	-replacementLevel	Determines the outcome when an imported policy matches an existing policy in the system: ALL: replace all existing policies NEWER: replace existing policies that are older than the imported policies. (Default) NONE: do not replace any existing policies.
-v	-verbose	Display more detailed information about command execution.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.

Name	Long Name	Description
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Examples

The following example shows the `importPolicies` command used to import data from the file `MyPolicies`.

```
btmcli importPolicies -inputFile MyPolicies -replace ALL
                -s http://localhost:8080/btmcentral/sphere/
                -l admin:abracadabra
```

10.23 importPolicyTemplates

Use the `importPolicyTemplates` command to import the policy templates contained in a previously created export file. The policy templates to import are taken from the file provided by the -i parameter, or from stdin if no such file is specified.

By default, this command replaces all existing templates that are older than the imported templates.

Command syntax

```
btmcli importPolicyTemplates [-i inputFile]
                [-replace ALL | all | NEWER | newer | NONE | none]
                [-v] -s sphereUrl -l username:password
```

Name	Long Name	Description
-i	-inputFile	The name of the file containing the data to be imported.
-replace	-replacementLevel	Determines the outcome when an imported policy template matches an existing policy template in the system: ALL: replace all existing policy templates. NEWER: replace existing policies that are older than imported policy templates. (Default.) NONE: do not replace any existing policy templates.
-v	-verbose	Display more detailed information about command execution.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.

Name	Long Name	Description
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `importPolicyTemplates` command used to import data from `MyPolicyTemplates`.

```
btmcli importPolicyTemplate -inputFile MyPolicyTemplates -replace NONE
                            -s http://localhost:8080/btmcentral/sphere/
                            -l admin:abracadabra
```

10.24 importProfile

Use the `importProfile` command to import a previously object profile into the system. The profile data to import is taken from the specified input file or from stdin if no file is specified.

Importing a profile does not cause new objects to be created, but only updates the profile attributes of existing objects. For example, importing a service profile does not create the service or register it with the system, it only updates the profile for a service that has already been registered (by virtue of discovery or manual registration).

Command Syntax

```
btmcli importProfile [-i importFile] [-m EXACT | BEST_FIT}
                     -s sphereUrl -l username:password
```

Name	Long Name	Description
-i	-inputFile	The name of the file containing the profile data to be imported. If not specified, the command imports data from stdin.
--	-match	Use when importing the profile for a set of related objects to specify how closely the imported set must match the objects that currently exist in the system. The possible values are:
		■ EXACT - the imported profile must match the existing objects exactly in both cardinality and structure.
		■ BEST_FIT - the imported profile will be matched against the existing objects as best as possible. Any mismatches in either cardinality or structure are ignored. This is the default value.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.

Name	Long Name	Description
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Examples

The following example shows the `importProfile` command used to import data from MyContainers.

```
btmcli importProfile -inputFile MyContainers -match EXACT
                     -s http://localhost:8080/btmcentral/sphere/
                     -l admin:abracadabra
```

10.25 importSchedules

Use the `importSchedules` command to import previously exported schedules to the target sphere.

By default, this command replaces all existing schedules that are older than the imported schedules.

Command Syntax

```
btmcli importPolicyTemplates [-i inputFile]
                     [-replace ALL | all | NEWER | newer | NONE | none]
                     [-v] -s sphereUrl -l username:password
```

Name	Long Name	Description
-i	-inputFile	The name of the file containing the data to be imported. If this is not specified, data is imported from stdin.
-replace	-replacementLevel	Determines the outcome when an imported schedule matches an existing schedule in the system:
		ALL: replace all existing schedules.
		NEWER: replace existing schedules that are older than imported schedules. (Default)
		NONE: do not replace any existing schedules.
-v	-verbose	Display more detailed information about command execution.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.

Name	Long Name	Description
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example imports schedules from the file `MySchedules` and replaces all existing schedules.

```
btmcli importSchedules -inputFile MySchedules -replace ALL
                       -s http://localhost:8080/btmcentral/sphere/
                       -l admin:abracadabra
```

10.26 importTransactionDefns

Use the `importTransactionDefns` command to import previously exported transaction definitions to the specified sphere.

By default, this command replaces all existing definitions that are older than the imported definitions and it recreates any missing properties that are needed to define the transaction.

Normally if a transaction contains errors, you cannot import it. Setting the `-importWithErrors` flag allows you to import the transaction anyway, but in a disabled state. You can then use console tools to fix the errors. Errors are displayed in the transaction's **Profile** tab.

It is possible that a transaction depends on one or more properties (for example, if these properties are used for correlation). In this case, the system will recreate the properties if any have been deleted before the transaction is imported. The system does not recreate the property if the transaction does not need it. You can specify the `-failOnMissingProperty` flag to have the import fail if it cannot recreate all the properties needed by the transaction. To take an example:

1. You create property X on operation A, and you use this property to correlate two operations in a transaction.

2. You create property Y on operation B. This property is not needed by the transaction; it might be useful to surface a particular value.

3. When you export the transaction, the system makes a copy of any properties that affect the transaction definition.

4. You now delete property X.

5. When you re-import the transaction, the system knows that a property needed for the transaction definition is missing. By default, it recreates the missing property. If you specify the `-failOnMissingProperty` flag, the import operation will fail.

6. Because the transaction does not depend on property Y, the system does not make a copy of it. This property is lost during the export/import process. You can recreate it after you import the transaction. Or you can use the `exportProfile` and

importProfile commands to have the system automatically recreate any such non-definitional properties.

Command Syntax

```
btmcli importTransactionDefns [-i inputFile]
                              [-replace ALL | all | NEWER | newer | NONE |none]
                              [-failOnMissingProperty]
                              [-importWithErrors]
                              -s sphereUrl -l username:password
```

Name	Long Name	Description
-i	-inputFile	The name of the file containing the data to be imported.
-replace	-replacementLevel	Determines the outcome when an imported transaction definition matches an existing definition in the system:
		ALL: replace all existing definitions.
		NEWER: replace existing definitions that are older than imported definitions. (Default)
		NONE: do not replace any existing definitions.
--	-failOnMissingProperty	By default, the system recreates a missing property if the transaction depends on it. Specify this flag if you want the import to fail if a property is missing.
--	-importWithErrors	Specify to have the command save the imported transaction even if it has errors.
		Setting this flag allows you to correct the errors in the console. Normally any error in the definition will cause the import to fail. With this flag, the import will succeed, but the imported transaction will be in the disabled state. You can then use console tools to view and correct the errors. Errors are displayed in the Profile tab of the transaction.
-s	-sphereUrl	The URL of the sphere.
		http://*hostname:port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password*. This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the encryptPassword command.

Example

The following example shows the importTransactionDefns command used to import data from the file MyTxDefns.

```
btmcli importTransactionDefns -i MyTxDefns -replace NONE -failOnMissingProperty
                              -s http://localhost:8080/btmcentral/sphere/
```

```
        -l admin:abracadabra
```

10.27 listInstruments

Use the `listInstruments` command to list the instruments defined by the policy templates in the system, as well as the attribute names and segments for each. You can use this information as arguments for the `retrieveObjectData` command.

Command Syntax

`btmcli listInstruments -s` *sphereUrl* `-l` *username*:*password*

Name	Long Name	Description
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Sample Output

Here is some sample output for the `listInstruments` command.

```
Avg Response Time
   Attribute Name: averageResponseTime
   Description: The average amount of time a transaction requires to complete,
                or a service/operation requires to respond.
   Segmented By:
           operation
           containingManagedObjectId
           clientAddress
           clientName
           businessObjectId
            businessObjectKey
Faults
    Attribute Name: faultCount
    Description: The number of faults generated by a service or operation.
    Segmented By:
        operation
        containingManagedObjectId
        clientAddress
        clientName
        faultCode
        businessObjectId
        businessObjectKey
```

Example

The following example shows the `listInstruments` command.

```
btmcli listInstruments -s http://localhost:8080/btmcentral/sphere/
                         -l admin:abracadabra
```

10.28 listNodeAliases

Use the `listNodeAliases` command to list the aliases for all known nodes or for the specified node.

Command Syntax

```
btmcli listNodeAlias [ n nodeName]
                        -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The canonical name of the network node. Typically a fully-qualified domain name. For example, jbujes-myPC.edgility.com
		If you do not specify this flag, aliases for all known nodes are displayed.
-s	-sphereUrl	The URL of the sphere.
		http://*hostname*:*port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `listNodeAlias` command.

```
btmcli listNodeAliases -n jbujesPC.edgility.com
                        -s http://localhost:8080/btmcentral/sphere/
                        -l admin:abracadabra
```

10.29 mergeServices

Use the `mergeServices` command to merge two services (thus removing duplicates). This command removes the source version specified with the -V flag and moves its endpoints to the target version specified with the -T flag. See Section 12.7, "Resolving Discovery Issues," for a discussion of the discovery issues that might require you merge two endpoints.

After the merge, the target version contains all the endpoints from both versions. If the source version has any profile attributes or message properties that you want to retain for the merged versions, you must recreate these on the target version.

Any historical data maintained for the source version is merged into the data maintained for the target version. Once the merge is completed, it is not possible to return to two distinct aggregated data sets for the two service versions.

You can preview the results of this command before committing to the changes it defines. After previewing, you must run the command again without the -P option to actually make the changes.

Use the `moveEndpoints` command to move selected endpoints from one service version to another.

Command Syntax

```
btmcli mergeServices -n friendlyName | -qname qname
                     -V version
                     -T targetVersion [-P]
                     -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The friendly name of the service for which versions should be merged. If two versions do not have the same friendly name, you must use the `-qname` flag to specify the service name.
-qname	--	The qualified name of the service for which versions should be merged. The qualified name has the form *WSDLtargetNamespace:serviceName*. For example: http://amberpoint.com/orderService:CreditServiceSOAP.
-V	-version	The version of the service that will be merged into the specified target. Use the "-" string to represent the default, unnamed version.
-T	-targetVersion	The target service version in which to merge the source version (-V). After the merge, this version will contain the endpoints from both versions.
-P	-preview	Specify this flag to preview the results of the merge without actually performing the merge.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `mergeServices` command:

```
btmcli mergeServices -f CreditService -X 1.5 -T 1.6
                     -s http://localhost:8080/btmcentral/sphere/
                     -l admin:abracadabra
```

10.30 moveEndpoints

Use the `moveEndpoints` command to move one or more endpoints from the source version to the target version of the specified service.

You can only move the entire set of endpoints defined for a service in a given WSDL as a unit, regardless of how you identify the endpoints.

You can preview the results of this command before committing to the changes it defines. After previewing, you must run the command again without the -P option to actually make the changes.

The `moveEndpoints` command does not move any measurements. The measurements for the endpoint remain associated with the endpoint and nothing happens to these when the endpoint is moved to a new service. The measurements for the old service version continue to include data previously aggregated from the moved endpoint. The historical measurements for the target service version are unchanged; new measurements for the endpoint are included in the target version of the service.

You can use this command to split a service in two by specifying the previously undefined version with the -T flag and using the -force flag.

Use the `mergeServices` command to merge all endpoints from one service version to another.

See Section 12.7, "Resolving Discovery Issues," for a discussion of the use of this command in resolving discovery issues.

Command Syntax

```
btmcli moveEndpoints -n friendlyName | -qname qname -w wsdlUrl | -e endpointUrl
                     -V version -T targetVersion [-P]
                     -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The friendly name of the service whose endpoints are moved.
		If two versions do not have the same friendly name, you must use the qname flag to specify the service name.
-qname	--	The qualified name of the service whose endpoints are moved.
		The qualified name has the form *WSDLtargetNamespace:serviceName*. For example: http://amberpoint.com/orderService:CreditServiceSOAP.
-w	-wsdlURL	The URL of the WSDL file that defines the service whose endpoints are moved.
-e	-endpointUrl	The URL of an endpoint in the source version of the service.
		The endpoint URL might identify multiple endpoints if the WSDL defines more than one endpoint at the same URL with different bindings. This is a common case for WSDLs that define more than one endpoint for a service.

Name	Long Name	Description
-V	-version	The version of the service that will be merged into the specified target. Use the "-" string to represent the default, unnamed version.
-T	-targetVersion	The service version in which to merge the source version (-V). After the merge, this version will contain the endpoints from both versions.
-P	-preview	Specify this flag to preview the results of the move without actually moving endpoints.
-f	-force	Create a new service version if the specified target version does not exist. You can assign this service a name different from that of the source version using the -newServiceName flag.
	-newServiceName	If you use the -f option, you can use the -newServiceName flag to specify a friendly name for the target version that is to be created. If a target version already exists, this flag is ignored.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the encryptPassword command.

Example

The following example shows the moveEndpoints command:

```
btmcli moveEndpoints -n catalogService
                     -e http://test.myhost.com:7412/catalogMgmt/svcs/CatalogSvc
                     -V 1.5 -T 1.6
                     -s http://localhost:8080/btmcentral/sphere/
                     -l admin:abracadabra
```

10.31 moveMeasurements

Use the moveMeasurements command to move the measurements collected for a service or endpoint that has been deleted from the sphere to another endpoint or service that is its logical successor.

Caution: Use this command with care. The system does not check to see that the services or endpoints are actually different iterations of the same object. Once merged, the measurements for the two objects cannot be separated.

When you deploy a new version of a service known to the sphere, and the modified WSDL is registered or discovered, the qualified name of the service defined in the WSDL is expected to match the name of the service found in the WSDL the last time it was read. If there is no match and the service name change option (in the service

versioning policy) has a value of "replace the previous endpoints (losing all measurements)," Business Transaction Management deletes the endpoints that were previously read from the WSDL and creates new endpoints belonging to the service currently defined in the WSDL.

If only the namespace or service name defined in the WSDL has changed, the new endpoints might look the same as the old ones (same URL and binding), but they will have a different unique ID. Business Transaction Management depends on this ID to associate measurements with their respective objects; if the ID changes, it leaves measurements made for the old endpoints orphaned and inaccessible. If the service and endpoints defined in the updated WSDL are logically the same, you should use the `moveMeasurements` command to re-associate the measurements with the new incarnation of the service or endpoint.

Since the old endpoint and service have already been deleted, the old object to move measurements from must be identified by its unique ID. You can determine what the ID of a deleted service is by checking the sphere system log, as follows:

1. Select **Administration > System Services** in the navigator.

2. Select AP_Sphere_Service in the summary pane.

3. Select **Open System Service Log for AP_Sphere_Service** from the **Admin** menu.

4. The Log Viewer is displayed: look for the deleted service, and note its Exchange ID value.

Because no metadata about the deleted object is known to the sphere, this command is limited in the validation it can do. The command will refuse to do anything if the target cannot be found or is not specified uniquely. It will also refuse to move any measurements if the ID specified refers to an extant object that has not been deleted from the sphere.

See Section 12.7, "Resolving Discovery Issues," for a discussion of the use of this command in resolving discovery issues.

Command Syntax

```
btmcli moveMeasurements -id id -n friendlyName | -qname qname [-V version]
                        [-e endpointUrl] [-bn bindingName]
                        -s sphereUrl -l username:password
```

Name	Long Name	Description
-id	--	The UUID of the deleted endpoint or service for which to move measurements.
		If the system deletes a service or endpoint because a WSDL has changed, it will send a system alert containing the UUID(s). Otherwise, you will need to check the service log for the sphere service (as described above) to get the id.
-n	-name	The friendly name of the target service where to move measurements.
-qName		The qualified name of the target service where to move measurements.
-V	-version	The version of the target service where to move measurements. Only required if there are multiple versions of the service.

Name	Long Name	Description
-e	-endpointUrl	The URL of the target endpoint to move measurements to.
-bn	-bindingName	The binding name of the target endpoint. (Only needed if the endpoint URL is not unique.)
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following examples show the use of the `moveMeasurements` command:

```
btmcli moveMeasurements -id 00EB2838-7E62-11DF-ABB5-27339B5EB390
                        -n MyCreditService
                        -s http://localhost:8080/btmcentral/sphere/
                        -l admin:abracadabra

btmcli moveMeasurements -id CAEAF3FF-C5A7-11DF-80D5-2DF5BB94EBDE
                        -e http://text12.myhost.com:7012/bookmart/ofder/OrderService
                        -s http://localhost:8080/btmcentral/sphere/
                        -l admin:abracadabra
```

10.32 putSetupData

Use the `putSetupData` command to read the setup file from standard input or a supplied input file and to save it as the setup data for the specified system service.

See Section 12.4, "Backing up and Restoring Business Transaction Management," for a discussion of the use of this command in backing up the system.

Command Syntax

```
btmcli putSetupData -e endpointUrl [-i inputFile]
                    -s sphereUrl -l username:password
```

Name	Long Name	Description
-e	-endpointUrl	The URL of the system service.
-i	-inputFile	The name of the file from which to get input.
-s	-sphereUrl	The URL of the sphere where the service whose data you want to read is installed. http://hostname:port/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.

Name	Long Name	Description
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following examples show files written as the setup file for another service at the specified endpoint.

```
cat myFile | btmcli putSetupData -e myServiceEndpoint
```

```
btmcli putSetupData -e myServiceEndpoint -i myInputFile
```

10.33 registerDevice

Use the `registerDevice` command to register a management device with the sphere. Although Business Transaction Management is able to infer the existence of load balancing devices by analyzing message traffic, explicitly registering a device enables the display of its friendly name and of other information, and provides access to the device's administrative interface if any.

Currently, hardware and software load balancers are the only supported device type for this command, and the F5 BigIP is the only fully-supported load-balancer.

In most cases, Business Transaction Management automatically detects and models routing entry points by observing message traffic and reading destination information from the message header. However, if the observed messages do not carry information about their original recipient (the load balancer) in the HTTP Host headers, you will need to manually assign a routing entry point to the load balancer. You will also need to add target entry points to indicate where the messages are being routed. You need to use the Business Transaction Management management console to do this. See Section 12.6, "Setting up Load Balancers," (setting up a load balancer) for a full discussion.

If you register multiple devices, you can use the `setDefaultLoadBalancer` command to set the default device. (Any newly discovered routing entry points will be modeled as part of the default device unless they belong to an F5 load balancer.)

To unregister a device, you must use the management console.

Command Syntax

```
btmcli registerDevice -baseAddress baseAddress -fn friendlyName
                      -vendor vendor
                      [-administratorUI administratorUI]
                      [-phaseInLifecycle DEBRECATED | deprecated
                      | DEVELOPMENT | development |PRODUCTION | production
                      | STAGING | staging |TEST | test]
                      -deviceLogin login:password
                      -s sphereUrl -l login:password
```

Name	Long Name	Description
--	-baseAddress	The base address of the URL of your load balancer; for example https://myf5:443/.
-fn	-friendlyName	The friendly name for your load balancer. This name is displayed in the Business Transaction Management Management Console.
--	-vendor	If you are registering an F5 BigIP device, specify F5 (case insensitive). When set to this value, the vendor flag is functional.
		For other types of load balancers, this flag is required, but it is purely descriptive. You can specify any value (except F5). The specified value is displayed in the Management Console.
--	-administratorUI	For devices other than F5, the URL of your load balancer's HTML administrative console. The management console uses this URL to display a link to your load balancer's console.
		This flag is not required for F5 devices because the URL is automatically obtained.
--	-phaseInLifecycle	Specifies the lifecycle phase of the device container and all of the endpoints that are created within it. Valid values are deprecated, development, production, staging, and test. These are case insensitive.
--	-deviceLogin	This flag is required for F5 devices and ignored for other types of load balancers.
		The login credentials for the F5 load balancer in the format *login:password.* A user role of Guest provides sufficient privileges.
		If this is a limited-privilege account, Business Transaction Management will discover the objects and policies managed by the device, but you might not be able to modify them using Business Transaction Management.
		You can encrypt passwords using the encryptPassword CLI command, for example:
		btmcli encryptPassword -password "myPassword"
-s	-sphereUrl	Required unless you have set the AP_SPHERE_URL environment variable for your sphere.
		Specify the URL of the sphere with which you want to register the device; for example:
		http://mySphereHost:8080/btmcentral/sphere/
		Replace the host name and port number with appropriate values. This URL always ends with btmcentral/sphere/.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the encryptPassword command.

Example

The following example shows how you use the `registerDevice` command to register an F5 load balancer.

```
btmcli registerDevice -baseAddress https://myF5:443
                      -fn MyF5LoadBalancer
                      -vendor F5
                      -phaseInLifecycle development
                      -deviceLogin my_F5_username:my_F5_password
                      -s http://mySphereHost:1771/btmcentral/sphere/
                      -l admin:abracadabra
```

10.34 registerExternalContainer

Use the `registerExternalContainer` command to group the endpoints of one or more entry points (not part of an observed container) into their own external container. This is only necessary for manually-registered endpoints. In the console, such endpoints show up under the **Unassigned Endpoints** node.

Note that there is no command that allows you to move endpoints between containers individually; you have to use this command to move all endpoints starting with the given base address.

By default, manually registered endpoints are allocated to the System container. The `registerExternalContainer` command allows you to create an external container for endpoints that you want to treat and examine as a group. For example, this container might allow you to model the fact that two or more base addresses for manually registered endpoints are located in the same application server.

You cannot use base addresses from a physical container or a device, but you can use base addresses from another external container if you specify the -f flag. That is, you cannot group endpoints that have been discovered into an external container.

The optional flags for this command allow you to specify information that is then displayed in the container's profile information. To update this information, execute this command again with different option values.

When updating an existing container, if the first entry in the list of base addresses changes, the container's primary base address also changes. If you omit base addresses that previously belonged to the container, they are excluded from the container and returned to the System container.

You can run this command before or after services with endpoints in this container have been registered.

Command Syntax

```
btmcli registerExternalContainer -baseAddress baseAddress1 [[baseAddress2]...]
                                 [-fn friendlyName] [-vendor vendor]
                                 [-containerInfo containerInfo]
                                 [-ofName osName] [-osVersion osVersion]
                                 [-administratorUI administratorUI]
                                 [-phaseInLifecycle DEBRECATED | deprecated
                                 | DEVELOPMENT | development
                                 |PRODUCTION | production
                                 | STAGING | staging |TEST | test]
                                 [-caseInsensitive] [-f]
                                 -s sphereUrl -l username:password
```

Name	Long Name	Description
--	-baseAddress	Known base addresses for endpoints that you want to associate with the container. If you specify more than one, the first one is considered the primary base address.
-fn	-friendlyName	The friendly name for the new container. By default, this is the *host:port* of the primary base address.
--	-vendor	Vendor of the application server software for the container; for example, IBM, Oracle, Microsoft.
--	-containerInfo	The name and version of the application server software for the container in the format name/version. For example, WebLogic Server/10.3.
--	-osName	The name of the operating system the container is running in. For example, Windows or Linux.
--	-osVersion	The version of the operating system the container is running in. For example: 2003, 4.3.
--	-administratorUI	A URL that can be used to access the administrative UI of the container.
--	-phaseInLifecycle	Specifies the lifecycle phase of the container.
--	-caseInsensitive	If set, URLs in this container are compared without attention to case.
-f	-force	Specifies that you want to re-associate a base address that currently belongs to a different external container.
-s	-sphereUrl	The URL of the sphere with which you want to register the container. http://hostname:port/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example registers an external container that will contain the endpoints with the specified base addresses.

```
btmcli registerExternalContainer
                    -baseAddress http://my_host:1234/ http://my_host/4567/
                    -s http://my_host:1234/btmcentral/sphere/
                    -l admin:abracadabra
```

10.35 registerMonitor

Use the `registerMonitor` command to register a monitor agent with the sphere.

Following installation and configuration of Business Transaction Management central services, you can deploy one or more monitor nodes to additional application servers

for processing observed messages. You need to run the `reigsterMonitor` command for each monitor you want to add.

If the nodes hosting Business Transaction Management monitors use aliases, you can run the `addNodeAlias` command to register the aliases with the sphere.

Command Syntax

```
btmcli registerMonitor -e endpointUrl [-fn friendlyName]
                        -s sphereUrl -l username:password
```

Name	Long Name	Description
-e	-endpointUrl	Specifies the URL of the monitor to register with the sphere. For example, http://localhost:8080/btmmonitor/agent/agent Replace the host name and port number with appropriate values. This URL always ends with btmmonitor/agent/agent.
-fn	-friendlyName	The friendly name of the endpoint for the monitor agent you want to register.
-s	-sphereUrl	Required. The URL of the sphere with which you want to register the container. http://hostname:port/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

```
btmcli registerMonitor -e http://localhost:8080/btmmonitor/agent/agent
                       -s sphereUrl http://localhost:8080/btmcentral/sphere/
                       -l admin:abracadabra
```

10.36 removeBaseAddressAlias

Use the `removeBaseAddressAlias` command to remove the specified alias to the entry point with the given base address

The alias must be defined locally for this entry point only. Aliases defined at the network node level must be removed using the `removeNodeAlias` command.

The `removeBaseAddressAlias` command allows you to remove incorrect or outdated aliases from the system. Although it does not undo the removal of duplicate artifacts caused by the complementary `addBaseAddressAlias` command, any new discoveries for the base address and the ex-alias will again be treated as separate entities.

See Section 12.7, "Resolving Discovery Issues," for a discussion of the use of this command in resolving discovery issues.

Command Syntax

```
btmcli removeBaseAddressAlias -baseAddress address -alias address
                              -s sphereUrl -l username:password
```

Name	Long Name	Description
--	-baseAddress	The base address of an existing entry point in the sphere. Use the format *scheme*://*host*:*port*/
-alias	--	The base address to remove as an alias for the above entry point.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `removeBaseAddressAlias` command.

```
btmcli -removeBaseAddressAlias http://jbujes:8080 -alias http://1.2.3.4:8080
                    -s http://localhost:8080/btmcentral/sphere/
                    -l admin:abracadabra
```

10.37 removeDuplicateEndpoint

Use the `removeDuplicateEndpoint` command to remove the specified duplicate endpoint.

Typically, you would use this command when the system discovers two endpoints with the same path but different host names due to a DNS alias that was not known to the system. Executing this command also adds the missing alias to make sure that future discoveries do not result in recreating the duplicate endpoint.

If duplicates differ only in their base address, it is assumed that other endpoints found under both base addresses are also duplicates. In addition, the service descriptors (WSDLs) for the duplicate endpoints are also deleted from the model. WSDLs and endpoints under the duplicate entry point that are not actually duplicates are not removed; they are moved under the remaining entry point.

If the specified endpoint URL refers to a dependency-discovered endpoint and the duplicate URL refers to a discovered or manually-registered endpoint, the command is reversed: the duplicate endpoint is preserved and the base address of the endpoint URL is treated as an alias.

You can preview the results of this command before committing to the changes it defines. After previewing, you must run the command again without the -P option to actually make the changes.

See Section 12.7, "Resolving Discovery Issues," for a discussion of the use of this command in resolving discovery issues.

Command Syntax

```
btmcli removeDuplicateEndpoint -e endpointUrl -d duplicateUrl [-P]
                               -s sphereUrl -l username:password
```

Name	Long Name	Description
-e	-endpointUrl	The URL of the endpoint to preserve.
-d	-duplicateUrl	The URL of the endpoint to remove as a duplicate.
-P	-preview	Preview the results of the move without actually moving endpoints.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Alternate Means of Removing Duplicates

In the following situations, the `removeDuplicateEndpoint` command will refuse to make the specified changes to avoid compromising the integrity of the model or destroying information that is not actually duplicated.

- The endpoints belong to different services or different versions of the same service.

- The endpoints implement different interfaces.

- The endpoints live in different deployments.

- The WSDL for the duplicate endpoint refers to services or endpoints that are not referenced by the WSDL for the remaining endpoint; that is, the WSDLs cannot be considered duplicates.

In these cases, you might need to do the following:

1. Use the `unregister` command to unregister the service endpoints that cannot be removed with the removeDuplicateEndpoint command.

2. Use the `addNodeAlias` or `addBaseAddressAlias` commands to add aliases that help the system recognize endpoint duplication.

Example

The following example shows the `removeDuplicateEndpoint` command:

```
btmcli removeDuplicateEndpoint
            -e http://jbujes.edge.com:7001/bookmart/credit/CreditService
            -d http://jbujes.delphi.com:7001/bookmart/credit/CreditService -P
            -s http://localhost:8080/btmcentral/sphere/
            -l admin:password
```

10.38 removeNode

Use the `removeNode` command to delete information about the specified node from the sphere database. Removing a node deletes all information related to that node including its aliases.

Removing a node is not something you would normally need to do unless aliases for different machines have gotten so mixed up that you face the choice of pruning nodes or reinstalling the system. Under most circumstances (bad hosts file corrected, DNS alias changed to point to a different machine), removing an alias from one node and adding it back to a different node should be all that you need to do.

Command Syntax

```
btmcli removeNode -n nodeName -alias aliasName
                  -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The canonical name of the network node. Typically a fully-qualified domain name. For example, jbujes-myPC.edgility.com
-f	-force	Forces the node to be removed even if it is still referenced by containers or entry points. If you assign this node's aliases to another network node before you remove this node, you can enable a container to associate itself with a different node.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `removeNode` command.

```
btmcli removeNode -n jbujes.edge.com
                  -s http://localhost:8080/btmcentral/sphere/
                  -l username:password
```

10.39 removeNodeAlias

Use the `removeNodeAlias` command to delete alias information from the sphere database.

See Section 12.7, "Resolving Discovery Issues," for a discussion of the use of this command in resolving discovery issues.

Command Syntax

```
btmcli removeNodeAlias -n nodeName -alias aliasName
```

```
-s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The canonical name of the network node. Typically a fully-qualified domain name. For example, jbujes-myPC.edgility.com
-alias	-alias	An alias for the network node. This can be a domain name or a static IP address.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the removeNodeAlias command.

```
btmcli removeNodeAlias -n jbujes.edge.com -alias Athena
                       -s http://localhost:8080/btmcentral/sphere/
                       -l admin:abracadabra
```

10.40 removePathAlias

Use the `removePathAlias` command to remove an alias representing an alternative URL path that can be used to access the specified WSDL or endpoint. Any future registration or discovery will again treat the two paths as distinct.

See also `removeNodeAlias` for host name aliases and `removeBaseAddressAlias` for other aliases in the authority part of the WSDL's URL.

The `removePathAlias` command allows you to remove outdated aliases from the system. For instance, if a service descriptor has an alias added due to an HTTP redirect that is later disabled (WSDL URL A redirected to URL B, but now the two URLs represent different WSDLs), this command removes the alias so that WSDL A can be registered independently.

See Section 12.7, "Resolving Discovery Issues," for a discussion of the use of this command in resolving discovery issues.

Command Syntax

```
ap removePathAlias -w wsdlUrl| -e endpointUrl -alias pathAlias
                   -s sphereUrl -l username:password
```

Name	Long Name	Description
-w	-wsdlUrl	The full URL or the WSDL (service descriptor) for which you want to remove an alias.

Name	Long Name	Description
-e	-endpointUrl	The full URL of the endpoint(s) for which to remove the alias. If multiple endpoints are matched, the alias will be removed from all of them.
-alias	--	The alternate path to remove from the service descriptor or endpoint. This should be just the path portion of the URL, starting with "/".
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `removePathAlias` command using an encrypted password.

```
btmcli removePathAlias -e http://jbujes:7001/credit/CreditService
                        -alias /bookmart/amex/CreditSrevice
                        -s http://localhost:8080/btmcentral/sphere/
                        -l admin:abracadabra
```

10.41 removePolicy

Use the `removePolicy` command to completely remove an applied policy from the Business Transaction Management system.

Command Syntax

```
btmcli removePolicy -name PlcyName [-type PlcyType] [-subType PlcySubType]
                    [-version PlcyVersion] [-wait true|false] [-f]
                    -s sphereUrl -l username:password
```

 To determine the type and subtype of a policy, select **Policies** in the Navigator. The type and subtype for each policy is displayed in the main pane.

Name	Long Name	Description
-n	-name	The name of the policy you want to remove. If this name is unique, you do not need to supply additional type or version information.
-t	-type	The policy type for the policy you want to remove. Use this flag if the policy name is not unique.
-subType	--	The policy subtype for the policy you want to remove. If the name and type combination is not unique, specify a policy subtype.

Name	Long Name	Description
-V	-version	The version associated with the policy you want to remove.
		If the name, type, and subtype triple is not unique, specify the version number. Version numbers are automatically appended to policy names to create unique policies.
-wait	--	TRUE: Wait for the policy to be removed before executing another script.
		The default value is FALSE.
-f	-force	Remove the policy even if it is marked internally "do not remove."
-s	-sphereUrl	The URL of the sphere where the policy is applied.
		http://hostname:port/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Example

The following example removes the policy `myLoggingPolicy` from the sphere, and waits until the policy is removed before executing any other command.

```
btmcli removePolicy -name myLoggingPolicy -wait true
                    -s http://localhost:8080/btmcentral/sphere/
                    -l admin:abracadabra
```

10.42 removePolicyTemplate

The `removePolicyTemplate` command removes the specified policy template from the system.

Command Syntax

```
btmcli removePolicy -name PolTemplName [-type Type] [-subType SubType]
                    [-version PolicyVersion] [-f]
                    -s sphereUrl -l username:password
```

To determine the type and subtype of a policy template, select **Policy Templates** in the Navigator. The type and subtype for each policy is displayed in the main pane.

Name	Long Name	Description
-name	--	The name of the policy template you want to remove. If this name is unique, you do not need to supply additional type or version information.
-type	--	The policy type for the policy template you want to remove. Use this flag if the policy name is not unique.

Name	Long Name	Description
-subType	--	The policy subtype for the policy template you want to remove. If the name and type combination is not unique, specify a policy subtype.
-version	--	The version associated with the policy you want to remove. If the name, type, and subtype triple is not unique, specify the version number. Version numbers are automatically appended to policy names to create unique policies.
-f	-force	Remove the policy template even if it is marked internally "do not remove."
-s	-sphereUrl	The URL of the sphere where the policy is applied. http://hostname:port/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password*. This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example removes the policy template `myPolicyTemplate` from the sphere.

```
btmcli removePolicyTemplate -name myPolicyTemplate
                            -s http://localhost:8080/btmcentral/sphere/
                            -l admin:abracadabra
```

10.43 renameEndpoint

Use the `renameEndpint` command to change the specified endpoint's friendly name in the sphere.

When multiple endpoints at the same URL are given the same name in their respective WSDL definitions, giving them unique friendly names can make it easier to distinguish between them in future commands and in command output. You must provide the endpoint URL and a new friendly name. All other parameters are optional and are included only to help you distinguish the desired endpoint from others with the same URL.

See Section 12.7, "Resolving Discovery Issues," for a discussion of the use of this command in resolving discovery issues.

Command Syntax

```
btmcli renameEndpoint -e endpointUrl -fn friendlyName [-bn bindingName]
                      [-w wsdlUrl] [-n name] [-service serviceName]
                      [-qname qname] [-V version] [-interfaceQName interfaceQName]
                      -s sphereUrl -l username:password
```

Name	Long Name	Description
-e	--	The endpoint URL of a web service.
-bn	-bindingName	The binding for the endpoint. Used to disambiguate endpoint URLs with multiple bindings. For example: SOAP and HTTP
-w	-wsdlUrl	The URL of the WSDL file.
-n	-name	The existing friendly name of the endpoint to be renamed.
-service	--	The friendly name of the service to which the renamed endpoint belongs.
-qname	--	The qualified name of the service to which the renamed endpoint belongs.
-V	-version	The version of the service to which the renamed endpoint belongs. Use with the -service or -qname flags if needed. Use the "-" string to represent the default, unnamed version.
--	-interfaceQName	The qualified name of the interface the renamed endpoint implements.
-fn	-friendlyName	The new friendly name for the endpoint to be renamed.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the encryptPassword command.

Example

The following example shows the renameEndpoint command used to rename CreditService to MyCreditService.

```
btmcli renameEndpoint -e http://jbujes:7001/bookmart/credit/CreditService
                      -fn MyCreditService
                      -s http://localhost:8080/btmcentral/sphere
                      -l admin:abracadabra
```

10.44 resetSphereUrl

Use the resetSphereUrl command to make sure that the sphere URL for container services currently registered with this sphere matches the current sphere URL. You might need to use this command if the base address of the sphere changes.

Command Syntax

```
btmcli resetSphereUrl -s sphereUrl -l username:password
```

Name	Long Name	Description
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/apcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `resetSphereUrl` command.

```
btmcli resetSphereUrl -s http://localhost:8080/btmcentral/sphere/
                      -l admin:abracadabra
```

10.45 retrieveObjectData

The `retrieveObjectData` command retrieves profile and measurement data for one or more objects. You can use this command's output file in the following ways:

■ Input to a reporting tool.

Business Transaction Management provides Oracle BI Publisher templates that you can use with the command's output file to produce formatted reports. See Chapter 9, "Creating Reports," for more information on using these templates.

■ Input to the `setBaseLines` command.

In this case, you set performance baseline values for a set of objects by using the output file of `retrieveObjectData` as an input file to `setBaselines`. Setting baselines enables you to then create a baseline SLA policy. (Do not set the `retrieveObjectData` command's attribute if you are using the output file to set baselines.)

Command syntax

```
btmcli retrieveObjectdata -t type1 [[type2] ...] [-query filterQuery]
          [-attributeNames attrName1 [attrName2] ... instrName1 [instrName2] ...]
          [-segments segment1 [segment2] ...]
          [-inTheLast inTheLast] | [-startTime startTime] [-endTime endTime]
          [-intervalSize intervalSize]
          [-format XML|CSV|SQL]
          [-o outputFile]
          [-metadata drop|create|none] [-tableName tableName]
          -s sphereUrl -l username:password
```

You can use the `listInstruments` command to get the information about attributes, segments, and instruments that you must pass as arguments to the `retrieveObjectData` command. If you specify a list of attributes and a list of segments, the command parser applies each segment to any instrument in the attribute list that has that segment.

If no time values are specified, the last hour's data is retrieved in a single segment.

Name	Long Name	Description
-t	-type	Specifies the object tree whose data is to be retrieved.
		At a minimum a root object type must be given that determines (along with the query) which objects will be retrieved.
		In addition to this, you can specify any number of 'child' types. These trees mirror those found in the custom data explorer. An example type tree is 'service endpoint operation' or 'observer probe endpoint'.
-query	-filterQuery	Limits the objects to be retrieved to the set defined by the query filter.
		The query must apply to the root object type of the tree. Only those objects that match the given query will be returned in the result set, along with the associated objects in the sub-tree.
		Use the Filter tool to construct AQL queries, and then copy and paste these into the command line.
--	-attributeNames	Use this argument to specify which attribute and instrument values to retrieve for each object, for example:
		-attributeNames friendlyName averageResponseTime
		In this example, friendlyName is an attribute and averageResponseTime is an instrument. You can specify multiple attribute names (separated by spaces) followed by multiple instrument names (separated by spaces).
		Attribute names can refer to any attribute that is available in the column chooser. Attributes can be qualified by an object type followed by two colons ('::'), for example:
		Endpoint::friendlyName
		If you specify a qualified attribute name, attribute values are retrieved for only the specified object type (in the example, that would be the friendly names of only endpoints). If you specify an unqualified attribute name, attribute values are retrieved for all object types listed in the type argument that have that attribute (for example, endpoints, services, and operations). See Section 10.1.6, "Attribute Names," for a complete list of commonly used attribute names and object types.
		For a complete list of instrument names, run the `listInstruments` command.
-segments	--	Specifies the segments to use when retrieving the data for any specified measurement attributes (instruments). This provides a breakdown of the data for each specific value of the named segment that was recorded during the specified time-frame.
-inTheLast	--	Specifies a time range extending from the current time back into the past by the indicated amount.
		The value is given as a number followed by a unit (s, S = seconds; m, M = minutes; h, H = hours; d, D = days) e.g. "1H" for "1 Hour".
-startTime	--	Specifies the start of a time range over which to produce values for any specified measurement attributes (instruments). If 'endTime' is not specified, the range ends at the current time.
		The current default date format is used, e.g. "2010-08-03 15:00:00".

Name	Long Name	Description
-endTime	--	Specifies the end of a time range over which to get values for any specified measurement attributes (instruments).
		The current default date format is used, e.g. "2010-08-03 18:00:00".
--	-intervalSize	Indicates the size of the intervals into which the specified time range is to be divided.
		The interval size is given as a number followed by a unit (s, S = seconds; m, M = minutes; h, H = hours; d, D = days) e.g. "1H" for "1 Hour". The interval size must be less than or equal to the length of the specified time range.
		If this optional argument is not specified the entire time range will be used and only a single interval is returned.
		Note: Do not set this attribute if you are using the output file as input to the setBaselines command. The setBaselines command cannot accept more than one value per instrument type per object. Omitting this attribute ensures that only one value per instrument type per object is output.
-format	--	Specifies the format of the output file. Legal values are XML, CSV, and SQL. XML is the default format.
-o	-outputFile	Name of the output file.
-metadata	--	Specifies whether SQL metadata commands are created.
		Valid values are "drop" to drop and create the table, and "create" to only create the table.
-tablename	--	Specifies the SQL table name used for SQL output; if no name is specified then a name is generated.
-s	-sphereUrl	The URL of the sphere. For example, http://*hostname:port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the encryptPassword command.

Example 1: Retrieving data for containers.

```
btmcli retrieveObjectData
        -t container
        -s http://myHost:7003/btmcentral/sphere/
        -l my_sphere_username:my_sphere_password
        - my_containers_output.xml
```

Example 2: Retrieving the average response time for each service for purposes of reporting (note that -intervalSize is specified).

```
btmcli retrieveObjectData
        -t service
        -s http://my_host:7003/btmcentral/sphere/
```

```
                    -l my_sphere_username:my_sphere_password
                    -attributeNames firendlyName averageResponseTime
                    -inTheLast 24h
                    -intervalSize 1h
                    -format xml
                    -o my_avg_response_time_output_for_reporting.xml
```

Example 3: Retrieving the average response time for each service, endpoint, and operation for purposes of setting baselines (note that `-intervalSize` is not specified).

```
btmcli retrieveObjectdata
          -t service endpoint operation
          -s http://my_host:7003/btmcentral/sphere/
          -l my_sphere_username:my_sphere_password
          -attributeNames friendlyName averageResponseTime
          -inTheLast 24h
          -o my_avg_response_time_output_for_baselines.xml
```

10.46 sendEventNotification

Use the `sendEventNotification` command to create and send an event notification using the Notifier Service. How the notification is handled is determined by the currently active subscriptions.

The system automatically generates alerts when one of the following happens:

- An SLA enforcement policy issues a failure, warning, or compliance alert

- A condition definition causes the Transaction Monitoring Service to issue a condition alert.

- A system service throws an exception, causing the system to issue a system alert.

In addition to these cases, an administrator might also want to issue an alert to notify users that the system needs to come down for maintenance or upgrade, or to let users know that some administrative action will regenerate the data used to create reports. The `sendEventNotification` command is provided to enable this kind of notification. (It would also be possible to send an event notification using the `sendToNotifier` command, but it would be a lot more work.)

Command Syntax

```
btmcli sendEventNotification -source source -n name
                  -topic topic [-subtopic subtopic]
                  -severity SEVERE | WARNING | INFO | CONFIG | FAILURE | SUCCESS
                  -message message
                  [-attributes name1:value1 [[name2:value2] ...]]
                  -s sphereUrl -l username:password
```

Name	Long Name	Description
-source	--	Indicates where the notification was generated. Possible sources include ASLM, ExM, and System. You can specify an arbitrary value.
-n	-name	The name associated with the event notification. This should provide information as to what the event is about.
-topic	--	The event notification's topic. Used to help categorize similar event notifications.

Name	Long Name	Description
-subtopic	--	The event notification's subtopic. Used to help categorize event notifications.
--	-severity	Indicates the importance of the situation that caused the notification.
-message	--	A textual description of what the event notification means and whether any user action is required.
--	-attributes	A list of *name:value* pairs that will be attached to the event notification and made available to anyone who receives the notification. These attributes typically reference objects related to the notification.
-s	-sphereUrl	The URL of the sphere. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `sendEventNotification` command used to notify current subscribers that the system is about to be backed up.

```
btmcli sendEventNotification -source System -n BackupWarning
            -topic SysNotifications -severity WARNING
            -message "System backup in 10 minutes. Please save your work."
            -s http://localhost:8080/btmcentral/sphere/
            -l admin:abracadabra
```

10.47 sendToNotifier

You can use the `sendToNotifier` command to manage email and web service subscriptions.

The `sendToNotifier` command sends a command (specified in an XML input file) to the Notifier Service for processing. You can create an email subscription, list existing subscriptions, delete subscriptions, and so forth. The command returns the response document returned by the Notifier Service. To issue additional commands, you must call `sendToNotifier` again, specifying an input file that contains a different command.

Command syntax for `sendToNotifier` is fairly simple, but creating the input command document can be complicated.

Business Transaction Management currently supports email and web service subscriptions only through the command line interface.

The command document, described at the end of this section, can contain an xml message (command) that accomplishes one of the following tasks:

■ Create an email subscription

- Create a subscription to call a notification operation on a web service

- List existing subscriptions

- Delete a subscription from the notifier

Command Syntax

```
btmcli sendToNotifier -i inputFile [-o outputFile]
                      -s sphereUrl -l username:password
```

Name	Long Name	Description
-i	-inputFile	An XML file containing the command to send to the Notifier Service. Each file can contain one command.
-o	-outputFile	A file to hold the response from the Notifier Service. If none is specified, the response is sent to stdout.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

10.47.1 Setting up Email Notification

The use of `sendToNotifier` is one step in setting up email notification. The overall procedure also includes the following steps:

1. Select **System Services** from the Navigator and select **AP_Notifier_Service** from the services listed in the summary area.

2. Edit its setup data (from the **Admin** menu) to specify the mail host and the email sender to be used.

3. Use the `sendToNotifier` command to register a subscription to SLA alerts, condition alerts, and so on. (The input command document will contain a command that specifies which alerts you are interested in.) For example, the following document registers a subscription for condition alerts:

```
<ap:registerSubscription
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <ap:subscription name="All Exception Alerts" enabled="true" userID="testID">
    <ap:deliveryMechanism>
      <ap:sendEmail format="long" style="html">
          <ap:from>BTM.TEST@oracle.com</ap:from>
          <ap:recipient type="TO">your.name@yourdomain.com</ap:recipient>
      </ap:sendEmail>
   </ap:deliveryMechanism>
   <ap:notificationFilters>
      <ap:source>
          <ap:value>ExM</ap:value>
          <ap:operation>in</ap:operation>
      </ap:source>
    </ap:notificationFilters>
```

```
        </ap:subscription>
      </ap:registerSubscription>
```

The input command document is described in Section 10.47.3, "The Notifier Command Document."

4. Define SLA or condition alerts for the objects of interest.

10.47.2 Managing Email Notifications

You can also use `sendToNotifier` to list subscriptions and delete subscriptions. Here are sample command documents that illustrate how you would do this.

To list subscriptions

```
<ap:getSubscription xmlns:ap="http://namespace.amberpoint.com/amf"/>
```

To cancel subscriptions

```
<ap:cancelSubscription xmlns:ap="http://namespace.amberpoint.com/amf">
 <ap:subscription name="All Exception Alerts" userID="testID"/>
 <ap:subscription name="All SLA Alerts" userID="testID"/>
 <ap:subscription name="All System Alerts" userID="testID"/>
</ap:cancelSubscription>
```

Examples

The following example shows the `sendToNotifier` command.

```
btmcli sendToNotifier -i MyNotifierCommand -o MyNotifierReponse
              -s http://localhost:8080/btmcentral/sphere -l admin:abracadabra
```

10.47.3 The Notifier Command Document

A Notifier Service command (xml message) can be included in a command document that is input to the `sendToNotifier` command. Notifier service commands are described in the following subsections; they allow you to do the following:

- Create an email subscription
- Create a web service subscription
- List subscriptions
- Delete a subscription
- Get notification interest
- Get a template

You may include only one command per document. To execute several commands, you must create a document for each command and invoke the `sendToNotifier` command for each document.

10.47.3.1 Notifier Service Commands

The following subsections describe the commands you can include in a Notifier Document.

Create an Email Subscription

This command supports the following variations:

- Create a subscription for all alerts of a particular type (e.g. system alert, SLA alert, condition alert)

- Create a subscription for all alerts on a particular object (e.g. transaction, service, endpoint, operation, consumer, system service)

- Create a subscription for all alerts of a particular severity (e.g. failures and warnings, but not compliance or info alerts)

- Create a subscription for all alerts generated by a particular policy (e.g. all alerts generated by the Baseline SLA Policy)

- Create a subscription for all alerts generated by a particular condition (e.g. all condition alerts for the FaultMonitoring condition)

In addition to the sample commands below, you also need to be familiar with the metadata used to specify alert types. This is described in the next subsection.

```
<ap:registerSubscription xmlns:ap="http://namespace.amberpoint.com/amf"
                xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
                xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/1999/XMLSchema"
                soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <ap:subscription name="All SLA Alerts" enabled="true" userID="testID">
   <ap:deliveryMechanism>
         <ap:sendEmail format="short" style="text">
           <ap:from>BTM.TEST@oracle.com</ap:from>
           <ap:recipient type="TO">your.email@oracle.com</ap:recipient>
         </ap:sendEmail>
   </ap:deliveryMechanism>
   <ap:notificationFilters>
         <ap:source>
           <ap:value>ASLM</ap:value>
           <ap:operation>in</ap:operation>
         </ap:source>
   </ap:notificationFilters>
 </ap:subscription>
</ap:registerSubscription>
```

Here is a more detailed example with comments:

```
<ap:registerSubscription xmlns:ap="http://namespace.amberpoint.com/amf"
            xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/1999/XMLSchema"
            soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <ap:subscription name="Warning and Failure SLA Alerts" enabled="true"
                                                    userID="testID">
     <ap:deliveryMechanism>
       <ap:sendEmail format="long" style="html">
         <ap:from>BTM.TEST@oracle.com</ap:from>
         <ap:recipient type="TO">your.name@oracle.com</ap:recipient>
       </ap:sendEmail>
     </ap:deliveryMechanism>
     <ap:notificationFilters>
      <ap:source>
         <ap:value>ASLM</ap:value><!-- subcribe to SLA alerts -->
         <ap:operation>in</ap:operation>
      </ap:source>
      <ap:severity>
       <ap:value>WARNING</ap:value> <!-- subcribe to WARNING and FAILURE alerts -->
       <ap:value>FAILURE</ap:value>
```

```
      <ap:operation>in</ap:operation>
     </ap:severity>
    </ap:notificationFilters>
 </ap:subscription>
 <ap:subscription name="Exception Alert transaction OrderService_submit"
                                      enabled="true" userID="testID">
 <ap:deliveryMechanism>
   <ap:sendEmail format="short" style="text">
     <ap:from>BTM.TEST@oracle.com</ap:from>
     <ap:recipient type="TO">your.name@oracle.com</ap:recipient>
   </ap:sendEmail>
 </ap:deliveryMechanism>
 <ap:notificationFilters>
    <ap:event>
     <ap:value>Exception Detected</ap:value> <!-- subscribe to Exception alert
                                                event-->
     <ap:operation>in</ap:operation>
    </ap:event>
   <ap:info>
     <ap:name>correlationName</ap:name> <!-- subcribe to tx by friendly name-->
     <ap:value>OrderService_submit</ap:value>
     <ap:operation>in</ap:operation>
   </ap:info>
  </ap:notificationFilters>
 </ap:subscription>
</ap:registerSubscription>
```

Create a Web Service Description

This command supports the following variations:

- Create a subscription for all alerts of a particular type (e.g. system alert, SLA alert, condition alert)

- Create a subscription for all alerts on a particular object (e.g. transaction, service, endpoint, operation, consumer, system service)

- Create a subscription for all alerts of a particular severity (e.g. failures and warnings, but not compliance or info alerts)

- Create a subscription for all alerts generated by a particular policy (e.g. all alerts generated by the Baseline SLA Policy)

- Create a subscription for all alerts generated by a particular condition (e.g. all condition alerts for the FaultMonitoring condition)

```
<ap:registerSubscription xmlns:ap="http://namespace.amberpoint.com/amf"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema"
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <ap:subscription name="Dispatch" enabled="true" userID="ABC">
       <ap:deliveryMechanism>
         <ap:callWebService>
           <ap:httpEndPoint>
         <ap:uri>http://10.148.118.72:8080/eProxy/svc/NotifPort_Proxy</ap:uri>
               <!-- this is my test webService, replace with your own.-->
         <ap:soapAction>ReplaceWithRealSoapActionForDotNetService</ap:soapAction>
           </ap:httpEndPoint>
         </ap:callWebService>
```

```
                  </ap:deliveryMechanism>
              <ap:notificationFilters>
                 <ap:source>
                              <ap:value>ASLM</ap:value>
                              <ap:value>ExM</ap:value>
                              <ap:value>System</ap:value>
                              <ap:operation>in</ap:operation>
                  </ap:source>
              </ap:notificationFilters>
        </ap:subscription>
  </ap:registerSubscription>
```

Get Subscriptions

This command supports the following variations:

- List all email subscriptions

- List all web service subscriptions

- List all subscriptions for a particular email address

- List all web service subscriptions for a particular web service

- List all subscriptions for a particular object (transaction, service, endpoint, operation, system service, consumer)

Here are some examples for these actions:

Get all subscriptions

```
<ap:getSubscription xmlns:ap="http://namespace.amberpoint.com/amf"/>
```

Get subscriptions by subscription name

```
<ap:getSubscription xmlns:ap="http://namespace.amberpoint.com/amf">
        <ap:subscription name="Dispatch"/>
</ap:getSubscription>
```

Get subscriptions by user name

```
<ap:getSubscription xmlns:ap="http://namespace.amberpoint.com/amf">
    <ap:subscription userID="mary"/>
</ap:getSubscription>
```

Cancel Subscription

This command supports the following variations:

- Delete all subscriptions for a particular email address

- Delete all web service subscriptions for a particular web service

- Delete all subscriptions on a particular object (e.g. transaction, service, endpoint, operation, system service, consumer)

For example,

```
<ap:cancelSubscription xmlns:ap="http://namespace.amberpoint.com/amf">
        <ap:subscription name="Dispatch" userID="mary"/>
</ap:cancelSubscription>
```

Get Format Information

This command returns available formats such as "long HTML", "short text." The following is sample output for the command:

```
<ap:getFormatInformationResponse xmlns:ap="http://namespace.amberpoint.com/amf"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <ap:for value="Email" xmlns:ap= "http://namespace.amberpoint.com/amf">
   <ap:format value="long"xmlns:ap="http://namespace.amberpoint.com/amf"/>
   <ap:format value="short" xmlns:ap="http://namespace.amberpoint.com/amf"/>
   <ap:style value="html" xmlns:ap="http://namespace.amberpoint.com/amf"/>
   <ap:style value="text" xmlns:ap="http://namespace.amberpoint.com/amf"/>
  </ap:for>
</ap:getFormatInformationResponse>
```

Get Notification Interest

Use a command like the following:

```
<ap:getNotificationInterest xmlns:ap="http://namespace.amberpoint.com/amf"/>
```

Get Template

Use a command like the following:

```
<ap:getTemplate/>
```

10.47.3.2 Metadata for Alert Types

You will need to be familiar with metadata for SLA, Exception, and system alerts. You use this information to properly format a subscription for a particular type of information. Metadata information is shown in the following tables.

Metadata for SLA Alerts

The following table specifies metadata for SLA alerts.

Metadata	Available values	Sample subscription

(header) source	ASLM	`<ap:notificationFilters>`
		`  <ap:source>`
		`   <ap:value>ASLM</ap:value>`
		`   <ap:operation>in</ap:operation>`
		`  </ap:source>`
		`</ap:notificationFilters>`

(header) event	enforcement_condition	`<ap:notificationFilters>`
		`   <ap:event>`
		`      <ap:value>enforcement_condition</ap:value>`
		`      <ap:operation>in</ap:operation>`
		`   </ap:event>`
		`</ap:notificationFilters>`

(header) severity	WARNING
	FAILURE
	SUCCESS
	INFO
		`<ap:notificationFilters>`
		`   <ap:severity>`
		`      <ap:value>WARNING,FAILURE</ap:value>`
		`      <ap:operation>in</ap:operation>`
		`   </ap:severity>`
		`</ap:notificationFilters>`

Metadata	Available values	Sample subscription
(header) templateSet	ASLM-Template	``` <ap:notificationFilters> <ap:templateSet> <ap:value>ASLM-Template</ap:value> <ap:operation>in</ap:operation> </ap:templateSet> </ap:notificationFilters> ```
(info) com.amberpoint.notification.enforcement Value	(value of SLA enforcement that triggered the alert)	``` <ap:notificationFilters> <ap:info> <ap:name>com.amberpoint.notification. instrumentName</ap:name> <ap:value>85</ap:value> <ap:operation>in</ap:operation> </ap:info> <ap:notificationFilters> ```
(info) com.amberpoint.notification.instrument Name	(qualified name of instrument)	See Note below table.
(info) com.amberpoint.notification.objectType	Service, Endpoint, Transaction	``` <ap:notificationFilters> <ap:info> <ap:name>com.amberpoint.notification. objectType</ap:name> <ap:value>Service</ap:value> <ap:operation>in</ap:operation> </ap:info> <ap:notificationFilters> ```

(info) com.amberpoint.notification.objectId	(UUID of the object that SLA policy is applied on)	```
		<ap:notificationFilters>
		  <ap:info>
		    <ap:name>com.amberpoint.notification.
		                        objectId</ap:name>
		    <ap:value>uuid:DA936829-98FF-11DF-828B-
		                    A3EAF532919E</ap:value>
		    <ap:operation>in</ap:operation>
		  </ap:info>
		<ap:notificationFilters>
		``

(info) com.amberpoint.notification.policyId	(UUID of the SLA policy)	```
		<ap:notificationFilters>
		  <ap:info>
		    <ap:name>com.amberpoint.notification.
		                        policyId</ap:name>
		    <ap:value>uuid:896FCF93-9900-11DF-8753-
		                    57E6CFA43E9B</ap:value>
		    <ap:operation>in</ap:operation>
		  </ap:info>
		<ap:notificationFilters>
		``

(info) com.amberpoint.notification.senderId	(UUID of the AP_ Enforcer_Service)	```
		<ap:notificationFilters>
		  <ap:info>
		    <ap:name>com.amberpoint.notification.
		                        senderId</ap:name>
		    <ap:value>uuid:896FCF93-9900-11DF-8753-
		                    57E6CFA43E9B</ap:value>
		    <ap:operation>in</ap:operation>
		  </ap:info>
		<ap:notificationFilters>
		``

Metadata	Available values	Sample subscription

(info) com.amberpoint.notification.senderUrl	(Location of AP_Enforcer_Service)	```<ap:notificationFilters>
		  <ap:info>
		    <ap:name>com.amberpoint.notification.
		                          senderUrl</ap:name>
		    <ap:value>http://localhost:8080/
		              btmperformance/endorcer</ap:value>
		    <ap:operation>in</ap:operation>
		   </ap:info>
		<ap:notificationFilters>```

(info) com.amberpoint.notification.transactionId	UUID of transaction (only if the SLA is applied on a Transaction)	```<ap:notificationFilters>
		  <ap:info>
		     <ap:name>com.amberpoint.notification.
		                          transactionId</ap:name>
		    <ap:value>uuid:896FCF93-9900-11DF-8753
		                          -57E6CFA43E9B</ap:value>
		    <ap:operation>in</ap:operation>
		   </ap:info>
		<ap:notificationFilters>```

(info) endpointId	(UUID of AP_Enforcer_Service system endpoint)	```<ap:notificationFilters>
		  <ap:info>
		   <ap:name>com.amberpoint.notification.
		                          endpointId</ap:name>
		    <ap:value>uuid:896FCF93-9900-11DF-8753-
		                          57E6CFA43E9B</ap:value>
		    <ap:operation>in</ap:operation>
		   </ap:info>
		<ap:notificationFilters>```

(info) serviceID	(UUID of AP_Enforcer_Service)	```<ap:notificationFilters>
		  <ap:info>
		     <ap:name>com.amberpoint.notification.
		                          serviceId</ap:name>
		    <ap:value>uuid:896FCF93-9900-11DF-8753-
		                          57E6CFA43E9B</ap:value>
		    <ap:operation>in</ap:operation>
		   </ap:info>
		<ap:notificationFilters>```

Sample subscription for (info) com.amberpoint.notification.instrumentName is as follows:

```
<ap:notificationFilters>
  <ap:info>
     <ap:name>com.amberpoint.notification.instrumentName</ap:name>
     <ap:value>http://schemas.amberpoint.com/instruments/transactions/
                               types:startedTransactionCount</ap:value>
     <ap:operation>in</ap:operation>
   </ap:info>
<ap:notificationFilters>
```

Metadata for Condition Alerts

The following table specifies metadata for condition alerts.

Metadata	Available Values	Sample Subscription
(header) source	ExM	``` <ap:notificationFilters> <ap:source> <ap:value>ExM</ap:value> <ap:operation>in</ap:operation> </ap:source> </ap:notificationFilters> ```
(header) event	Exception Detected	``` <ap:notificationFilters> <ap:event> <ap:value>Exception Detected</ap:value> <ap:operation>in</ap:operation> </ap:event> </ap:notificationFilters> ```
(header) severity	FAILURE, WARNING, INFO	``` <ap:notificationFilters> <ap:severity> <ap:value>WARNING</ap:value> <ap:operation>in</ap:operation> </ap:severity> </ap:notificationFilters> ```
(header) templateSet	Exm-Template	``` <ap:notificationFilters> <ap:templateSet> <ap:value>Exm-Template</ap:value> <ap:operation>in</ap:operation> </ap:templateSet> </ap:notificationFilters> ```
(info) com.amberpoint.notification. objectType	Transaction	``` <ap:notificationFilters> <ap:info> <ap:name>com.amberpoint.notification.objectType</ap:name> <ap:value>Transaction</ap:value> <ap:operation>in</ap:operation> </ap:info> <ap:notificationFilters> ```
(info) com.amberpoint.notification. objectId	(UUID of Transaction)	``` <ap:notificationFilters> <ap:info> <ap:name>com.amberpoint.notification.objectId</ap:name> <ap:value>uuid:896FCF93-9900-11DF-8753-57E6CFA43E9B </ap:value> <ap:operation>in</ap:operation> </ap:info> <ap:notificationFilters> ```
(info) com.amberpoint.notification. transactionId	(UUID of Transaction)	``` <ap:notificationFilters> <ap:info> <ap:name>com.amberpoint.notification.transactionId</ap:name> <ap:value>uuid:896FCF93-9900-11DF-8753-57E6CFA43E9B </ap:value> <ap:operation>in</ap:operation> </ap:info> <ap:notificationFilters> ```
(info) correlationName	(Friendly name of function)	``` <ap:notificationFilters> <ap:info> <ap:name>correlationName</ap:name> <ap:value>testTransaction1</ap:value> <ap:operation>in</ap:operation> </ap:info> <ap:notificationFilters> ```

Metadata	Available Values	Sample Subscription
(info) detectionTime	(Human readable time stamp. E.g. Thu Jul 22 12:25:49 PDT 2010)	```<ap:notificationFilters>
<ap:info>
<ap:name>detectionTime</ap:name>
<ap:value>Thu Jul 22 12:25:49 PDT 2010</ap:value>
<ap:operation>in</ap:operation>
</ap:info>
<ap:notificationFilters>```
(info) exceptionPriority	failure, warning, informational	```<ap:notificationFilters>
<ap:info>
<ap:name>exceptionPriority</ap:name>
<ap:value>warning</ap:value>
<ap:operation>in</ap:operation>
</ap:info>
<ap:notificationFilters>```
(info) exceptionTemplate	(friendly name of condition)	```<ap:notificationFilters>
<ap:info>
<ap:name>exceptionTemplate</ap:name>
<ap:value>condition2</ap:value>
<ap:operation>in</ap:operation>
</ap:info>
<ap:notificationFilters>```
(info) exceptionType	(name of transaction + name of condition Or name of service + name of condition)	```<ap:notificationFilters>
<ap:info>
<ap:name>exceptionType</ap:name>
<ap:value>testTransaction1+condition2</ap:value>
<ap:operation>in</ap:operation>
</ap:info> <ap:notificationFilters>```
(info) exceptionsURL	(URL that links back to the exception instance UI page)	```<ap:notificationFilters>
<ap:info>
<ap:name>exceptionsURL</ap:name>
<ap:value>http://localhost:8080/btmui</ap:value>
<ap:operation>in</ap:operation>
</ap:info>
<ap:notificationFilters>```
(info) instanceID	(UUID of exception instance)	```<ap:notificationFilters>
<ap:info>
<ap:name>instanceId</ap:name>
<ap:value>uuid:896FCF93-9900-11DF-8753-57E6CFA43E9B
</ap:value>
<ap:operation>in</ap:operation>
</ap:info>
<ap:notificationFilters>```

Metadata for System Alerts

The following table specifies metadata for system alerts.

Metadata	Possible Values	Sample Subscription
(header) source	System	```<ap:notificationFilters>
<ap:source>
<ap:value>System</ap:value>
<ap:operation>in</ap:operation>
</ap:source>
</ap:notificationFilters>```

Metadata	Possible Values	Sample Subscription

(header) event	business_object_creation	```<ap:notificationFilters>```
		``` <ap:event>```
		``` <ap:value>business_object_creation</ap:value>```
		``` <ap:operation>in</ap:operation>```
		``` </ap:event>```
		```</ap:notificationFilters>```

(header) severity	INFO	```<ap:notificationFilters>```
		``` <ap:severity>```
		``` <ap:value>INFO</ap:value>```
		``` <ap:operation>in</ap:operation>```
		``` </ap:severity>```
		```</ap:notificationFilters>```

(header) templateSet	System-Template	```<ap:notificationFilters>```
		``` <ap:templateSet>```
		``` <ap:value>System-Template</ap:value>```
		``` <ap:operation>in</ap:operation>```
		``` </ap:templateSet>```
		```</ap:notificationFilters>```

(info) com.amberpoint.notification.senderId	(UUID of service that sends out this alert.)	```<ap:notificationFilters>```
		``` <ap:info>```
		``` <ap:name>com.amberpoint.notification.senderId</ap:name>```
		``` <ap:value>uuid:896FCF93-9900-11DF-8753-57E6CFA43E9B</ap:value>```
		``` <ap:operation>in</ap:operation>```
		``` </ap:info>```
		```<ap:notificationFilters>```

(info) com.amberpoint.notification.senderUrl	(Location of service that sends out this alert.)	```<ap:notificationFilters>```
		``` <ap:info>```
		``` <ap:name>com.amberpoint.notification.senderUrl</ap:name>```
		``` <ap:value>http://localhost:8080/btmperformance/endorcer```
		``` </ap:value>```
		``` <ap:operation>in</ap:operation>```
		``` </ap:info>```
		```<ap:notificationFilters>```

10.48 setBaselines

The setBaselines command sets baselines for the specified objects. Setting baselines enables you to then create a baseline SLA policy.

You can use the retrieveObjectData command to retrieve and output historical performance data to a file; then you can use that file as input to the setBaselines command. If you create the input file by hand, use the same format as the output of the retrieveObjectData command.

You can also use this command to delete existing baselines for the specified object or for all managed objects.

Command Syntax

```
btmcli setBaselines -i inputFile [-delete] [-all]
                    -s sphereUrl -l username:password
```

Name	Long Name	Description
-i	--	An XML document that indicates the baseline values for services, service operations, endpoints, endpoint operations, and transactions. Baseline values can be used for baseline service level agreements (SLAs based on historically-derived performance thresholds).
-delete	--	Delete any existing baselines for the referenced managed objects before setting new baselines.
-all	--	When combined with the -delete option, deletes all existing baselines for all managed objects. In this case, an input file is not required.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the encryptPassword command.

Example

The following example shows the setBaselines command.

```
btmcli setBaselines -i MyBookmartBaselines -delete
            -s http://mySphereHost:1771/btmcentral/sphere/
            -l my_btm_admin_username:my_btmadmin_password
```

10.49 setDefaultLoadBalancer

Use the setDefaultLoadBalancer command to set the default device used to model routing entry points discovered from observed message traffic.

In a simple environment with only one known load-balancer, that device is automatically used as the default. If you register additional devices, use this command to set which device should be used to model routing entry points.

See Section 12.6, "Setting up Load Balancers," for a complete discussion of load balancers and how the default is set.

Command Syntax

```
btmcli setDefaultLoadBalancer -baseAddress baseAddress | -fn friendlyName
                            -s sphereURL -l username:password
```

Name	Long Name	Description
--	-baseAddress	The base address of the URL for the load balancer. For example, https://myf5:443/

Name	Long Name	Description
-fn	--	The friendly name of the load-balancer in the sphere. This name is displayed in the summary pane when you select Devices from the navigator. It is the name you assigned when you registered the load balancer.
-s	-sphereUrl	Required. The URL of the sphere with which you want to register the container. http://hostname:port/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example specifies the default routing device:

```
btmcli setDefaultLoadBalancer -baseAddress https://myF5:443/
                  -s http://localhost:8080/btmcentral/sphere -l admin:abracadabra
```

10.50 showService

Use the `showService` command to show the structure of the service specified by the friendly name, qualified name, or URL of any endpoint for this service. You can also specify a single version of the service (-version) or the display of more detailed information (-verbose).

The non-verbose output includes: the version number, the qualified name of the service, its friendly name, its binding, who it is managed by and its full URL.

Command Syntax

```
btmcli showService -n friendlyName  | -qname qname| -e endpointUrl
                                                      | -wwsdlUrl
                  [-V version] [-v]
                  -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The friendly name of the service for which information is sought. You can qualify the name using the-versionflag. Otherwise, information about all the services with the friendly name is displayed.
-qname	--	The qualified name of the service for which information is sought. The qualified name has the form *WSDLtargetNamespace:serviceName*. For example: http://amberpoint.com/orderService:CreditServiceSOAP.
-e	-endpointUrl	The URL of any endpoint of the service. If multiple services define endpoints at this location, information about all these services is displayed.

Name	Long Name	Description
-w	-wsdlUrl	The URL of the WSDL file that defines one or more of the service endpoints.
-V	-version	The version of the service for which information is displayed. Use the "-" string to represent the default, unnamed version.
-v	-verbose	Display additional information for each endpoint.
-s	-sphereUrl	The URL of the sphere. http://*hostname*:*port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password*. This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

The following shows the output to some sample commands.

```
btmcli showService -n OrderService

OrderService (Version -)
QName: http://amberpoint.com/orderService:OrderService

   Name: OrderServiceSOAP  Binding: SOAP/HTTP   Managed by: Monitored by Observer
   http://islam.us.oracle.com:7412/bookmart/order/OrderSerrvice


btmcli showService -qname http://schemas.amberpoint.com/sphere:SphereService

AP_Sphere_Service (Version 7.0.0.0-ExM-Hegemony)
QName: http://schemas.amberpoint.com/sphere:SphereService

   Name: AP_Sphere_Service_SOAP Binding: SOAP/HTTP
                                      Managed by: AmberPoint AP_Core_Service
   http://islam.us.oracle.com:8080/btmcentral/sphere/
```

Example

The following example uses the `showService` command to display information for the service defined by the specified wsdl:

```
btmcli showService
  -w http://jbujes-desk.edgility.com:7001/bookmart/credit/CreditService?wsdl
  -s http://localhost:8080/btmcentral/sphere -l admin:abracadabra
```

10.51 unmonitorEndpoint

Use the `unmonitorEndpoint` command to stop monitoring the specified endpoint with the monitor agent where it is registered.

Once monitoring stops, no performance measurements are recorded, no messages are logged, and no transactions are traced.

Use the `monitorEndpoint` command to re-enable monitoring.

Command Syntax

```
btmcli unmonitorEndpoint -n name | -e endpointUrl1 [[endpointUrl2] ...]
                          -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The friendly name of the endpoint to stop monitoring.
-e	-endpointUrl	A list of endpoint URLs for the endpoints you want to stop monitoring.
		If this is a web service, you can supply the actual URL to reach the endpoint. If this is another type of component, look in the Profile tab in the management console to determine the endpoint's URL.
-s	-sphereUrl	The URL of the sphere.
		http://*hostname:port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Example

The following example shows the `unmonitorEndpoint` command used to stop monitoring `MyEndpoint`.

```
btmcli unmonitorEndpoint -n MyEndpoint
                          -s http://localhost:8080/btmcentral/sphere/
                          -l Admin:abracadabra
```

10.52 unregister

Use the `unregister` command to unregister the service, WSDL, or endpoint from the sphere.

- For each service unregistered using the -w or -n options, the command outputs information about the item unregistered in the following format:

  WSDL_URL:serviceFriendlyName

- For each endpoint unregistered using the `-e` and `-interfaceQName` options, the command outputs information about the item unregistered in the following format:

  endpointUrl:endpointBinding

This command produces no output if the items to be removed are not registered with the sphere.

To unregister a specific endpoint, specify the service the endpoint belongs to using `-n` or `-qname` flags, and the endpoint to be removed using the `endpointUrl` and

-interfaceQName flags. This option is generally useful if a dependency-discovered endpoint was created incorrectly. If you do not specify the -interfaceQName flag, only dependency-discovered endpoints are removed.

When unregistering replicated services composed of endpoints from one WSDL file, specifying the service name or the WSDL URL, results in the same behavior. Unregistering replicated services composed of endpoints from multiple WSDL files, results in different behaviors, depending on how the service is identified:

- If a service is identified by its friendly or qualified name, the entire service is unregistered.

- If a WSDL is identified by the -wsdlUrl flag, only the endpoints from that WSDL are unregistered. Endpoints discovered from other WSDLs are not affected.

Once you unregister a service or endpoint, you will no longer be able to access its history. If you think you might want to re-register the service or endpoint and access its history, you will need to save its UUID (shown in the **Profile** tab).

Command Syntax

```
btmcli unregister -n serviceName | -qname qualifiedName | -w wsdlUrl1
                                              [[wsdlUrl2...]
          [-V version] [-e endpointUrl] [-interfaceQName interfaceQName] [-f]
          -s sphereUrl -l username:password
```

Name	Long Name	Description
-n	-name	The friendly name of a service to remove from the sphere.
		If the name is not unique, it must be qualified using the -V (version) flag.
-qname	--	The qualified name (WSDL targetNamespace plus service name) of the service to remove from the sphere.
		This option is useful when multiple services with different qualified names have been assigned the same friendly name in the sphere.
		If the name is not unique, it must be qualified using the -V (version) flag.
-w	-wsdlUrl	The URL for the WDSL file whose service(s) you want to unregister.
		You must specify the -w or -n flag, but not both, with this command.
-V	-version	The version of a service to remove. Must be used in conjunction with the-qname or -name options.
		Use the "-" string to represent the default, unnamed version.
-e	--	The URL of a specific endpoint; in this case the corresponding WSDL is not unregistered.
	-interfaceQName	The qualified name of the interface for a specific endpoint to unregister.
-f	-force	Unregister a service even if some of its endpoints are monitored. They will be unmonitored prior to removing the service.

Name	Long Name	Description
-s	-sphereUrl	The URL of the sphere where the WSDL is installed. http://hostname:port/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example unregisters the service in the specified wsdl.

```
btmcli unregister -w http://myProductionBox:28006/OrderService.wsdl
                  -s http://mydomain.com/btmcentral/sphere/ -l Admin:abracadabra
```

10.53 unregisterContainer

Use the `unregisterContainer` command to unregister a container previously registered with Business Transaction Management. Use this command after the container has been taken offline permanently. It is faster to use this command to unregister the container than to unregister each endpoint in it.

Physical containers must be offline long enough for the system to mark them down before you can unregister them.

This command also unregisters all the system services, service descriptors, and endpoints in the container. Any monitored endpoints in the container are no longer monitored, and measurements for these endpoints are deleted.

Command Syntax

```
btmcli unregisterContainer -baseAddress baseAddress | -fn friendlyName
                           -s sphereURL -l username:password
```

Name	Long Name	Description
-i	-baseAddress	The base address of an existing entry point in the sphere. Base address format: *scheme*://*host*:*port*/
-fn	-friendlyName	Friendly name of the container to unregister. If the friendly name is not unique, you must use the baseAddress option instead.
-s	-sphereUrl	Required. The URL of the sphere residing in the container you want to unregister. http://*hostname:port*/btmcentral/sphere/ This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.

Name	Long Name	Description
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Example

The following command unregisters the container with the friendly name `MyApps`.

```
btmcli
btmcli unregisterContainer -fn MyApps
                        -s http://localhost:8080/btmcentral/sphere/
                        -l admin:abracadabra
```

10.54 unregisterMonitor

Use the `unregisterMonitor` command to unregister the specified monitor from the sphere.

Use this command only when the monitor has been taken offline permanently. Monitors must be offline long enough for the system to mark them down before you can unregister them.

Unregistering a monitor agent has the following consequences:

- Any monitored endpoints associated with the monitor will be unmonitored.

- Measurements for unmonitored endpoints will be lost; use this command only when the monitor has been taken offline permanently.

Command Syntax

```
btmcli unregisterMonitor -baseAddress address  | -fn friendlyName
                        - s sphereUrl -l username:password
```

Name	Long Name	Description
--	-baseAddress	The base address of the monitor in the sphere. For example,
		http://localhost:8080/btmmonitor/agent/agent
-fn	-friendlyName	The friendly name of the monitor agent to unregister.
		If the friendly name is not unique, you must use the -baseAddress option instead.
-s	-sphereUrl	The URL of the sphere where the WSDL is installed.
		http://hostname:port/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.

Name	Long Name	Description
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role. See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials. You can encrypt passwords using the `encryptPassword` command.

Example

The following example unregisters the monitor with the specified base address.

```
btmcli unregisterMonitor -base address
                            http://localhost:8080/btmmonitor/agent/agent
          -s http://mydomain.com/btmcentral/sphere/ -l admin:abracadabra
```

10.55 updateProfileData

Use the `updateProfileData` command to update profile attribute data for the specified set of objects. This command only updates existing objects; if it cannot find the specified objects, it returns an error.

This command is useful in two distinct cases:

- You want to update a collection of objects with a common set of attribute values. For example, you want to set the application group for a set of services to the same value.

- You want to set the attribute values for a collection of objects to different values. For example, you want to import data obtained from an external data source.

See the discussion below for additional information.

Command Syntax

```
btmcli updateProfileData -t type [-query filterQuery]
        [-val name1:val1 [[name2:val2]...]]] | [-profileData profileData]
      -s sphereUrl -l username:password
```

Name	Long Name	Description
-t	-type	The type of objects to update.
-query	-filterQuery	Selects the set of objects that match the query. Use the Filter tool to construct AQL queries, and then copy and paste these into the command line.
-val	-attributeValues	A list of *name:value* pairs used to update the selected objects. Each entry in the list specifies the name of an attribute and its value. If you provide this argument, you must also provide the -query argument. Look at section "Attribute Names" for a complete list of commonly used attribute names and object types.

Name	Long Name	Description
--	-profileData	Specifies the name of a file containing the profile data to update. This file can have one of two formats, depending on the flags specified.
		If you provide the -query argument, the file should be a Java properties file where each entry is an attribute name and its associated value.
		If there is no -query argument, this file should be a text-delimited file in csv form where the first row specifies the attributes to update, and subsequent rows provide values for a series of object instances. In order to identify the objects to be updated, the key attributes must be included (either the object id or the attributes that make up the type's natural key).
-s	-sphereUrl	The URL of the sphere.
		http://*hostname*:*port*/btmcentral/sphere/
		This flag is not required if you have set the AP_SPHERE_URL environment variable for your sphere.
-l	-userLogin	The username and password associated with the sphere, in the format: *username:password.* This set of credentials must belong to a user in the *btmadmin* role.
		See Section 10.1.3, "Security Options in Accessing CLI Commands" for information on furnishing login credentials.
		You can encrypt passwords using the `encryptPassword` command.

Discussion

This command has essentially three forms of use:

- In order to update one or more attributes to the same values for a set of objects use the following form:

  updateProfileData -t *myType* -query *myQuery* -val *attr1=valueattr2=value* ...

  or the form

  updateProfileData -t *myType* -query *myQuery* -profileData *javaPropFile*

- In order to perform a bulk update of a set of objects where the attribute values for each object are different (such as when updating from an external database), use the form:

updateProfileData -t *myType* -profileData *CSVFile_with_1_row_per_Object_to_update*

Example

The following example shows the `updateProfile` command.

```
btmcli updateProfileData -t endpoint -profileData MyJavaPropertyFile
        -s http://localhost:8080/btmcentral/sphere/ -l Admin:abracadabra
```

11

Utilities

This chapter describes the Business Transaction Management utilities. You use these utilities to connect to and manage a database, to manage system service deployment log files, and to display information about the Business Transaction Management product. This chapter includes the following sections:

- About Command Line Utilities
- datastoreUtil utility
- logMerger utility
- showProductInfo utility

11.1 About Command Line Utilities

Business Transaction Management includes a set of command line utilities that allow you to perform certain tasks from the command line.

For Java platforms, the utilities are located in: [*install_dir*]/tools

On Windows platforms, utilities use the JRE bundled with Business Transaction Management.

On Unix-like systems, you must specify a JRE to use by doing one of the following:

- set the JAVA_HOME environment variable and remove the directory

  *BTM*_install_dir/server/jre directory

- replace *BTM*_install_dir/server/jre with a link to a valid JRE directory

The following table lists Business Transaction Management utilities.

Name	Description
datastoreUtil	Allows you to connect to a database. Once connected, you can do the following: ■ generate the DDL of the specified schema definition within the connected database ■ check the status of the specified schema within the connected database ■ create the specified schema within the connected database ■ upgrade the specified schema within the connected database ■ drop the entire specified schema from the connected database ■ save the database connection information into a file for easy reconnection to the database at a later time ■ display the owner of the database instance (service name and UUID) ■ reset ownership of the database instance by removing knowledge of its current owner ■ enter the database explore mode (commands to search and display data)
logMerger	Merges Business Transaction Management system service deployment log files (logdir) to consolidate analysis or archiving.
showProductInfo	Outputs information about the Business Transaction Management product.

11.2 datastoreUtil utility

The most common use of the datastoreUtil utility is to assist in the installation of the Business Transaction Management product. During configuration, the system automatically creates the appropriate database tables for database users for the sphere, performance, and transaction databases unless you choose to create them beforehand with the `datastoreUtil` utility.

By using the `generateSchema` command, you can create DDL for the targeted database environment. The output of the `generateSchema` command can be used as input (with the appropriate database utility) to create the necessary tables and views for the Business Transaction Management product.

Use the `datastoreUtil` utility to connect to a database. Once connected, you can do the following:

■ generate the DDL of the specified schema definition within the connected database

■ check the status of the specified schema within the connected database

■ create the specified schema within the connected database

■ upgrade the specified schema within the connected database

■ drop the entire specified schema from the connected database

■ save the database connection information into a file for easy reconnection to the database at a later time

■ display the owner of the database instance (service name and UUID)

■ reset ownership of the database instance by removing knowledge of its current owner

- dump the definition of the connected database to a text file containing SQL dml statements in the given database format

- enter the database explore mode (commands to search and display data)

Invoking the datastoreUtil Utility

To invoke the `datastoreUtil` utility, navigate to the tools directory and use the following from the command line:

- **Windows**:datastoreUtil.bat

- **Unix**: datastoreUtil.sh

After starting the utility, you can call the commands described in the next subsection. For commands that have multiple arguments, you must call the arguments in the order described.

Use the `exit` command to exit the utility.

11.2.1 Commands

`datastoreUtil` commands include the following:

Command	Command	Command
help	generateSchema	checkOwner
connect	createSchema	resetOwner
saveConnection	checkSchema	dumpDataBase
close	upgradeSchema	--
exit	dropSchema	--

help Command

Use the following syntax to get help:

```
help | help command
```

Use the `help` command to view help for all commands, or enter a command name to receive help for a single command

connect Command

Connect to a database using the user-specified connection information

```
connect databaseType|filename
```

databaseType - specify `oracle`. This is the only supported value.

filename - name of a file as specified by the `saveConnection` command

Use the `connect` command to enter database connection information and connect to the database. You must have the following information for the database to which you want to connect:

- driver name

- username

- password

- URL connection string

Once connected, you might issue the `saveConnection` command to save the connection information within a file. The next time you want to connect to the same database, you can provide the file name with the `connect` command. If you provide the database type, the utility automatically selects the corresponding default driver.

close Command

Close a connection previously opened with the `connect` command. This command takes no parameters.

generateSchema Command

Generate the DDL of the specified schema definition. You do not have to be connected to a database to call this command.

Alternatively, you can use `generate` for this command.

```
generateSchema   schemaType databaseType
                 [[directory] targetSchema] -partition|-nopartition

generate schemaType databaseType [[directory]
                        targetSchema] -partition|-nopartition
```

schemaType - one of the known schema types:

- `sphere` - schema for the sphere database (the sphereDB user)
- `exm` - schema for the transaction database (the transactionDB user)
- `performance` - schema for the performance database (the measurementDB user)
- `monitorgroup` - schema for a monitor group
- `msglog` - schema for the system message log

databaseType - one of the supported database types:

- `oracle`

directory - location to generate the DDL file (defaults to the local directory)

targetSchema -target the generated schema for a specific user, for example sphereDB, transactionDB, or measurementDB.

For example, you have a high-level administrative permissions on the database, but you want to create the schema for someone with much lower-level permissions. You would use something like the following example:

```
generateSchema sphere oracle \temp sphereUser
```

`-partition | -nopartition`

This flag is required if your specified *schemaType* is `performance` or `monitorgroup`. If your specified *schemaType* is any other value, this flag is not required and is ignored if you use it.

If you are using Oracle Enterprise Edition, you can create a `performance` or `monitorgroup` schema that takes advantage of Oracle's partitioning feature by specifying the `-partition` flag. If you do not want to take advantage of this feature or if your Oracle edition does not provide the partitioning feature, you must specify `-nopartition` (if you are creating a `performance` or `monitorgroup` schema).

checkSchema Command

Check the status of the specified schema within the connected database

Alternatively, you can use `check` for this command as the syntax diagram shows.

checkSchema *schemaType*

check *schemaType*

schemaType - one of the known schema types:

- `sphere` - schema for the sphere database (the sphereDB user)

- `exm` - schema for the transaction database (the transactionDB user)

- `performance` - schema for the performance database (the measurementDB user)

- `monitorgroup` - schema for a monitor group

- `msglog` - schema for the system message log

Use the `checkSchema` command to check for the specified schema type within a connected database. You must successfully execute the connect command before executing the `checkSchema` command. The `checkSchema` command displays the status of the schema as found in the database. If the status of the database schema is not up-to-date, this command displays the differences found, and the DDL you must apply to upgrade the database schema.

This command does not change the database schema. If you wish to automatically upgrade the database schema from this command line utility, use the command `upgradeSchema`.

createSchema Command

Create the specified schema within the connected database. You must successfully execute the `connect` command before executing the `createSchema` command.

Alternatively, you can use `create` for this command.

createSchema *schemaType* -partition | -nopartition

create *schemaType* -partition | -nopartition

schemaType - one of the known schema types:

- `sphere` - schema for the sphere database (the sphereDB user)

- `exm` - schema for the transaction database (the transactionDB user)

- `performance` - schema for the performance database (the measurementDB user)

- `monitorgroup` - schema for a monitor group

- `msglog` - schema for the system message log

-partition | -nopartition

This flag is required if your specified *schemaType* is `performance` or `monitorgroup`. If your specified *schemaType* is any other value, this flag is not required and is ignored if you use it.

If you are using Oracle Enterprise Edition, you can create a `performance` or `monitorgroup` schema that takes advantage of Oracle's partitioning feature by specifying the `-partition` flag. If you do not want to take advantage of this feature or if your Oracle edition does not provide the partitioning feature, you must specify `-nopartition` (if you are creating a performance or monitorgroup schema).

You may want to use the commands `checkSchema` and `upgradeSchema` before or instead of the `createSchema` command.

upgradeSchema Command

upgrade the specified schema within the connected database

Alternatively, you can use upgrade for this command.

upgradeSchema *schemaType*

upgrade *schemaType*

schemaType - one of the known schema types:

- sphere - schema for the sphere database (the sphereDB user)
- exm - schema for the transaction database (the transactionDB user)
- performance - schema for the performance database (the measurementDB user)
- monitorgroup - schema for a monitor group
- msglog - schema for the system message log

Use the upgradeSchema command to upgrade the specified schema of the connected database. You must successfully execute the connect command before executing the upgradeSchema command. The upgradeSchema command will add any missing table, column, or index to the database.

You may want to execute the checkSchema command before executing the upgradeSchema command.

Note that the upgradeSchema command only adds missing elements; it does not remove anything. You cannot roll back the upgradeSchema command.

dropSchema Command

Drop the entire specified schema from the connected database

Alternatively, you can use drop for this command.

dropSchema *schemaType*

drop *schemaType*

schemaType - one of the known schema types:

- sphere - schema for the sphere database (the sphereDB user)
- exm - schema for the transaction database (the transactionDB user)
- performance - schema for the performance database (the measurementDB user)
- monitorgroup - schema for a monitor group
- msglog - schema for the system message log

Use the dropSchema command to drop the entire schema from the connected database. You must successfully execute the connect command before executing the dropSchema command.

Warning: the dropSchema command removes all tables and any data stored within these tables. If you have any unsaved data you want to keep from the connected database, export or save the data before executing this command. You cannot roll back the dropSchema command.

saveConnection Command

Save the database connection information to a file

Alternatively, you can use save for this command.

```
saveConnection filename
```

```
save filename
```

filename - name of the file where you want to save user, password, url, and driver values.

Use the `saveConnection` command to save user, password, url, and driver values to a file.

Once connected to a database, you might issue the `saveConnection` command to save the connection information to a file. You can later reconnect to the same database by providing the file name with the connect command. Make sure to remove any unused files as they contain connection information.

checkOwner Command

Displays the owner of the database instance (service name and UUID)

Alternatively, you can use `owner` for this command.

```
checkOwner
```

```
owner
```

Once connected to a database, you might issue the `checkOwner` command to display the service UUID and name that owns the database instance. Only a service owner can connect to a specific database instance. To reset ownership, use the `resetOwner` command.

resetOwner Command

Reset ownership of the database instance by removing knowledge of its current owner

Alternatively, you can use `reset` for this command.

```
resetOwner
```

```
reset
```

Use the `resetOwner` command to remove the service ownership information associated with a specific database. You must successfully execute the `connect` command before executing the `resetOwner` command. Once reset, the next service that accesses the database takes ownership of the database instance.

dumpDatabase

Export the contents of the given database to a text file containing SQL dml statements in the given database format

Alternatively, you can use `dump` for this command.

A command to connect must have been successfully executed before this command can be issued.

```
dumpDatabase schemaType databaseType [directory]
```

```
dump schemaType databaseType [directory]
```

Use the `dumpDatabase` command to export the data contained in the specified schema to a text file containing SQL dml statements that you can use to import into another

database. You specify the database type and the text file is formatted into SQL statements supported by that database type.

Note: If your database contains clob fields, they are truncated to 4000 characters in the resulting text file.

schemaType - one of the known schema types:

- `sphere` - schema for the sphere database (the sphereDB user)
- `exm` - schema for the transaction database (the transactionDB user)
- `performance` - schema for the performance database (the measurementDB user)
- `monitorgroup` - schema for a monitor group
- `msglog` - schema for the system message log

databaseType - one of the supported database types:

- oracle

directory - where the SQL file is generated. The default location is the local directory.

exit Command

Use the following command to exit the `datastoreUtil` utility.

```
exit
```

11.3 logMerger utility

Use the `logMerger` utility to merge Business Transaction Management system service log files (logdir) for analysis or archiving. You can use the `-config` flag to load the utility options from a named log merger configuration file. Command line options will override options defined in a configuration file.

The log merger configuration file is described after the discussion of the `logMerger` utility.

Command Syntax

```
logMerger -config config_file_name |-dir directory|-url url
          [-username username -password password]
          [-output output_file_name] [-overwrite true|false]
          [-format format_pattern]
          [[-from start_time -to end_time]|[-last number_of_hours]]
          [-loggerName logger_name] [-loggerLevel level|level+]
          [-className class_name] [-methodName method_name]
          [-userName user_name] [-V]
```

Name	Description
-config	Load options from the named configuration file. Command line options override options defined in the configuration file. For an example of a `logMerger` configuration file, see Section 11.3.1, "logMerger Configuration File."
-dir	Directory that contains Business Transaction Management logdir directory you want to merge. The `logMerger` utility searches all subdirectories of the specified directory for logdir directories. You can provide multiple -dir directory options. Each directory must be a separate entry.

Name	Description
-url	URL of the running Business Transaction Management service whose logdir you want to merge, with the inclusion of the user credentials required to access the service. The `logMerger` utility will communicate with the service on this URL using the Business Transaction Management user credentials specified by the `-username` and `-password` options. You can provide multiple -url service URL options, but all services must have the same username and password credentials. Each URL must be a separate entry. **Note:** If you want to merge logging from multiple service URLs that require different username and password credentials for access to each service, you must use a logMerger configuration file.
-username	The username credential for access to the Business Transaction Management service associated with the url option. **Note:** The -username flag is different from the -userName flag described below; the two flag names are case-sensitive.
-password	The password credential for access to the Business Transaction Management service associated with the url option.
-output	Specifies where to save the merged result. The logMerger utility will exit without overwriting if the output file already exists, unless the -overwrite option has a value of "true". If no output option is specified, the resulting output file with be written to the directory from which the logMerger utility was called, with a default value of "logmerger_<*date*>_<*time*>.log".
-overwrite	Specify `true` to overwrite existing output file. Default is `false`.
-format	Specify the output format of each log entry in the merged result. Possible format tokens include log entry properties: ■ `time` ■ `methodName` ■ `loggerName` ■ `loggerLevel` ■ `className` ■ `userName` and four more special tokens: ■ `message` - the actual log message ■ `logName` - log identifier which is the 'logName' attribute found in the 'logHeader' element of each log file ■ `eol` - end of line character to separate text ■ `tab` - tab character to separate text Each format token is separated by a plus sign '+'. There is always an End of Line character after each log entry is output. Default format pattern is: `time+logName+eol+loggerLevel+message`.
-from	Merge log entries with timestamp later than *start_time*. Absence of this option means no start time limit unless -last option is specified. Use time format of "MM/dd/yy HH:mm:ss". For example: "10/05/06 00:00:01"

Name	Description
-to	Merge log entries with timestamp earlier than *end_time*.
	Absence of this option means no end time limit.
	Use time format of "MM/dd/yy HH:mm:ss". For example: "10/05/06 17:21:57"
-last	Merge log entries from last number of hours.
	This option is ignored if -from or -to is specified.
-loggerName	Merge only log entries generated by the logger with *logger_name*.
-loggerLevel	If only level is specified, then merge log entries with the specified logger level.
	If the level+ is specified, then merge log entries with any level equal to or higher than the specified level.
	The following `loggerLevel` attribute values are listed in order from highest to lowest:
	SEVERE
	WARNING
	INFO
	CONFIG
	FINE
	FINER
	FINEST
-className	Merge only log entries generated by the class with *class_name*.
-methodName	Merge only log entries generated by the method with *method_name*.
-userName	Merge only log entries generated by the user with *user_name*.
	Note: The `-userName` flag is different from the `-username` flag described as a sub -flag of the -url flag above; the two flag names are case-sensitive.
-V	Display version information for the utility and exit.

Examples

1. The following example collects log entries under the server `\BTM_INSTALL_DIR\btmstorage` directory (and all sub-directories below it), sends them to an output file named `BTMLoggingOutputTo10_05_06`, the overwrite option is set to `true`, all entries with a loggerLevel of `INFO` or higher (`WARNING`, `SEVERE`) will be collected, with multiple format options, and a date and time range over two-and-one-half days:

    ```
    logMerger -dir "C:\BTM_INSTALL_DIR\btmstorage"
            -output BTMloggingOutputTo10_05_06 -overwrite true
            -loggerLevel INFO+ -format time+loggerName+userName+message+eol
            -from "10/03/06 00:00:01" -to "10/05/06 12:00:00"
    ```

2. The following example collects log entries under the local machine's directories `\BTM_INSTALL_DIR\btmstorage\btmui` and `BTM_INSTALL_DIR\btmstorage\btmtransaction`, sends the entries to an output file named `BTMuiNtransactionLoggingOutLast24`, the overwrite option is set to true, all entries with a loggerLevel of `INFO` will be collected, with multiple format options, the utility collects all logging entries generated by the action of a user with the name `SalesOpsManager` that have occurred over the last 24 hours.

    ```
    logMerger -dir "C:\BTM_INSTALL_DIR\btmstorage\btmui"
    ```

```
            -dir "C:\BTM_INSTALL_DIR\btmstorage\btmtransaction"
            -output BTMuiNtransactionLoggingOutLast24 -overwrite true
            -loggerLevel INFO -format time+userName+messgae
          -userName SalesOpsManager -last 24
```

3. The following example collects log entries from three URLs:

```
http://remoteServer1:8080/btmcentral/sphere
http://remoteServer1:8080/btmcentral/sphere
http://remoteServer1:8080/btmcentral/sphere
```

The user and password credentials to access these service URLs is the same (otherwise, you must use a configuration file). The log entries are sent to an output file named `remoteServiceOutputFrom10_13_06`, all entries with a loggerLevel of `INFO` or higher (`WARNING`,`SEVERE`) are collected, with multiple format options, and a date range from one second after midnight on 10/13/06.

```
logMerger -url "http://remoteServer1:8080/btmcentral/sphere"
          -url "http://remoteServer2:8080/btmcentral/container"
          -url "http://remoteServer1:8080/btmcentral/agent"
          -username BTMAdminUser -password BTMAdminUserPwd
          -output remoteServiceOutputFrom10_13_06
          -loggerLevel INFO+ -format time+message+eol -from "00:00:01 10/13/06"
```

4. The following example uses settings from within a user-defined LogMerger configuration file named `myLogMergerConfigFile.xml`. Use this option when you frequently use `logMerger` to collect the same type of information. You can add options to the command line to overwrite the options defined in the configuration file.

```
logMerger -config myLogMergerConfigFile.xml
```

11.3.1 logMerger Configuration File

The following text contains the formatting and content for a sample log merger configuration file. Options specified in the file will be overridden by command line options.

If you want to encrypt the passwords for accessing remote service URLs via the configuration file, you can use the `encryptPassword` command.

The attributes defined in the configuration file, which are described in the following subsections, are enclosed in the following:

```
<ap:logmerger xmlns:ap="http://namespace.amberpoint.com/amf" debug="false">
.
.
.
</ap:logmerger>
```

Sources Attribute Example and Discussion

```
<ap:sources>
 <ap:source dir="C:/Program Files/AmberPoint/SOAManagementSystem/
                                              server/amberpoint"/>
 <ap:source url="http://remoteServerHost1:7001/apcentral/sphere"
                         username="SMSUser1" password="SMSUserPwd1"/>
 <ap:source url="http://remoteServerHost2:7001/apcontainer/container"
                         username="SMSUser2" password="SMSUserPwd2"/>
```

```
    <ap:source url="http://remoteServerHost3:7001/eProxy/agent/agent"
                          username="SMSUser3" password="SMSUserPwd3"/>
  <ap:source url="http://remoteServerHost1:8080/apcentral/sphere"
                          username="ALSMUser1" password="ALSMUserPwd1"/>
  </ap:sources>
```

You can provide multiple source `dir` entries. Each source directory must be a separate entry.

`dir`: Directory that contains the Business Transaction Management logdir directory you want to merge. The `logMerger` utility searches all subdirectories of the specified directory for logdir directories.

You can provide multiple source URL entries. Each source URL must be a separate entry.

`url`: URL of running Business Transaction Management service. This utility will communicate with the service on this URL using the user credentials specified by the `-username` and `-password` options.

`username`: username to access the Business Transaction Management service

`password`: password to access the Business Transaction Management service

Filter Attribute Example and Discussion
```
<ap:filter
  from="10/02/06 09:00:00"
  to="10/02/06 11:59:00"
  last="24"
  loggerName="com.amberpoint.services.lifecycle.initialize"
  loggerLevel="INFO+"
  className="com.amberpoint.util.soa.services.urservice.UrServiceImpl"
  methodName="createServices"
  userName="BTMdministrator"
```

The filter attributes are used to query the log history for messages that contain an exact match of all attributes.

For example, filter attributes of last="24" loggerLevel="INFO" will merge INFO messages logged within the last 24 hours.

If no filter options are specified, then all log messages will be included.

`from`: log entries with a timestamp later than "from" will be included. Absence of this attribute means no start time limit unless "last" attribute is specified.

Use time format of "MM/dd/yy HH:mm:ss". E.g. from="12/25/05 10:34:25"

`to`: log entries with a timestamp earlier than "to" will be included. Absence of this attribute means no end time limit.

Use time format of "MM/dd/yy HH:mm:ss". E.g. to="12/25/05 10:34:25"

`last`: last number of hours of log entries will be included. this attribute is ignored if "from" or "to" is specified.

`loggerName`: only log entries generated by the named logger attribute will be included.

`loggerLevel`: this attribute accepts values in two formats: level or level+.

If level is specified, only log entries with the specified logger level will be included.

If level+ is specified, log entries with a level equal to or higher than the specified level will be included.

The following loggerLevel attribute values are listed in order from highest to lowest:

SEVERE

WARNING

INFO

CONFIG

FINE

FINER

FINEST

`className`: only log entries generated by the specified class will be included.

`methodName`: only log entries generated by the specified method will be included.

`userName`: only log entries generated by the specified user will be included.

-->

Output File Example and Discussion

```
<
<ap:output file="c:\temp\testmerger.log" overwrite="true"
                      format="time+logName+loggerName+eol+loggerLevel+message"/>
```

`file`: where the merged log file will be saved.

The LogMerger utility will stop processing if the output file already exists, unless the overwrite attribute has a value of "true".

Absence of the `file` attribute will save the merged log file to a new file named `logmerger_<date>_<time>.log`.

`overwrite`: allows the utility to overwrite the file specified by the "file" attribute if it already exists. The default value of overwrite is "false".

`format`: defines the output format of each log entry in the merged result.

Possible format tokens include all log entry properties (attributes on logEntry node in log file).

Examples of log entry properties with values are below:

```
<logEntry
     entryId="1127753975442:0"
     time="Mon Sep 26 09:59:35 PDT 2005"
     entryType="1"
     dataSize="102"
     loggerName="com.amberpoint.agent.backplane.servlet.ServletDriver"
     loggerLevel="INFO"
     methodName="findOldStorageDirectory"
     className="com.amberpoint.agent.backplane.servlet.ServletDriver">
```

There are four additional special tokens:

- `message`: the actual log message

- `logName`: name of the log file that is stored in <logHeader>. The logHeader is used to identify the service that generates the log message.

- `eol`: end of line character to separate text

■ `tab`: tab character to separate text

Each format token must be separated by a plus sign.

The LogMerger utility inserts an End of Line character after each log entry that is written to the output file.

11.4 showProductInfo utility

Use the `showProductInfo` utility to collect information about the Business Transaction Management release you have installed on each machine. You should also check this information and report it when dispatching support requests.

The `showProductInfo` utility reports the following information about the installed product:

■ release number, build number, and build date associated with the installed product

■ location of the uninstaller for the installed product

■ URL for Business Transaction Management support

■ URL for the product documentation

■ URL to main Business Transaction Management web site

12

Administration of Business Transaction Management

This chapter provides information to help you administer Business Transaction Management and includes the following sections:

- Observers
- Persistent Data
- Business Transaction Management System Security
- Backing up and Restoring Business Transaction Management
- Migrating Data
- Setting up Load Balancers
- Resolving Discovery Issues
- Miscellaneous Administration Topics

12.1 Observers

This section provides information to help you administer Business Transaction Management observers and includes the following subsections:

- About Observers
- Configuring the Observer and Monitor
- Logging Observer Errors and Debugging Information

12.1.1 About Observers

Observers are Business Transaction Management components that you install into the application server of business applications you want to monitor. The observers monitor messages and calls between the components of your business applications.

Observers contain one or more subcomponents called *probes*. Each probe provides the observer with the capability of monitoring a particular type of business component. The monitoring capabilities of an observer are, therefore, dependent on the collection of probes that it contains.

The following table lists the types of observers provided by Business Transaction Management, the probes contained within each observer, and the monitoring capabilities conferred on the observer by each of the probes.

Table 12–1   Available observers, the probes they contain, and the types of components they monitor.

Observer	Probe	Components Monitored
JavaEE	EJB	Enterprise JavaBean (EJB)
-	JAVA	Java (monitors local Java method calls)
-	JAXRPC	JAX-RPC (includes monitoring of JMS traffic that uses the JAX-RPC API)
-	JAX-RS	JAX-RS (monitors RESTful applications, see JSR 339)
-	JAXWS	JAX-WS (includes monitoring of JMS traffic that uses the JAX-WS API)
-	JDBC	JDBC (monitors Java database calls)
-	JMS	JMS (monitors traffic that uses the JMS API)
-	RMI	Remote method invocation (RMI)
-	WEB_APP	Java servlet application
OSB	OSB	Oracle Service Bus Proxy and Business Services
Oracle SOA Suite	SOA_ADAPTER	Oracle SOA Suite Adapter (this probe is provided only in the observer for Oracle Service Bus 10g)
-	SOA_BIZRULE	Oracle SOA Suite Business Rule
-	SOA_B2B	SOA B2B (Business to business)
-	SOA_BPEL	Oracle SOA Suite Business Process Execution language (BPEL)
-	SOA_BPMN	Oracle SOA Suite Business Process Modelling Notation (BPMN)
-	SOA_CALLBACK	Callback requests of asynchronous calls
-	SOA_DIRECT	Direct binding calls between SOA composites and between SOA and OSB
-	SOA_EDN	Oracle SOA Suite Event Delivery Network
-	SOA_JCA	JCA adapters including AQ, Database, File, FTP, JMS, MQ Series, Socket, and Oracle Applications
-	SOA_MEDIATOR	Oracle SOA Suite Mediator
-	SOA_REST	Oracle SOA Suite REST
-	SOA_SPRING	Oracle SOA Suite Spring Bean
-	SOA_WORKFLOW	Oracle SOA Human Workflow components and notifications
-	SOA_WS	Oracle SOA Suite web service (including Human Workflow web services)
-	SOA_WSA	Oracle SOA Suite web service adapter
-	WEB_APP	Java servlet application
Oracle Fusion Applications (supports ADF-UI, ADF-BC and SOA deployments)	ESS	Oracle Enterprise Scheduling Service

Table 12–1 (Cont.) Available observers, the probes they contain, and the types of components they monitor.

Observer	Probe	Components Monitored
-	All probes found in the JavaEE observer except for the JDBC probe	Refer to the entry for the JavaEE observer
-	All probes found in the SOA Suite observer	Refer to the entry for the SOA Suite observer
Universal	All probes found in the JavaEE, OSB, Oracle SOA Suite, and Oracle Fusion Applications observers	Refer to the entries for the JavaEE, OSB, Oracle SOA Suite, and Oracle Fusion Applications observers.
WCF	WCF	Microsoft WCF services
Oracle Enterprise Gateway (OEG)	OEG	Web services fronted by an OEG Web Services Proxy

> **Note:** For a complete and up-to-date list of the types of services and components that Business Transaction Management can discover and monitor, refer to the Business Transaction Management Certification Matrix. You can locate this document by searching for "BTM certification" online at `http://support.oracle.com`.

A single observer installation can monitor any number of components that are running in the application server, as long as the observer contains the appropriate probes.

Observers communicate with the Business Transaction Management sphere by way of another Business Transaction Management component called the *monitor*. One of the jobs of the monitor is to distribute configurations to the observers. When an observer starts up, it contacts the monitor and obtains a configuration. The observer periodically polls the monitor for updates to its configuration.

The observer configuration is generated from an Observer Communication policy. By default, a preconfigured Observer Communication policy is applied to all monitors (this default policy is named **Observer Communication Policy - Default**). This policy configures the monitors to which it is applied and also provides those monitors with an observer configuration that, by default, they distribute to all of their associated observers. You can edit this default policy and/or apply your own.

Once running, the observers measure various aspects of your business applications' message and/or call flow, such as throughput, fault count, and response time (for a complete list of measurements, see Chapter 6.3, "About Instruments"). The observers periodically send these measurements to the monitor for analysis and eventual storage in a database, as shown in the following diagram:

Figure 12–1   Example of deployed observer showing probes.
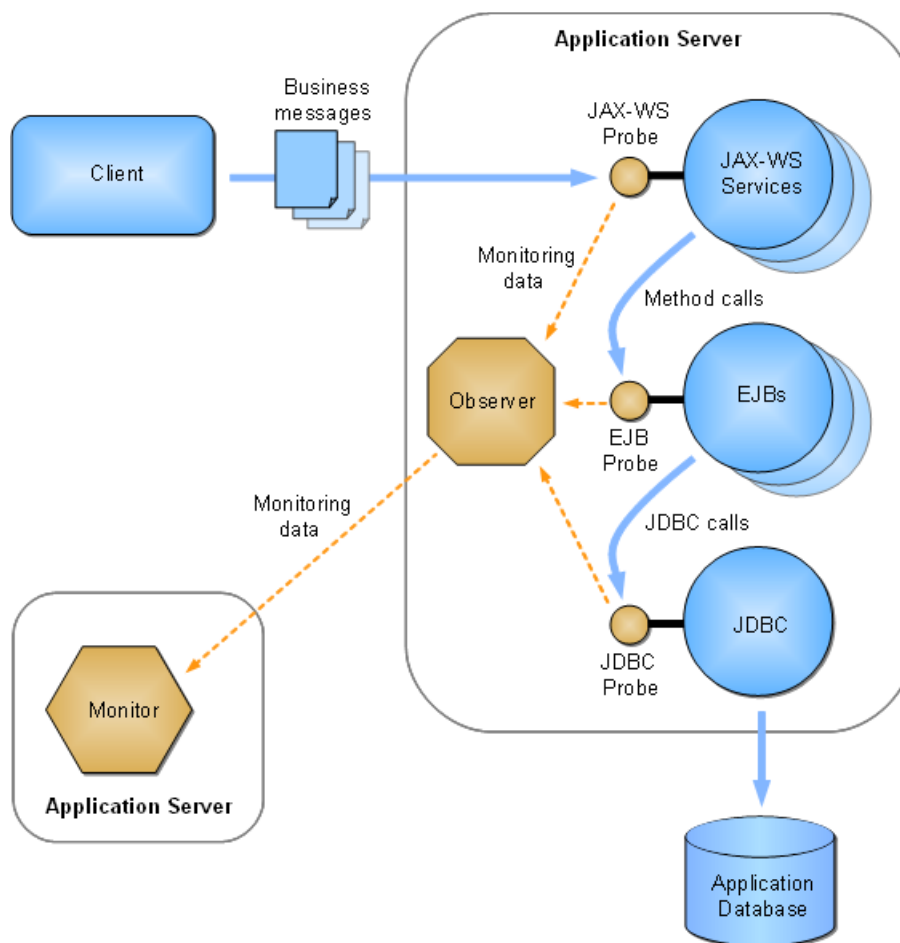
If configured to do so, the observers convert the various types of messages and calls into standardized XML-formatted messages for the purposes of message logging and further analysis. The observers forward these messages to the monitor. Note that these messages are copies, and that the original messages/calls are never altered or redirected.

Observers are installed into your business component's application server, and clients continue to access the business component in the same way as before the observer was installed. You can install any number of monitors and any number of associated observers per monitor, but observers are never installed in the application server hosting the Business Transaction Management central servers or monitors.

For an overview of the entire Business Transaction Management system, see Section 1.3, "Architectural Overview." For information about scaling up your monitoring system by replicating the monitors, refer to the *Business Transaction Management Installation Guide*.

12.1.2 Configuring the Observer and Monitor

The Observer Communication policy sets up communication between observers and a monitor or monitor group. By default, this policy configures both monitors and observers by:

- setting up the observer-to-monitor communication channel

- configuring the runtime settings for the observer

Prior to release 12.1.0.4, you could apply only a single Observer Communication policy to any one monitor, which meant that all observers associated with the same monitor would receive the same configuration. To provide greater flexibility in configuring observers, release 12.1.0.4 introduced the ability to apply multiple Observer Communication policies to a single monitor and to then target each of these policies at a different observer or set of observers. In such a scenario, you designate one (and only one) policy as being used for configuring the monitor, and you designate all of the remaining policies as being used only for configuring observers. The policy that configures the monitor can also be used for configuring observers, if you so desire. For more information on this topic, see Section 12.1.2.7, "Targeting Observers."

By default, a preconfigured Observer Communication policy is applied to all monitors registered in the system. This default policy is preconfigured to target all associated observers. If necessary, you can edit this default policy and/or create new policies.

To edit the default Observer Communication policy instance:

1. Select **Administration > System Policies** in the Navigator.

2. Select **Observer Communication Policy - Default** in the summary area (in previous releases, this policy is named **Default Observer Communication Policy**).

3. Choose **Modify > Edit Definition for Observer Communication Policy - Default**.

To create a new instance of an Observer Communication policy:

Choose **Admin > Create System Policy > Observer Communication**.

12.1.2.1 Common Tasks

The following are common configuration tasks you can perform using this policy:

- Activating and Deactivating Probes

- Adding Probes

- Setting up the Observer-to-Monitor Communication Channel

- Configuring SSL on the Observation Message Flow

- Configuring Observer Authentication

- Targeting Observers

- Enabling/Disabling Drilldowns from Real User Experience Insight

- Request Monitoring and Operation Modeling for the WEB_APP Probe

- You can find information related to the following and other advanced tasks in the Advanced Settings Field Reference table in Section 12.1.2.11:

  - Logging observed messages and observer errors (see the **Observer Troubleshooting** entry on page 12-26)

  - Adjusting message size and queue behavior on the observer (see the **Observer Message** entry on page 12-27)

  - Adjusting message size and queue behavior on the monitor (see the **Monitor Message Queue** entry on page 12-28)

  - Controlling how SOA, OSB, JMS, and local EJB components are modeled (see the **Model Configuration** entry on page 12-30)

– Enabling JDBC Summary Mode, in which observations of related JDBC calls are aggregated and sent to the monitor as a single observation message (see the **JDBC Summary** entry on page 12-31)

12.1.2.2 Activating and Deactivating Probes

Observers contain different types of probes for monitoring the various types of components that make up your business applications. You can use this policy to individually activate or deactivate probes that are installed on your system (any particular probe is considered to be installed if it is contained within an observer that is installed on your system). By default, all probes in a newly created policy except JAVA and RMI are activated.

> **Notes:**   The JAVA probe monitors local Java calls, which in most cases is not needed and can be distracting because of the typically large number of local Java calls that occur. In order to use the JAVA probe, you must first deploy and configure it. For information about deploying and configuring the JAVA probe, enter a service request at My Oracle Support (support.oracle.com).
>
> In most situations, you should leave the RMI probe deactivated. Most applications utilize RMI by way of higher level APIs, such as JAX-RPC, JAX-WS, EJB, and JMS. In such cases, it is better to activate only the probes for these higher-level components. However, if your application makes RMI calls directly you might want to activate the RMI probe.

There is no need to explicitly deactivate probes that are not installed—neither for the sake of performance nor for any other reason (uninstalled probes are inherently not activated). The only reason to deactivate a probe is if: (1) the probe is installed, AND (2) you do not want to monitor the type of business component the probe monitors. Furthermore, you must deactivate (or activate) the SOA Suite probes as a group (the SOA Suite probes are SOA_ADAPTER, SOA_B2B, SOA_BIZRULE, SOA_BPEL, SOA_BPMN, SOA_CALLBACK, SOA_EDN, SOA_JCA, SOA_MEDIATOR, SOA_REST, SOA_SPRING, SOA_WS, and SOA_WSA).

The **Active Probes** section of the policy provides an **Enable Discovery** and **Monitor Upon Discovery** checkbox for each type of probe.

Select the **Enable Discovery** checkbox to activate the discovery mechanism for the associated component type. Components of that type are then discovered and displayed in the Management Console the next time they receive a message or call.

Select the **Monitor Upon Discovery** checkbox for a component type if you want to immediately begin monitoring components of that type as they are discovered.

> **Note:**   If you enable discovery but not monitoring and then later edit the policy and enable monitoring, the system will not begin monitoring previously discovered components. The system will begin monitoring only the components discovered after you enable monitoring. For information on enabling monitoring for previously discovered components, see Section 12.8.4, "Start and Stop Monitoring of Endpoints."

12.1.2.3 Adding Probes

After upgrading Business Transaction Management, you might have new types of probes available for your monitoring purposes. However, the upgrade process does not automatically add these new probes to your existing Observer Communication policies. In order to use a new probe in an existing policy, you must manually add it to the policy as follows:

1. Select **Administration > System Policies** in the Navigator.

2. Select your policy in the summary area.

3. Choose **Modify > Edit Definition for** *My_Policy*, where *My_Policy* is the name of your policy.

4. Scroll to the bottom of the **Active Probes** section and click **[add probe]**.

   The cursor should now be inserted into an empty text field.

5. With the cursor already inserted into the empty text field, click into the text field.

   A drop list opens that contains the names of all probe types.

6. Choose the probe type that you want to add.

7. Edit the settings of the **Enable Discovery** and **Monitor Upon Discovery** checkboxes as needed (refer to section 12.1.2.2, "Activating and Deactivating Probes" for descriptions of these fields).

8. Click **Apply**.

12.1.2.4 Setting up the Observer-to-Monitor Communication Channel

The default Observer Communication policy sets up *direct communication* between the observer and monitor. Direct communication allows you to use multiple singleton monitors, with each monitor collecting observations from multiple observers.

If you intend to replicate the monitor by placing a load balancer in between the observers and a group of replicated monitors, you must set the values of the following fields in the Communication Channel section as indicated:

Field Name	Value
Communication path	If your observers will communicate through a load balancer to a monitor group, choose **Through router to monitor group**. This choice displays the following fields in the policy
Router IP address	Specify the IP address of the load balancer that will receive the observation messages.
Router port number	Specify the port number on which the load balancer will receive the observation messages. **Note:** You must also configure this port on your router.
Monitor port number	Specify the port number on which the monitors will receive the forwarded observation messages.

For more information on this topic, refer to the *Business Transaction Management Installation Guide*.

12.1.2.5 Configuring SSL on the Observation Message Flow

The default Observer Communication policy sets up a secure socket connection that is used for sending observation messages from the observer to the monitor. If you prefer

to use a nonsecured socket for this connection, disable the policy's **Enable SSL** checkbox.

With SSL enabled, the monitor is required to authenticate itself to the observer. By default, the SSL connection uses built-in, preconfigured security stores. If you want to use your own security stores, disable the **Use Default Stores** checkbox and fill in the additional fields using information from the following table:

> **Note:** If you are using .NET-based observers, you must deploy a certificate to the machines hosting the observers in order to use an SSL connection. If you are using the default stores, use the preconfigured certificate located at nanoagent\config\ssl\server.cer in the observer installation directory. Refer to the *Business Transaction Management Installation Guide* for more information on deploying the preconfigured certificate.

Field Name (boldface denotes a section name)	Description
Protocol	Required if displayed. Select the SSL protocol. Choices are TLSv1, SSLv3, or Any. SSLv3 is not supported by the .NET observers. This field configures both monitors and observers.
Use Default Stores	This checkbox is enabled by default.
	Leave this checkbox enabled if you want to use the built-in, preconfigured security stores. In this case, you are finished if you are using Java-based observers only. If you are using .NET-based observers, you must also deploy a preconfigured certificate to the machines hosting the observers. You can find the preconfigured certificate at nanoagent\config\ssl\server.cer in the observer installation directory. Refer to the *Business Transaction Management Installation Guide* for more information on deploying the preconfigured certificate.
	Disabling this checkbox displays additional fields and permits you to specify your own security stores.
Monitor	---------- This is a section label----------
	The following five fields pertain to the monitor's key store. All of the following fields are displayed only if the **Use Default Stores** checkbox is disabled.
Key Store Location	Required if displayed. Specify the location of the monitor's SSL key store. You can specify this location as either an absolute path, if the key store file is local to your monitor, or as an HTTP(S) URL, if the file is accessible by HTTP GET.
	The initial value when you open a new policy is **AP-MONITOR-SSL:DefaultKeyStore.ks**. This value points to the built-in, preconfigured key store located at WEB-INF/ssl/DefaultKeyStore.ks in the btmmonitor.war deployment.
Key Store Password	Required if displayed. Specify the password for accessing the SSL key store.
Key Store Type	Required if displayed. Specify the type of JCE (Java Cryptographic Extensions) key store for the monitor to use, for example, **JKS**, **JCEKS**, or **PKCS12**. The initial value is **JKS**.
Key Name	Required if displayed. Specify the certificate and private key. You can enter a key alias or a certificate attribute of the form CN=*value*, UID=*value*, etc.

Field Name (boldface denotes a section name)	Description
Key Password	Required if displayed. Specify the password for accessing the certificate and private key. If unspecified, the password for the key store is used.
Auto-Dispatch Trust Store to Java Observers	If this checkbox is enabled, the monitor will serialize the trust store, and automatically send it to all associated Java observers. This option is ignored for .NET observers. This checkbox is disabled by default.
Auto-Dispatch Java Trust Store	---------- This is a section label ----------
	The following three fields pertain to the auto-dispatched trust store and are displayed only if the **Auto-Dispatch Trust Store to Java Observers** checkbox is enabled.
Trust Store Location	Required if displayed.
	Specify the location of the SSL trust store that the monitor will dispatch to Java observers. You can specify this location as either an absolute path, if the trust store file is local to your monitor, or as an HTTP(S) URL, if the file is accessible by HTTP GET
	The initial value when you open a new policy is **AP-MONITOR-SSL:DefaultTrustStore.ks**. This value points to the built-in, preconfigured trust store located at WEB-INF/ssl/DefaultTrustStore.ks in the btmmonitor.war deployment.
Trust Store Password	Required if displayed. Specify the password for accessing the SSL trust store that the monitor will dispatch to Java observers.
Trust Store Type	Required if displayed. Specify the type of JCE (Java Cryptographic Extensions) trust store that the monitor will dispatch to Java observers, for example, **JKS**, **JCEKS**, or **PKCS12**. The initial value is **JKS**.
Java Observer	---------- This is a section label ----------
	The following three fields pertain to manually installed trust stores and are displayed only if the **Auto-Dispatch Trust Store to Java Observers** checkbox is disabled.
Trust Store Location	Required if displayed.
	Specify the location of the SSL trust store to be used by observers deployed to Java execution environments. You can specify this location as either an absolute path, if the trust store file is local to your observer, or as an HTTP(S) URL, if the file is accessible by HTTP GET.
	The initial value when you open a new policy is **AP-OBSERVER-SSL:DefaultTrustStore.ks**. This value points to the built-in, preconfigured trust store located at nanoagent\config\ssl\DefaultTrustStore.ks in the observer installation directory.
Trust Store Password	Required if displayed. Specify the password for accessing the SSL trust store.
Trust Store Type	Required if displayed. Specify the type of JCE (Java Cryptographic Extensions) trust store that the monitor will dispatch to Java observers, for example, **JKS**, **JCEKS**, or **PKCS12**. The initial value is **JKS**.

12.1.2.6 Configuring Observer Authentication

The default Observer Communication policy requires the observer to authenticate itself to the monitor each time it establishes a connection. You can adjust this setting by

way of the **Observer Authentication** field. You can turn observer authentication off by setting the field to **None**.

You can also require the observer to authenticate itself each time it sends a message to the monitor by setting the field to **Use Message Authentication**. Note, however, that the use of message authentication can significantly degrade performance. You should use this setting only when necessary. For example, if your observer sends its messages to a monitor group whose load balancer is configured for per-message balancing (rather than per-connection), then you cannot use connection authentication. In this case, you must set this field to either **None** or **Use Message Authentication**.

> **Note:** The **Observer Authentication** field is displayed only if the **Enable SSL** field is enabled. Disabling the **Enable SSL** field disables observer authentication as well as the SSL connection.

12.1.2.7 Targeting Observers

New functionality provided with release 12.1.0.4 gives you greater flexibility in configuring observers. In prior releases, you could apply only a single Observer Communication policy to any one monitor. The monitor then generated a single observer configuration from this policy and distributed it to all of its associated observers (you associate an observer to a monitor at installation time by providing the observer with the monitor's location).

As of release 12.1.0.4, you can apply multiple Observer Communication policies to a single monitor and then target each of these policies at a different observer or set of observers (the observers you target must be associated with the monitor). The following steps describe the procedure for targeting observers:

> **Note:** The order of these steps is intended to help you understand the concepts involved in targeting observers. However, when you actually perform the task, it is most efficient to complete all the steps in one policy and then move on to the next policy.

1. Designate one policy as the source for generating the monitor's configuration by leaving the **Generate Configuration for Observers Only** checkbox disabled (we'll refer to this policy as the "monitor policy").

   The "monitor policy" generates the monitor configuration and also one observer configuration.

2. Apply as many additional policies to the monitor as you require for configuring your observers (we'll refer to these policies as "observer policies").

   a. Designate each "observer policy" as a source for generating only an observer configuration by enabling the **Generate Configuration for Observers Only** checkbox.

   b. Ensure that the values of all the other fields in the **Communication Channel** section of each "observer policy" matches the values in the "monitor" policy.

3. Target your "observer policies" at specific observers.

   a. Use the **Configuration Label** and/or **Observer Base Address** fields to specify which observers you want to target with the observer configuration (for more information about these and related fields, refer to section 12.1.2.7.1, "Observer Configuration Labels" and section 12.1.2.7.5, "Field Reference for Targeting

Observers").

b. Ensure that labels and addresses specified in one "observer policy" are not specified in any other policy that is applied to the same monitor (for additional information, see section 12.1.2.7.2, "Rejection of Observer Communication Policies").

c. *Optional* – Enable the **Validate Addresses** checkbox to ensure that all targeted observers are known to Business Transaction Management.

If you target an unknown observer and enable this field, the policy will be rejected. If you want to target an observer that is not currently known but will be later, you should disable this field.

d. *Optional* – (Do not enable this checkbox if you apply the policy to multiple monitors.) Enable the **Enforce to Monitor** field to validate that all the targeted observers are associated with the monitor to which you apply the policy.

4. *Optional* – Leave one policy untargeted to serve as the default observer configuration for all observers associated with the monitors to which the policy is applied.

An untargeted policy serves as the default configuration for observers that are not specifically targeted by another policy. You could, for example, leave the "monitor policy" untargeted and use it as a default observer configuration. If you attempt to create a second untargeted policy, the second policy will be rejected (for information about rejected policies, see section 12.1.2.7.2, "Rejection of Observer Communication Policies").

> **Note:** If you configure different monitors (or groups of monitors) using different policies, you can define a different default observer configuration for each.

As in previous releases, you can use a single policy for configuring monitors and all of their associated observers, if you desire (in other words, you don't have to target specific observers). In such a scenario, you would use a "monitor policy" that is not targeted at specific observers. This policy will generate a default observer policy, and, since no specific observers are targeted, all observers associated with the monitors will receive this default configuration.

12.1.2.7.1 Observer Configuration Labels An observer configuration label is a simple text string that conceptually identifies a set of observers (for example, CONFIG_LABEL_ MY_OBSERVERS). You target an observer by specifying either a label or the absolute address of the application server in which the observer is deployed (for example, http://my_host.com:7011). You can specify any number of labels and/or addresses, and target any number of observers in a single policy.

Labels enable you to group observers logically rather than physically for configuration purposes. Applying a label is a two-step procedure that you can perform in either order:

- In the application server that hosts the observer, create a system property named ap.nano.config.label and set its value to your label string (refer to the *Business Transaction Management Installation Guide* for details on how to do this).

- Set the **Configuration Label** field of the Observer Communication policy that you will use to configure your observer to the same value as ap.nano.config.label.

12.1.2.7.2 Rejection of Observer Communication Policies There are a numbers of ways, related to the targeting of observers, in which you might inadvertently cause an Observer Communication policy to be rejected. All of the following scenarios will cause a policy to be rejected:

- Attempting to apply more than one untargeted policy (default observer configuration) to the same monitor

- Attempting to specify the same observer configuration label in two different policies that are applied to the same monitor

- Attempting to specify the same observer base address in two different policies that are applied to the same monitor

Labels are scoped to the monitor to which a policy is applied. This means that you can reuse a particular label name across policies if the policies are applied to different monitors, but you cannot reuse a label name across policies applied to the same monitor. This scoping principle also pertains to untargeted policies.

If a policy is rejected, select the policy in the work area of the console and display the **Targets** tab. This tab provides information about the cause of a rejected policy.

12.1.2.7.3 Order of Precedence The order of precedence that determines which configuration an observer will receive is as follows:

1. Observer base address

   If a policy specifies the observer's base address, then the observer will receive the configuration generated by that policy.

2. Observer configuration label

   If a policy specifies the observer's configuration label and no policy specifies the base address, then the observer will receive the configuration generated by the policy that specifies the configuration label.

3. Untargeted policy

   If a an untargeted policy exists and no policy specifies the observer's base address or configuration label, then the observer will receive the configuration generated by the untargeted policy.

12.1.2.7.4 Preconfigured Observer Communication Policies

Business Transaction Management provides a number of Observer Communication policies that are preconfigured for monitoring particular types of applications. You can edit any of these policies and tailor them to your monitoring needs. The name of each policy is displayed in bold, followed by a description:

- **Observer Communication Policy - Default**

   This policy generates both a monitor configuration and an untargeted (default) observer configuration. By default, this policy is applied to all monitors in the system. The observer configuration is distributed to all associated observers that are not specifically targeted by a different policy.

- **Observer Communication Policy - Fusion Applications**

   This policy generates only an observer configuration. By default, this policy is applied to all monitors in the system. The observer configuration is targeted at observers tagged with the label CONFIG_LABEL_FAPPS. This configuration activates probes and adjusts observer runtime settings for monitoring Oracle Fusion Application components.

- **Observer Communication Policy - JavaEE**

  This policy generates only an observer configuration. By default, this policy is applied to all monitors in the system. The observer configuration is targeted at observers tagged with the label CONFIG_LABEL_JAVAEE. This configuration activates probes and adjusts observer runtime settings for monitoring JavaEE components.

- **Observer Communication Policy - OSB**

  This policy generates only an observer configuration. By default, this policy is applied to all monitors in the system. The observer configuration is targeted at observers tagged with the label CONFIG_LABEL_OSB. This configuration activates probes and adjusts observer runtime settings for monitoring Oracle Service Bus components.

- **Observer Communication Policy - SOA**

  This policy generates only an observer configuration. By default, this policy is applied to all monitors in the system. The observer configuration is targeted at observers tagged with the label CONFIG_LABEL_SOA. This configuration activates probes and adjusts observer runtime settings for monitoring Oracle SOA components.

12.1.2.7.5 Field Reference for Targeting Observers

Field Name (boldface denotes a section name)	Description
Generate Configuration for Observers Only	Leave this checkbox disabled if you want this policy to generate both a monitor and observer configuration. You can apply only one such policy to any monitor. If you apply additional policies to a monitor, they must all have this checkbox enabled. With this checkbox enabled, the policy generates only an observer configuration. Even if this checkbox is enabled, you must still provide values for all other fields in the **Communication Channel** section of the policy, and their values must match all other policies applied to the same monitor.
Targeted Observers	---------- This is a section label ---------- Use this section to specify which observers receive the observer configuration generated by this policy.

Field Name (boldface denotes a section name)	Description
Target Specific Observers	Enable this checkbox if you want to configure only specific observers with the observer configuration generated by this policy. Enabling this checkbox displays additional fields that let you specify which observers should receive the configuration. These additional fields let you target observers by way of addresses and/or labels. If you leave this checkbox disabled, the policy is untargeted and will generate a default configuration for all observers associated with the monitors to which the policy is applied. If an associated observer is not targeted by a policy, it will receive this default configuration. You can apply only one untargeted policy to any monitor. If you attempt to apply a second untargeted policy to a monitor, the policy will be rejected. **Notes:** You can target observers by way of two different mechanisms—addresses and labels. But, only one policy can target any particular observer by way of the same mechanism. For example, if policy A targets an observer by way of a label, then policy B cannot target that same observer by way of a label. In this case, policy B would be rejected. Policy B could, however, target the observer by way of an address. In this case, the observer would receive its configuration from policy B because addresses take precedence over labels.
Configuration Label	*Optional* – Specify a comma-delimited list of observer configuration labels. Observers tagged with any of the specified labels will receive configurations generated by this policy (unless a different policy targets the observer by way of an address). This field accepts text input and is case insensitive. **Note:** You tag observers by way of a system property named ap.nano.config.label in the application server hosting the observer.
Known Address	---------- This is a section label ---------- *Optional* – Use this section to choose observer addresses from a drop-down list of addresses known by Business Management Transaction. The specified observers will receive configurations generated by this policy.
Observer Base Address	Use this drop-down list to select the address of the container where the observer is deployed.
[add observer address]	Click this link to add an **Observer Base Address** drop-down list.
Any Address	---------- This is a section label ---------- *Optional* – Use this section to manually enter observer addresses into a text field. The specified observers will receive configurations generated by this policy.
Observer Base Address	Use this field to manually enter the address of the application server where the observer is deployed, for example, http://my_host.com:7011.
[add observer address]	Click this link to add an **Observer Base Address** text field.
Validate Addresses	Enable this checkbox to ensure that all observers targeted by the policy are known to Business Transaction Management. If you target an unknown observer and enable this field, the policy will be rejected. If you want to target an observer that is not currently known but will be later, you should disable this field.

Field Name (boldface denotes a section name)	Description
Enforce to Monitor	Enable this checkbox to validate that all the targeted observers are associated with the monitor to which you apply the policy.
	Do not enable this checkbox if you apply the policy to multiple monitors.

12.1.2.8 Enabling/Disabling Drilldowns from Real User Experience Insight

You can control the ability of users to perform user-interface drilldowns from Oracle Enterprise Manager Real User Experience Insight into Business Transaction Management. By default, drilldown capability is enabled. To disable or re-enable drilldown capability, set the WEB_APP probe's rueiPresent attribute as described in Section 12.1.2.9.

12.1.2.9 Request Monitoring and Operation Modeling for the WEB_APP Probe

The WEB_APP probe lets you monitor web applications that are implemented as Java servlets. This probe provides a generic form of processing that can be used with all types of Java servlets and a specialized form of processing optimized for use with Oracle Application Development Framework (ADF) applications. This section refers to these different types of processing as *rulesets*. The generic type of processing is referred to as the URL ruleset, and the type of processing used for ADF applications is referred to as the ADF ruleset. The probe defaults to the URL ruleset.

Unless configured otherwise, the WEB_APP probe monitors all requests to the web applications in the monitored application server. In many cases, however, you might not want to monitor all requests. For example, you might not want to monitor requests for static resources such as image and HTML files. For this reason, the default Observer Communication policy is configured not to monitor requests for resources that have the following file extensions: jpg, jpeg, html, htm, css, gif, png, ico, js, swf, cur. This selective monitoring is specified by way of a snippet of XML configuration code that appears in the default policy's **WEB_APP probe configuration** field.

You can edit this default XML configuration code in order to control the types of requests that are monitored. You can also add XML elements to control how your application's operation names are abbreviated for display in the Management Console (operation names are derived from request URLs). The following table describes the XML elements and attributes that you can use in your configuration code. Usage examples are provided after the table.

> **Note:** Ordering of the XML elements is critical. The required ordering of the elements is described in the table. Incorrect ordering will cause the policy to be rejected.

Element	Attribute	Description	Supported rulesets
servletObserver	-	Encompassing tag containing configuration information for all applications monitored by the WEB_APP probe. There is only one <servletObserver> element.	ADF URL

Element	Attribute	Description	Supported rulesets
-	rueiPresent	Indicates that Oracle Enterprise Manager Real User Experience Insight is installed in front of the monitored applications. This attribute controls the ability of users to perform user-interface drilldowns from Real User Experience Insight into Business Transaction Management.	ADF URL
		Valid values: true or false. The default setting is true.	
		When this attribute is set to true, Business Transaction Management adds headers to the HttpResponse, thereby enabling drilldown capability. To disable drilldown capability, set this attribute to false.	
-	rueiMatches	Indicates that the Real User Experience Insight naming scheme matches the Business Transaction Management naming scheme. Valid values: true or false.	ADF URL
globalExcludeList	-	Use this element to globally exclude specified URLs from monitoring. You can specify URLs to exclude by file type, context root, pattern matching, or length.	ADF URL
		Element ordering: If used, you must place this element as the first child of the <servletObserver> element. There can be only one per <servletObserver>.	
-	ext	Contains a comma-delimited list of file extensions, for example, ext="html, htm, jpg, css". Files of the specified types are excluded from monitoring.	ADF URL
-	contextRoot	Contains a comma-delimited list of context roots, for example, contextRoot="console, medrec, bookmart". URLs containing any of the specified context roots are excluded from monitoring.	ADF URL
		To specify a blank context root, use "/", for example, contextRoot="console, /, bookmart".	
-	pathPattern	Contains a comma-delimited list of URL patterns. URLs that match any of the specified patterns are excluded from monitoring. Wild cards are allowed and denoted using "*".	ADF URL
		Notes on pattern matching (applies to both the pathPattern and pattern attributes):	
		▪ The pattern matching is not performed on the protocol, host name, port number, or query string. Pattern matching is performed only on the section of the URL that begins with and includes the "/" in front of the context root and that ends at but does not include the "?". Furthermore, your pattern must match that entire string. For example, in the URL "http://myhost:7654/my_context/test?param1=value1", pattern matching is performed on the string "/my_context/test". Some patterns that would match this URL include: "*context*", "/my*", and "*test".	
		▪ Pattern matching is performed on the decoded URL, so your pattern must therefore match the decoded URL. For example, the URL string "/your%20context/test" when decoded is "/your context/test" (%20 has been converted to a space). To match this URL, your pattern must match a space rather than "%20".	

Element	Attribute	Description	Supported rulesets
-	pathLength	Positive integer. URLs that exceed this number of characters are excluded from monitoring.	ADF URL
		Characters in the protocol, host name, port number, and query string are not included in the count. The section of the URL that this attribute operates on is the same as for the pathPattern attribute.	
application	-	Denotes an application to be monitored.	ADF URL
		Element ordering: This element is a child of the <servletObserver> element. It must not precede the <globalExcludeList> or <globalAdfOptions> elements. There can be any number per <servletObserver> element.	
-	contextRoot	The context root of the monitored application. The value of this attribute is used as the service name. To specify a blank context root, use "/", for example contextRoot="/".	ADF URL
framework	-	This element is used to specify which URLs should be handled by which ruleset by way of the <include> child element.	ADF URL
		Element ordering: If used, you must place this element as the first child of the <application> element. There can be one for each ruleset type per <application> element.	
-	type	Specifies the ruleset that should handle the <include> URL patterns. Valid values are ADF and URL. For ADF web applications, set this attribute to ADF. For other web applications, set this attribute to URL. The requests processed by each ruleset are mutually exclusive.	ADF URL
		If no <framework> tag is specified, the probe defaults to the URL ruleset. If a <framework> tag *is* specified, then there is no default value.	
include	-	This element allows you to exclude all requests except those that match the given wild card expression when mapping a request to a ruleset.	ADF URL
		You cannot use this element to include previously excluded URLs. For example, if you used <globalExcludeList> to exclude the "png" extension, you cannot override that exclusion by specifying "*.png" in the <include> element.	
		Element ordering: This element is a child of the <framework> element. There can be many per <framework> element.	
-	pattern	The URL pattern to match. Wild cards are allowed and denoted using "*".	ADF URL
		This attribute follows the same pattern matching rules as the pathPattern attribute. Refer to the <globalExcludeList> element's pathPattern attribute for information on these rules.	
excludeList	-	Use this element to specify file types within a particular application that you want to exclude from monitoring. This element overrides the <globalExcludeList> element for a specific application.	ADF URL
		Element ordering: This element is a child of the <application> element. It must not precede any <framework> elements and must precede all <adfOptions> and <operationRule> elements. There can be only one per <application> element.	

Element	Attribute	Description	Supported rulesets
-	ext	Contains a comma-delimited list of file extensions, for example, "html, htm, jpg, css". Files of the specified types (within the parent application) are excluded from monitoring.	ADF URL
		If the you want to monitor all file types in the application but the ext attribute of the <globalExcludeList> element is set, then set this ext attribute to the null string (for example: ext="").	
adfOptions	-	Use this element to partition operations within a particular application by appending property values contained in the ADF UI request parameter **oracle.adf.view.rich.monitoring.UserActivityInfo** to the operation name. In order to use this feature, the UserActivityInfo request parameter must be enabled (see section 12.1.2.9.1, "Enabling the UserActivityInfo Request Parameter"). The attributes of this element correspond to properties in the UserActivityInfo request parameter. The attributes control whether the property values are appended to the operation name. The property values are appended only if they exist in the request parameter.	ADF URL
		Element ordering: This element is a child of the <application> element. It must not precede any <framework> or <excludeList> elements and must precede all <operationRule> elements. There can be only one per <application> element.	
-	appendRegionViewId	Set this attribute to **true** to append the value of the regionViewId property to the operation name using the format **__regionViewId**. The default setting is **false**.	ADF URL
operationRule	-	This element specifies the parts of the URL for which a unique combination of values will constitute an operation. Use this element to abbreviate the operation name that is derived from the URL.	ADF URL
		Element ordering: This element is a child of the <application> element. It must not precede any <framework>, <excludeList>, or <adfOptions> elements. There can be only one per <application> element.	
-	excludeDirectories	Contains a comma-separated list of directory levels to exclude from the operation name. For example, you could exclude "/faces" or the session ID. Note that the context-root is not considered a directory level. Also, the excludeDirectories count starts with "1", not "0".	ADF URL

Element	Attribute	Description	Supported rulesets
paramGroup	-	Use this element to partition operations by multiple request parameters. You specify the request parameters by adding \<partitionByParam> elements as children of this element. Partitioning occurs only if all specified parameters exist in the request. The parameter names and values are appended to the operation name in this format: `_name1_value1__name2_value2__name3_value3` Notice that there are two underscore characters between each name-value pair. This element can contain a maximum of three \<partitionByParam> elements. **Note:** Using this element to partition an operation that is used in an existing transaction definition changes the semantics of the transaction. For example, requests that contain the specified parameter will not be counted as requests for the original operation and, therefore, will not be counted as belonging to the transaction. You might need to update the definition of your transaction accordingly. **Element ordering:** If used, you must place this element as the first child of the \<operationRule> element. There can be multiple \<paramGroup> elements. This element must precede all stand-alone \<partitionByParam> elements (that is, those that are not children of a \<paramGroup> element).	ADF URL

Element	Attribute	Description	Supported rulesets
partitionByParam	-	This element partitions an operation based on the value of the specified request parameter. Each unique parameter value is modeled as a separate operation. The parameter can be either a URL parameter or a POST parameter.	ADF URL
		For example, assume we have an **orderApplication.jsp** that takes a parameter named **action**. Normally, requests to orderApplication.jsp would be modeled as requests to a single operation named orderApplication.jsp. However, if we use <partitionByParam> and partition by the action parameter, all requests to orderApplication.jsp that contain an action parameter will be modeled as requests to an operation named **orderApplication.jsp_action_***paramValue*, where *paramValue* is the value of the action parameter. And, importantly, requests containing an action parameter will not be counted as requests to the operation orderApplication.jsp. (See also, "Example 2 – Adding a parameter name/value pair to an operation name" on page 12-22.)	
		Note: Using this element to partition an operation that is used in an existing transaction definition changes the semantics of the transaction. For example, requests that contain the specified parameter will not be counted as requests for the original operation and, therefore, will not be counted as belonging to the transaction. You might need to update the definition of your transaction accordingly.	
		Element ordering: If used, you can place this element in two positions:	
		■ inside of a <paramGroup> element	
		■ inside of the <operationRule> element, following all <paramGroup> elements	
		There can be any number per <operationRule> element. If you want to use a single <partitionByParam> element before a <paramGroup>, place it inside of its own <paramGroup> element.	
		Note: This element does not support ADF page input parameters.	
-	name	The name of the parameter to use for partitioning requests. Each distinct value of the given parameter corresponds to its own operation. The parameter is appended to the operation name as "*_name_value*", where *name* is the name of the parameter and *value* is its value.	ADF URL
secureParam	-	This element represents a URL or POST parameter whose value should be kept hidden or not get stored at all (for example, a password), both in operation names and in Business Transaction Management messages.	ADF URL
		Element ordering: This element is a child of the <operationRule> element. It must not precede any <paramGroup> or <partitionByParam> elements. There can be any number per <operationRule> element.	
-	name	The name of the parameter whose value should be hidden or not stored at all.	ADF URL

> **Notes:**
>
> **(1)** Service and operation names are derived from the request URL. In order to conform to XML standards, the probe substitutes an underscore symbol ("_") in place of special characters such as slashes, question marks, and equal signs ("/", "?", "=").
>
> **(2)** Service and operation names are shortened if they exceed 255 bytes. This shortening is performed by truncating the name to 252 bytes, and then appending "…" to it.
>
> **(3)** Processing for the pathLength attribute occurs before the processing for the pathPattern attribute. Service and operation name shortening occurs at the end of processing. The complete order of execution is as follows:
>
> 1. The pathLength attribute of the <globalExcludeList> element
> 2. The contextRoot attribute of the <globalExcludeList> element
> 3. The pathPattern attribute of the <globalExcludeList> element
> 4. The ext attribute of the <excludeList> element if it exists
>
>   Otherwise:
>
>   The ext attribute of the <globalExcludeList> element
> 5. The include elements within the <framework> element
> 6. Service and operation name shortening

Example 1 – Abbreviating an operation name

```
<ap:servletObserver xmlns:ap="http://namespace.amberpoint.com/amf">
 <ap:application contextRoot="/mywebshop">
  <ap:operationRule excludeDirectories="1, 2" />
 </ap:application>
</ap:servletObserver>
```

The preceding configuration code applied to this request URL:

```
http://secure.banking.de:7001/mywebshop/shopping/s28373/basket/checkout.jsp
```

produces the following objects in Business Transaction Management:

Object	Value	Explanation
Service	mywebshop	The service name is the value of the contextRoot attribute.
Endpoint	http://secure.banking.de:7001/mywebshop	The endpoint is the physical location of the monitored web application plus the service name (the value of the contextRoot attribute).
Operation	basket_checkout.jsp	By default, the operation name consists of the directories and filename from the request URL. In this case, the default operation name would be shopping/s28373/basket/checkout.jsp. However, because the <operationRule> element's excludeDirectories attribute is set to "1, 2", the first and second directories (shopping/s28373/) are excluded.

Example 2 – Adding a parameter name/value pair to an operation name

```
<ap:servletObserver xmlns:ap="http://namespace.amberpoint.com/amf">
 <ap:application contextRoot="/physician">
  <ap:operationRule>
   <ap:partitionByParam name="lastName"/>
  </ap:operationRule>
 </ap:application>
</ap:servletObserver>
```

The preceding configuration code applied to this request URL:

```
http://stbdm02:7011/physician/physicianSection/viewRecordSummary.action
```

with a POST parameter of "lastName=Einstein", produces the following objects in Business Transaction Management:

Object	Value	Explanation
Service	physician	The service name is the value of the contextRoot attribute.
Endpoint	http://stbdm02:7011/physician	The endpoint is the physical location of the monitored web application plus the service name (the value of the contextRoot attribute).
Operation	physicianSection_viewRecordSummary.action_lastName_Einstein	The name and value of the parameter specified by the <partitionByParam> element is appended to the default operation name.

Example 3 – Filtering requests and applying rulesets

```
<ap:servletObserver xmlns:ap="http://namespace.amberpoint.com/amf">
 <ap:application contextRoot="/em">
  <ap:framework type="ADF">
   <ap:include pattern="*/faces*"/>
  </ap:framework>
  <ap:framework type="URL">
   <ap:include pattern="*/console*" />
  </ap:framework>
  <ap:operationRule excludeDirectories="1, 2" />
 </ap:application>
</ap:servletObserver>
```

The preceding configuration code applied to these request URLs:

```
http://myhost:17861/em/faces/ocamm/managers/ocammHome
http://myhost:17861/em/console/all/targets/search
http://myhost:17861/em/em2go/about.jsp
```

produces the following objects in Business Transaction Management:

Object	Value	Explanation
Service	em	The service name is the value of the contextRoot attribute.

Object	Value	Explanation
Endpoint	http://myhost:17861/em	The endpoint is the physical location of the monitored web application plus the service name (the value of the contextRoot attribute).
Operation	managers_ocammHome	*(This pertains to the first example URL)*
		Because the <framework> element's type is set to ADF, the ADF ruleset is used. Therefore, by default, the operation name consists of the directories and filename from the request URL. In this case, the default operation name would be faces/ocamm/managers/ocammHome. However, because the <operationRule> element's excludeDirectories attribute is set to "1, 2", the first and second directories (faces/ocamm/) are excluded.
Operation	targets_search	*(This pertains to the second example URL)*
		Because the <framework> element's type is set to URL, the URL ruleset is used. Therefore, by default, the operation name consists of the directories and filename from the request URL. In this case, the default operation name would be console/all/targets/search. However, because the <operationRule> element's excludeDirectories attribute is set to "1, 2", the first and second directories (console/all/) are excluded.
N/A	N/A	*(This pertains to the third example URL)*
		A <framework> element of type URL was specified with only the pattern "*/console*". This request does not fit that pattern. It also does not fit the pattern specified in the ADF <framework> element. Thus it is excluded from monitoring.

Example 4 – Adding multiple parameter name-value pairs to an operation name

```
<ap:servletObserver xmlns:ap="http://namespace.amberpoint.com/amf">
 <ap:application contextRoot="/physician">
  <ap:operationRule>
   <ap:paramGroup>
    <ap:partitionByParam name="firstName"/>
    <ap:partitionByParam name="lastName"/>
   </ap:paramGroup>
  </ap:operationRule>
 </ap:application>
</ap:servletObserver>
```

The preceding configuration code applied to this request URL:

```
http://st02:7011/physician/physicianSection/viewRecordSummary.action
```

with POST parameters of "firstName=John" and "lastName=Doe" produces the following objects in Business Transaction Management:

Object	Value	Explanation
Service	physician	The service name is the value of the contextRoot attribute.
Endpoint	http://st02:7011/physician	The endpoint is the physical location of the monitored web application plus the service name (the value of the contextRoot attribute).

Object	Value	Explanation
Operation	physicianSection_ viewRecordSummary.action _firstName_John__ lastName_Doe	The name and value of the parameters specified within the <paramGroup> element are appended to the default operation name in the order they are listed. If either the firstName or lastName parameters were not available, then this <paramGroup> element would have been ignored, resulting in a value of "physicianSection_viewRecordSummary.action".

Example 5 – Parameter priority handling

```
<ap:servletObserver xmlns:ap="http://namespace.amberpoint.com/amf">
 <ap:application contextRoot="/physician">
  <ap:operationRule>
   <ap:paramGroup>
    <ap:partitionByParam name="firstName"/>
    <ap:partitionByParam name="lastName"/>
   </ap:paramGroup>
   <ap:partitionByParam name="lastName"/>
   <ap:partitionByParam name="middleName"/>
  </ap:operationRule>
 </ap:application>
</ap:servletObserver>
```

The preceding configuration code applied to this request URL:

```
http://st02:7011/physician/physicianSection/viewRecordSummary.action
```

with POST parameters of "lastName=Smith" and "middleName=Rodney" produces the following objects in Business Transaction Management:

Object	Value	Explanation
Service	physician	The service name is the value of the contextRoot attribute.
Endpoint	http://st02:7011/physician	The endpoint is the physical location of the monitored web application plus the service name (the value of the contextRoot attribute).
Operation	physicianSection_ viewRecordSummary.action _lastName_Smith	The probe first checks whether the parameters specified in the <paramGroup> element are provided in the request. Because the firstName parameter is not provided, that entire <paramGroup> element is skipped. The probe then checks the next <paramGroup> or stand-alone <partitionByParam> element. Because the next element specifies a "lastName" parameter and that parameter is provided in the request, the parameter name and value is appended to the operation name (_lastName_Smith). After finding this match, the probe stops checking for parameters, meaning that the "middleName" parameter is not appended to the operation name.

12.1.2.9.1 Enabling the UserActivityInfo Request Parameter If you want to use the operation partitioning features provided by the <adfOptions> and <globalAdfOptions> elements, then you must ensure that the UserActivityInfo request parameter is enabled for your application. Oracle Fusion Applications environments should have this enabled by default as well as environments using the ADF support in Oracle Enterprise Manager Real User Experience Insight.

To enable the UserActivityInfo request parameter in your ADF application, set the following property in the application's web.xml file:

```
<context-param>
  <description>
    This parameter notifies ADF Faces that the ExecutionContextProvider
    service provider is enabled. When enabled, this will start
    monitoring and aggregating user activity information for the client
    initiated requests. By default, this param is not set or is false.
  </description>
  <param-name>
    oracle.adf.view.faces.context.ENABLE_ADF_EXECUTION_CONTEXT_PROVIDER
  </param-name>
  <param-value>true</param-value>
</context-param>
```

12.1.2.10 Info Settings Field Reference

Field Name	Description
Name	Required. Specifies the name of your policy. You can set this field to any unique string.
Version	Optional. This field is descriptive only and is provided for you to enter any pertinent information about the policy.
Notes	Optional. This field is descriptive only and is provided for you to enter any pertinent information about the policy.

12.1.2.11 Advanced Settings Field Reference

Field Name (boldface denotes a section name)	Description
Observer Behavior	-
Configuration polling interval	Required. Use this field to specify, in seconds, how often the observer checks for a new configuration.
Instrument update interval	Required. Use this field to specify, in seconds, how often the observer sends measurement data to the monitor.
Number of connections	Use this field to specify the number of socket connections that the observer opens. Using multiple connections improves throughput of observations.

Field Name (boldface denotes a section name)	Description
Mapping Algorithm	Specifies the algorithm used to modify the host name-port number portion of the request and WSDL URLs. Choose from these values:
	As sent - The observer does not rewrite the URL and forwards it unchanged to the monitor.
	Use hostname - The observer replaces the host name portion of the URL with the fully qualified name of the server's host. It replaces the port number portion of the URL with the port number on which the server is listening. The host name and port number are obtained from the deployment environment.
	This algorithm is useful for clustered servers fronted by a load balancer. In this scenario, the original request URL is that of the load balancer, with the load balancer's host name and port number. If the observer passes the original request URL to the monitor, the entire cluster of servers is modelled as a single server. With the algorithm set to useHostname, each server is modelled separately.
	Use IP address - The observer converts the URL's host name into an IP address and leaves the port number unchanged. The IP address is obtained from the deployment environment.
	This algorithm can be useful if the monitor cannot resolve hostnames to valid IP addresses. You should not use this algorithm if the server has multiple IP addresses.
	Use fully qualified name (FQN) - The observer converts the URL's host name into a fully qualified name and leaves the port number unchanged.
	This algorithm can be useful for a server that has multiple IP addresses.
	Use alternate - This algorithm lets you provide specific values for the host name, port number, and protocol. Use the following three fields to input the values. If you do not specify a value in any one of the fields, the corresponding portion of the URL is left unchanged.
	Note: In the case of the OSB observer, the target service URL is always set to FQN in the observer configuration, but this setting is not visible in this policy.
Alternate host	The value to use as the host portion of the URL.
Alternate port	The value to use as the port number portion of the URL.
Alternate protocol	The value to use as the protocol portion of the URL.
Discovery processing interval	Specifies the regular interval (in minutes) at which the observer attempts to discover new components. The default value is 3 minutes. Any value over 1440 minutes (one day) is interpreted as 1440 minutes. Any negative value is interpreted as 3 minutes.
Enable dynamic discovery period	Enables an algorithm for the timing when the observer attempts to discover new components. When enabled, the discovery interval doubles after each discovery, to a maximum of 32 times the specified Discovery processing interval. Any calculated value over 1440 minutes (one day) is interpreted as 1440 minutes. This feature prevents the frequent processing of components that have not changed.
	For example, if the Discovery processing interval is set to 300 and this option is enabled, the second discovery occurs after 600 minutes, then 1200 minutes, but will not increase beyond 1440 minutes.
Observer Troubleshooting	-

Field Name (boldface denotes a section name)	Description
Enable trace logging	Trace logging is always enabled and is set to **Info** by default. Use this checkbox to enable the **Trace logging level** field so that you can edit the setting.
	For information on other types of observer error logging, location of error log files, and configuring the location of error logs, see section 12.1.3, "Logging Observer Errors and Debugging Information".
Trace logging level	Use this field to specify the level of information you want written to the log file. The possible values, in order from least to most information, are:
	Info, Fine, Finer, Finest
Trace file size	This field specifies, in kilobytes, the size of the trace log files.
Trace files count (rotation)	This field specifies the maximum number of trace log files. When the maximum number of trace log files are full, rotation occurs, whereby the oldest file is overwritten with a new file.
	In general, you will change this setting only when asked to do so by the Oracle support team.
Log observed messages to file	Enable this checkbox if you want observed messages written to a file.
Observation log directory	The path to the directory containing the observation log files. For WebLogic, OC4J, WebSphere, and JBoss servers, you can specify an absolute path or a relative path. For other servers, you can specify only an absolute path. Relative paths are relative to the default location. The default locations are:
	■ **WebLogic** – the domain directory
	■ **OC4J** – the j2ee\home directory inside your SOA Suite installation directory
	■ **Enterprise Gateway** – the home directory (top-level installation directory) of the Enterprise Gateway server.
	■ **WebSphere** – the profile directory
	■ **JBoss** – the JBOSS_HOME/bin directory
	■ **WCF and ASP.NET** – the C:/temp/NanoAgentBaseDir directory
	Note: The default log location for WCF and ASP.NET is not a true default. It is simply the default setting of the AmberPoint:NanoLogBaseDir key. If you set this key to null, log files will not be created.
Observer Message Queue	The fields in this section affect the behavior of the observer's observation queue. The observer copies observed, service-bound messages to this outgoing queue. These observations are then pulled off the queue and sent to the monitor.
Queue size	Required. This field specifies the maximum number of messages the observer's observation queue can hold. A larger number allocates more memory for the queue.

Field Name (boldface denotes a section name)	Description
Maximum message size	Optional. This field specifies, in kilobytes, the maximum size of a message that can be placed on the observer's observation queue. Messages larger than the specified size are first truncated and then placed on the queue.

You can use this field to reduce the load on the network and monitor.

The truncation applies only to the body of the message. The message envelope is left intact.

Note: If your service is a client to other services, you must set this setting to the same value for the observers monitoring those services. Failure to do so will disrupt the dependency tracking mechanism and cause the appearance of nonexistent clients in dependency diagrams. |
| If queue is full | **Caution:** Leave this field at its default setting unless you are instructed by the Oracle support team to edit it. The default setting for this field is **Forward service-bound messages without copying them onto queue**.

The **If queue is full** field specifies the behavior of the observer's observation queue if it fills up. Choose between these options:

Delay service-bound messages until there is room to copy them onto queue - If the queue is full, the observer waits until the queue frees up enough space to hold the observation before forwarding the original message on to the service. This setting ensures that copies of all messages are forwarded to the monitor. Choosing this option together with the **Delay receipt of observed messages over socket until queue has room** option in the **Monitor Configuration** section of the policy ensures that all observations are logged. However, in high-traffic situations, such a setting might result in the slowing down of message processing by the monitored application.

Forward service-bound messages without copying them onto queue - If the queue is full, the observer forwards the original message on to the service without copying it to the queue. Choosing this option ensures that the observer does not slow down the monitored application's message processing in order to log observations. However, in high-traffic situations, this setting might result in a loss of observations.

Note: In no case, does the observer discard the original service-bound messages. |
| **Monitor Message Queue** | The fields in this section affect the behavior of the monitor's observation queue. As observations arrive at the monitor, they are placed on this incoming queue. The monitor then pulls the observations off the queue and processes them in order to gather data on performance, transactions, and so forth. |
| Queue size | Required. This field specifies the maximum number of messages the monitor's observation queue can hold. A larger number allocates more memory for the queue. |

Field Name (boldface denotes a section name)	Description
Maximum message size	Optional. This field specifies, in kilobytes, the maximum size of messages that are accepted on the monitor's observation queue. You can use this field to reduce the load on the monitor by constraining the processing of large messages.
	By specifying a value in this field, you instruct Business Transaction Management to drop both the request and response message (or fault message, in case of a fault) if either message in the pair is larger than the specified value. Oversized messages are dropped without being processed and are not used in calculating performance measurements such as throughput or average response time.
Idle socket timeout	Required. This field specifies the maximum number of milliseconds that the socket on which the monitor receives observations remains open in the absence of traffic.
Retain request messages for a maximum of	Optional. This setting specifies the number of seconds the monitor holds on to a request message before assuming that no response will arrive. Once this time has been exceeded, the request is processed as if the response message timed out. The default value of 60 seconds is used if you leave this field blank or set it to 0.
Number processing threads handling messages	Optional. This field specifies the number of threads the monitor allocates for processing observation messages. The default value of 5 is used if you leave the field blank or set it to 0.
Number processing threads handling endpoint discovery	Optional. This field specifies the number of threads the monitor allocates for processing endpoint discovery messages. The default value of 2 is used if you leave the field blank or set it to 0.
If queue is full	Required. This field specifies the behavior of the monitor's observation queue if it fills up. Choose between these options:
	Delay receipt of observed messages over socket until queue has room - If full, the queue rejects incoming observations (message copies) until it has freed up space for them. In this case, the observer resends observations until they are successfully placed on the queue. Choosing this option together with the **Delay service-bound messages until there is room to copy them onto queue** option in the **Observer Configuration** section of the policy ensures that all observations are logged. However, in high-traffic situations, such a setting might result in the slowing down of message processing by the monitored application.
	Discard incoming observed messages - If full, the queue discards incoming observations. Choosing this option together with the **Forward service-bound messages without copying them onto queue** option in the **Observer Configuration** section of the policy ensures that the observer does not slow down the monitored application's message processing. However, in high-traffic situations, this setting might result in a loss of observations.
	Note: In no case, does the monitor discard the original application messages.
Servlet Observer Configuration	-

Field Name (boldface denotes a section name)	Description
Specify WEB_APP probe configuration	This checkbox pertains to the WEB_APP probe and allows you to configure the selective monitoring of requests and modeling of operations. Enable this checkbox to display the **WEB_APP probe configuration** field, where you can input configuration code for these options. This field is enabled by default.
	Note: If you provide a custom observer configuration in the **Custom observer configuration** field, do not enable this checkbox. Instead, you must add your filtering/modeling code to your custom observer configuration in the **Custom observer configuration** field. This code must be contained in a <servletObserver> element and added as the last child of the custom configuration's <nanoAgentConfigurations> element (the root element in the custom configuration.)
WEB_APP probe configuration	This field is displayed only if the **Specify WEB_APP probe configuration** checkbox is enabled.
	Use this field to input a <servletObserver> element into the configuration for the WEB_APP probe. This element provides control over the selective monitoring of requests and the modeling of operations.
	By default, this field contains the following code, which instructs the probe not to monitor requests for files with the specified extensions.
	<ap:servletObserver rueiPresent="true" rueiMatches="false"
	xmlns:ap="http://namespace.amberpoint.com/amf">
	<ap:globalExcludeList ext="jpg, jpeg, html, htm, css, gif,
	png, ico, js, swf, cur"/>
	</ap:servletObserver>
	For information on coding the <servletObserver> element, see Section 12.1.2.9, "Request Monitoring and Operation Modeling for the WEB_APP Probe."
Custom Observer Configuration	-
Use custom configuration	If you require observer configuration options that are not available in this policy, enable this checkbox and input your observer configuration in the following field. With this checkbox enabled, your custom configuration overrides all other fields in this policy.
Custom observer configuration	Use this field to input a custom observer configuration. This field is displayed only if the **Use custom configuration** checkbox is enabled.
Model Configuration	The fields in this section control how Business Transaction Management models particular types of components.
	Warning: You should adjust these fields to the proper setting before the observer is installed and discovers components. If you edit these settings for components that have already been discovered, you might have to modify your existing transaction definitions or reset your model (see section 10.11, "deleteAll" for information on how to reset your model).

Field Name (boldface denotes a section name)	Description
SOA	This field controls how SOA components are modeled. Choose between these options:
	Model All - Model all SOA components.
	Model Edge of Flow - Model only the first component of each SOA composite application, for example, a web service interface. This is the default setting. This setting is not applicable to SOA_B2B or SOA_JCA where all components are always modelled regardless of this setting.
Local EJB	This field controls how local EJB components are modeled. It does not affect the modeling of remote EJB components (all remote EJBs are always monitored). Choose between these options:
	Model All - Model all local EJB components.
	Model None - Do not model local EJB components. This is the default setting.
	Model Edge of Flow - Model only the first local EJB component of each local request flow.
	Model ORA-WS - Model only those local EJB components that implement business logic for Oracle Web Services (ORA-WS) web services based on the presence of any of the following annotations on their EJB implementation classes:
	■ oracle.webservices.annotations.PortableWebService
	■ oracle.webservices.annotations.PortableWebServiceProvider
	This option is designed specifically for ORA-WS, which is primarily used in Oracle packaged applications like Fusion Applications.
OSB	This field controls how Oracle Service Bus components are modeled. Choose between these options:
	Model All - Model all Oracle Service Bus components, including OSB business services, proxy services, split-join tasks, and the parallel flow of messages issuing from split-joint tasks.
	Model Edge of Flow - Model only proxy services. This is the default setting.
JMS	This field controls how JMS topics, queues, and message listeners are modeled. Choose between these options:
	Model All - Model all JMS topics, JMS queues, and their associated message listeners.
	Model Edge of Flow - Model only the message listeners associated with JMS topics. This is the default setting.
JDBC Summary	The fields in this section control JDBC Summary Mode. When JDBC Summary Mode is enabled, observations of related JDBC calls are aggregated and sent to the monitor as a single summary observation message. Enabling Summary Mode can improve your BTM system's performance and reduce its database disk space requirements, particularly if your monitored services make heavy use of JDBC calls and you have enabled message logging on these operations. These gains are achieved both by reducing the volume of observation messaging and by constraining the number of JDBC calls written to the message log database. Options are provided that allow you to control the constraints on message logging.
	Note: JDBC Summary Mode does not affect the collection, recording, or display of performance measurements, such as response time.

Field Name (boldface denotes a section name)	Description
Enable Summary Mode	Use this field to enable or disable JDBC Summary Mode. This setting is enabled by default.
Number of Slowest JDBC Calls to Log	Specify how many of the slowest JDBC calls should be logged. For example, if you specify 2, only the two slowest calls will be included in the summary observation message and then logged to the message log database (assuming that message logging is enabled for the operation). The default value is 3 calls.
Time Limit for JDBC Calls	Deprecated. This feature will only send system alerts when the specified time limit is reached. Oracle Corporation recommends that you set this time limit to be greater than the longest time expected SQL query time. Specify in seconds the longest period of time you think any JDBC call should require for completion. If any JDBC call surpasses the time limit, the summary observation message is sent immediately. This incomplete summary observation message serves as a warning that queries are taking longer to complete than you expected (the uncompleted JDBC call is indicated by a value of **time out** as its response time). If an incomplete JDBC call later completes, or other related calls are observed, then a follow-up summary observation message is sent that combines the information from the previous summary message with the information from the now completed and/or newly observed call or calls. The default value is 10 seconds.
Number of Fault Messages to Log	Specify how many fault messages should be logged. For example, if you specify 1, only the first fault message received will be included in the summary observation message and then logged to the message log database (assuming that message logging is enabled for the operation). If you specify 2, the first and last will be logged. If you specify 3, the first, second, and last will be logged (and so on). The default value is 2 faults.
Only capture summaries when the caller is configured to capture content	Normally, when an endpoint is included in a transaction and logging is enabled, messages are captured even for invocations of the endpoint from outside the transaction.
	Check this box to log message content (for fault and slowest messages) only if logging is enabled for JDBC *and* for operations calling JDBC. (Logging is enabled for operations calling JDBC either because instance logging is turned on for the transaction as a whole or because logging is turned on for the calling operation.)
	If you do not check this box, all messages sent to the JDBC endpoint will be counted even if the caller is not included in the transaction. Content will be logged for fault messages and slowest messages.
JDBC Advanced Configuration	The fields in this section provide more control over the JDBC functionality.
Show SQL Parameter	Normally, when observations of related JDBC calls are aggregated and sent to the monitor as a single summary observation message, all constant parameters are hidden. If you enable this option the parameters are displayed. Note that parameters may contain sensitive information.
Show Call Stack	If this option is enabled, the Java Call Stack is displayed. Note that enabling this option might affect performance.
Show Oracle SQL-ID	If this option is enabled, the SQL-ID for each query is displayed. This allows further analysis of any query to be performed by a database administrator.

12.1.2.12 Criteria

Use this section to choose the monitors to which this policy should apply.

12.1.3 Logging Observer Errors and Debugging Information

The observer writes error and debugging information to the following log files:

- **NanoAgentErrorTrace.log** – contains single occurrences of all errors and warnings logged to the other log files. Each error and warning entry is referenced by a unique identifier within a <Ref> element, for example:

  ```
  <Ref: Dq/QGNWqOmbdXPigC+vsO40eXgs=>
  ```

  You can use this identifier to search for all occurrences of the error or warning in the other log files, typically within NanoAgent.log. This is generally the first log file you should check when a problem occurs.

  The default size of this log file is 10M and it is recreated on each restart of the server. However, because its default rotation is set to 2, the previous log file is retained after a server restart.

- **NanoAgent.log** – contains runtime errors, configuration-related errors, and debugging information (you can adjust this logger's settings using the Enable trace logging option in the Observer Communication policy.)

- **NanoAgentPreprocessTrace.log** – contains information about bytecode instrumentation errors and debugging, class-loading, and preprocessing. This file is regenerated on each restart of the server. The maximum size of this log file is 10 MB.

  This file was renamed for release 12.1.0.2.2. For observers of previous releases, the file was named AWTrace.log.

  > **Note:** You can also configure the observer to log observed messages. For information on this topic, refer to the **Log observed messages to file** entry on page 12-27 in Section 12.1.2.11, "Advanced Settings Field Reference."

The default location of the log files is as follows:

- **WebLogic** – the *domain_root_directory*/nanoagent/logs/*server_name* directory (if that directory cannot be determined, then it defaults to the domain root directory)

- **OC4J** – the j2ee\home directory inside your SOA Suite installation directory

- **Enterprise Gateway** – the home directory (top-level installation directory) of the Enterprise Gateway server.

- **WebSphere** – the profile directory

- **JBoss** – the JBOSS_HOME/bin directory

- **WCF and ASP.NET** – the C:/temp/NanoAgentBaseDir directory

  > **Note:** The default log location for WCF and ASP.NET is not a true default. It is simply the default setting of the AmberPoint:NanoLogBaseDir key. If you set this key to null, log files will not be created.

If you want the log files generated in a different directory, set the AP_NANO_LOG_BASEDIR Java property or AmberPoint:NanoLogBaseDir Windows key. For Java application servers, you can set the property to either an absolute path or a path that is

relative to the default log directory. For Enterprise Gateway, WCF and ASP.NET, you must set the property or key to an absolute path. The following examples illustrate how to set this property or key:

- On WebLogic, if you configure your server by editing local scripts, edit the nanoEnvWeblogic script located in WL_HOME/nanoagent/bin directory. In the options section of the file, add **-DAP_NANO_LOG_BASEDIR="my_log_dir"** to the end of the NANOAGENT_JAVA_OPTIONS. This relative path would generate the log files in the directory my_log_dir under your domain directory.

  If you configure you WebLogic server using the Node Manager, open the WebLogic Administration Console, select your server, and display the **Configuration / Server Start** tab. Then add **-DAP_NANO_LOG_BASEDIR=my_log_dir** to the **Arguments** field. This relative path would generate the log files in the directory my_log_dir under your domain directory.

- On OC4J, add **-DAP_NANO_LOG_BASEDIR=my_log_dir** to the Java startup options. This relative path would generate the log files in the directory my_log_dir under the j2ee\home directory inside your SOA Suite installation directory.

- On Enterprise Gateway, open OEG_HOME/system/conf/jvm.xml in a text editor and add **<SystemProperty name="AP_NANO_LOG_BASEDIR" value="C:\OEG\my_log_dir"/>** as a child of the <JVMSettings> element. This absolute path would generate the log files in the directory C:\OEG\my_log_dir.

- On WebSphere, in the WebSphere Administrative Console, navigate to Servers > Application servers > server1 > Server Infrastructure > Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties (you might have to substitute a different server name for server1). Create a custom property named **AP_NANO_LOG_BASEDIR** and set it's value to **my_log_dir**. This relative path would generate the log files in the directory my_log_dir under your profile directory.

- On JBoss, edit your server startup script JBOSS_HOME/bin/run. In the options section of the file, add **set JAVA_OPTS=-DAP_NANO_LOG_BASEDIR="my_log_dir".** This relative path would generate the log files in the directory JBOSS_HOME/bin/my_log_dir.

- For WCF or ASP.NET, edit the application configuration file (for example, Web.config) and set the value for the AmberPoint:NanoLogBaseDir key to **C:/Inetpub/wwwroot/my_log_dir**. This absolute path would generate the log files in the directory my_log_dir under your default web site directory, for example:

```
<configuration>
 <configSections>
  ...
 </configSections>
 <AmberPoint>
  <NanoAgentDataSection>
   <add key="AmberPoint:NanoConfig"
value="c:/temp/NanoAgentLogBaseDir/nanoagentDiscovery.CONFIGURATION"/>
   <add key="AmberPoint:NanoLogBaseDir" value="c:/Inetpub/wwwroot/my_log_dir"/>
   <add key="AmberPoint:NanoCreateLogBaseDir" value="false"/>
  </NanoAgentDataSection>
 </AmberPoint>
 <system.web>
  ...
 </system.web>
</configuration>
```

In order for the observer to generate the log files, ensure that the user under which the observer is running has permission to write to the log directory. For Java observers, the user is the user that is running the application server. For IIS observers (WCF and ASP.NET), the user is as follows:

- IIS 5.x – the observer user is ASPNET

- IIS 6.x and 7.x – the observer user is NETWORK SERVICE

By default, the directory specified by the AP_NANO_LOG_BASEDIR property is automatically created if it does not exist. If you do not want this directory to be automatically created, set the property AP_NANO_CREATE_LOG_BASEDIR to **false**. In this case, you must create the directory yourself. Set this property in the same way you set AP_NANO_LOG_BASEDIR.

> **Notes:** *For Java application servers* – If the log directory does not exist and AP_NANO_CREATE_LOG_BASEDIR is set to false, runtime errors might occur and the observer might not initialize.
>
> *For IIS* – If the NanoLogBaseDir Windows key is set to null, log files are not created.

12.2 Persistent Data

This section provides information to help you administer Business Transaction Management persistent data and includes the following subsections:

- Configuring the Business Transaction Management Database Credentials

- Setting up the Message Log Database

- About Persistent Storage Directories

- Relocating Business Transaction Management Persistent Storage Directories

12.2.1 Configuring the Business Transaction Management Database Credentials

When you installed Business Transaction Management, you configured it to use an Oracle database. If the credentials used for accessing the database change, you must modify the associated setting in Business Transaction Management accordingly.

To modify the database credentials setting:

1. Choose **Admin > Edit Database Settings**.

   The **Edit Database Settings** tool opens. This tool lets you set the user name and password used by the Business Transaction Management central services to access the Business Transaction Management databases.

   > **Note:** Do not select the **Embedded Database** option. All three databases should have the **External Database** option selected.

2. Edit the user name and password as appropriate for each database.

3. Click **Apply**.

12.2.2 Setting up the Message Log Database

If you enable message logging on a transaction, then you must ensure that a database is set up for the monitors to log messages to. During installation and initial configuration of Business Transaction Management, you should have created a message log database (messageLogDB) and provided connection settings for this database. These connection settings were automatically stored in the Default Message Log Database policy and applied to all monitors.

However, you are not restricted to using a single database for message logging. You can create additional databases and configure some monitors to use one database and other monitors another database. You do this by first editing the Criteria section of the default policy so that the policy no longer applies to the monitors that will log to a different database. You then create a new policy for each new database, and use each policy's Criteria section to apply the policy to the appropriate monitors. You must take care that each monitor has only one policy applied to it. For information about creating a message log database, refer to the *Business Transaction Management Installation Guide*.

If you change the location or the logon credentials of any of your message log databases, then you must reconfigure the settings your monitors use to connect to it. You do this by editing the appropriate message log database policy.

To view the monitors to which an existing message log policy is applied:

1. Select **Administration > System Policies** in the Navigator.

2. Select the policy in the main area.

3. Click the **Targets** tab.

To edit or apply a new message log database:

1. Create or identify the Oracle database instance that you want to use for your message log database.

2. Open the message log database policy you will use for configuring the database connection:

   If you want to edit the default message log database policy:

   a. Select **Administration > System Policies** in the Navigator.

   b. Select the **Default Message Logging Database Policy** in the main area.

   c. Choose **Modify > Edit Definition for Default Message Logging Database Policy**.

   If you want to apply a new message log database policy, choose **Admin > Create System Policy > Message Log Database**.

   The **Create Message Log Database** policy opens.

3. Ensure that the Database Type is set to **Use External Oracle Database**.

   > **Note:**   The embedded database is not supported for production systems.

4. If you want to allow Business Transaction Management central services to directly access the message log database, ensure that the **Allow Central Access** option is enabled.

   If this option is disabled, the central services can access the message log database only by way of the monitor.

Some central services (such as the transaction monitoring component) require access to message content stored in the message log database. These central services can access the database either by way of the monitor or by a direct connection. Using a direct connection improves the performance of message log queries. You should enable this communication channel whenever possible.

In some deployment scenarios, you might not want the central services querying the database directly and would prefer that the monitor do so on their behalf. One such case is when the monitor and database are firewalled off from the central services. In such a scenario, the central services could communicate with the monitor, but presumably not with the database.

5. Provide a connection string and user credentials for accessing the database.

The user whose credentials you provide must have privileges to create and drop tables and indexes.

6. Adjust the Maximum Transaction Metrics Rows, if necessary.

In the beginning, you should probably leave this at the default setting.

This field specifies the maximum number of rows recorded in the temporary tables used for tracking individual transaction start and end messages prior to the computation of aggregated transaction measurements. Increasing the value allows the performance server to process transaction measurements more efficiently at the expense of more disk usage by the message log database.

7. You can edit advanced options by enabling the **Show Advanced Options** checkbox.

The following table describes the advanced options:

Advanced Options	UI Default Setting	Description
Indexer Tuning Parameters	-	-
Use Auto Statistics	enabled	Boolean
		If this parameter is enabled, the monitor gathers database statistics from the database on a regular basis. It is essential that up-to-date database statistics are maintained to allow message log queries to run efficiently. The statistics are gathered based on the number of inserts to the database that have occurred.
Log Bundle Read Batch Size	300	Integer
		Determines how many messages are processed by the indexer in a single database transaction.
Indexer Wakeup Interval	10	Integer - time (in seconds)
		Determines how often the indexer should wake up to check for any impending work.
Clean Database Check Interval	120	Integer - time (in seconds)
		Determines the interval at which the indexer performs various maintenance tasks. When performing maintenance, the indexer:
		1. Deletes expired information from the database.
		2. Cleans the summary statistics table.
		3. Removes database tables which hold the result from expired cursors.

Advanced Options	UI Default Setting	Description
Clean Cursors Check Interval	3600	Integer - time (in seconds)
		Determines the interval at which the indexer will remove expired query results from the database.
		Although this task is part of the indexer's normal maintenance, this may need to be done more often than other tasks.
Stop Indexing	disabled	Boolean
		If set to true, this option tells the indexer to suspend all activity. Content to be indexed will still be captured by active logging policies, but will not be transferred from on-disk storage into the database until indexing is resumed.
		This option is especially useful during times of heavy message traffic, when optimization of resources and a steady flow of traffic is more important than being able to inspect indexed messages. You can later set the Stop Indexing value to false to allow Business Transaction Management to index the messages and enter them into your database.
		Note: Be aware that during the time the indexer is suspended, Business Transaction Management does nothing to manage the disk space being used. It is up to you to make sure that there is enough empty disk space to capture messages being logged by logging policies.
Database Error Min Delay	10	Integer - time (in seconds)
		Specifies the minimum amount of time the indexer will wait before retrying logging-related database operations when a database error occurs. On each successive failure, the delay will be adjusted upward by multiplying the current delay by the value of the Database Error Delay Expansion Factor parameter. The maximum wait time between retries is bounded by Database Error Max Delay.
		An example of a database error that this parameter applies to would be the monitor being unable to contact the database. For example, at the default settings, if the monitor loses its connection to the database, it will attempt to reconnect after 10 seconds. If it cannot reconnect, it will wait 20 seconds and try again, and so on. The longest it will wait between attempts is 3600 seconds (1 hour).
Database Error Max Delay	3600	Integer - time (in seconds)
		See description for Database Error Min Delay.
Database Error Delay Expansion Factor	2.0	See description for Database Error Min Delay.
Max Messages Indexed per Bundle Run	5000	Integer
		Limits the maximum number of messages indexed for a particular endpoint on each indexer run. All endpoints in a single monitor are indexed by a single worker.
Maximum Indexer Query Execution Time	300	Long - time (in seconds)
		Specifies an upper-bound time limit on the run time of any indexer-initiated query.
Maximum Query Execution Time	30	Long - time (in seconds)
		Specifies an upper-bound time limit on the run time of any user-initiated query. Users may initiate long-running queries against the message log. Once submitted, users do not have a way to cancel the query and must wait for it to complete.
		The default value for this parameter is 30 seconds. Setting this value to 0 allows all queries to run to their completion regardless of their complexity. For this reason, this setting (0) is not recommended.

Advanced Options	UI Default Setting	Description
Num Indexer Worker Threads	3	Long
		Specifies the number of worker threads used by the log policy indexer. The indexer cycles through the endpoints with applied logging policies and indexes each endpoint in turn. Adding threads allows for more endpoints to be indexed concurrently.
Metadata Insert Batch Size	300	Long
		Controls metadata insert statements. This parameter specifies the number of rows of a particular type to batch together before running a SQL statement. The actual batch size is also influenced by the Log Bundle Read Batch Size parameter because it sets the maximum transaction size.
Message Insert Batch Size	30	Long
		Controls message insert statements. This parameter specifies the number of rows of a particular type to batch together before running a SQL statement. The actual batch size is also influenced by the Log Bundle Read Batch Size parameter because it sets the maximum transaction size.
Num User Query Connections	5	Long
		Specifies the number of connections to the message log database that should be created for the purpose of user queries. The pool is a shared pool and consists of connections created for system processing (controlled by Num Indexer Worker Threads) and connections for user queries (controlled by Num User Query Connections).
Reuse Tables	disabled	Boolean
		The Rotation Interval setting in the Message History policy controls how long messages are retained in the database. By default, messages are deleted by deleting tables and added by adding tables. Enable this setting if you want to reuse tables rather than delete and create new tables. The tables are cleared before being reused. In most scenarios, it is more efficient to leave this setting disabled.
Min Entries per Fragment	0	Long
		Messages are stored in sets of tables, called fragments. This setting specifies a minimum number of messages a fragment must have before being rotated. This constraint is in addition to that of the Rotation Interval setting in the Message History policy. Note: a request/response pair is considered to be two messages.
Indexer Setup Data Version	-	-
label.IndexerSetupData.generateEndpointStatistics	disabled	In general, you should enable this field only if requested to do so by the Oracle support team.
		If a monitor is managing an endpoint that participates in a transaction, then the monitor will be running the message indexer (the out-of-band indexer) for the purpose logging. When the message indexer is running in a monitor, the monitor's Status tab includes information about the performance of the message indexer. By default, the tab displays summary indexer statistics for all the endpoints for which message indexing is active. If you enable this setting, the tab's indexer statistics include detailed performance information for each endpoint participating in message indexing.

8. In the **Criteria** section, choose the monitors that will log to the database.

> **Note:** Take care that you do not apply more than one message log database policy to any single monitor. This means that if you are applying a new message log database policy, you must first edit the Criteria section of your existing policies so that they don't apply to the same monitors as your new policy. If you apply more than one message log database policy to a single monitor, Business Transaction Management generates a system alert.
>
> All monitors in a monitor group must log to the same message log database.

9. Click **Apply**.

12.2.3 About Persistent Storage Directories

At initial startup, Business Transaction Management creates a set of persistent storage directories to collect system output log entries and store user preferences for the system deployments. By default, the persistent storage directories are created within the application server's installation directory at *WL_install_dir*/user_projects/domains/*domain_name*/servers/*server_name*/btmstorage/*.

Your company's in-house procedures and rules for persistent storage might require you to place the persistent storage directories in a different location. In such a case, you can reconfigure the location of the persistent storage directories.

An installed Business Transaction Management system is composed of a set of deployments (EAR files), which are themselves composed of subdeployments (WAR files). Each subdeployment has an associated persistent storage directory of the same name, minus the ".war". The following table lists the names of the deployments, subdeployments, and persistent storage directories.

Table 12–2 Business Transaction Management deployments, subdeployments, and persistent storage directories

Deployments (EARs)	Subdeployments (WARs)	Persistent storage directories
btmMain	btmui	btmui
	btmcentral	btmcentral
	btmcontainer	btmcontainer
btmPerformanceServer	btmcontainer	btmcontainer
	btmperformance	btmperformance
btmTransactionServer	btmcontainer	btmcontainer
	btmtransaction	btmtransaction
btmMonitor	btmmonitor	btmmonitor

12.2.4 Relocating Business Transaction Management Persistent Storage Directories

This topic explains how to change the default location of the persistent storage directories for Business Transaction Management deployments to a location outside of the container that hosts these deployments.

This topic contains the following subsections to guide you through the steps required to relocate persistent storage directories:

- Backup Before Relocating Persistent Storage Directories

- General Instructions for Relocating Persistent Storage Directories

- Detailed Instructions for Relocating Persistent Storage Directories

12.2.4.1 Backup Before Relocating Persistent Storage Directories

Before following the procedure for relocating persistent storage directory locations, it is very important that you backup any persistent storage directories that already exist in the default location in your container. These default persistent storage directories are created the first time you start up your Business Transaction Management deployments, and are listed in the sections below for each container. You will later need to copy the contents of these directories to the new location you have defined for each deployment's persistent storage directory.

If you do not backup and remove the existing persistent storage directories, the settings in your new persistent storage directories might not be loaded and used the next time you restart Business Transaction Management. By default, Business Transaction Management references the default locations for the deployments' persistent storage directories. If the default directories still exist after you have set their new location, the new location might not be recognized. User preferences are also contained within these storage directories. Business Transaction Management reads these user preference files on each restart.

The default location of the persistent storage directories is *WAS_install_dir*/profiles/*profile_name_name*/btmstorage/*node_name*/*server_name*/*.

You should document where you relocate your persistent storage directories because you will have to define their location again if you redeploy Business Transaction Management applications (for example, during an upgrade). It is also important to document your new persistent storage directory locations if you want to use the LogMerger tool to collect and merge output of system log messages from these locations. It is easiest to create a configuration file for the LogMerger tool, as that will also act as a documentation source for your new persistent storage directory locations. For more about the LogMerger tool and creating a configuration file for the tool, see Section 11.3, "logMerger utility."

12.2.4.2 General Instructions for Relocating Persistent Storage Directories

The following steps outline the general instructions to relocate the persistent storage directories.

1. Shutdown your Business Transaction Management deployments.

2. Backup your persistent storage directories and place them in a safe location.

3. Modify persistent storage directory locations in each deployment's web.xml file.

4. Move backup copies of the persistent storage directories to the new persistent storage directory locations.

> **Note:** If you do not plan to use the information already collected in the persistent storage directory in the new location, you must create an empty persistent storage directory in the new location using the same name as the original storage directory.

5. Package and redeploy, if required, each deployment whose persistent storage directories you want to relocate.

6. Restart your deployments.

7. Confirm new system output log (logdir) entries in the new locations.

12.2.4.3 Detailed Instructions for Relocating Persistent Storage Directories

The Business Transaction Management deployments can be found in the following directory locations. You will need to locate these deployments in order to edit the location of the persistent storage directory in each deployment's web.xml file:

```
WL_install_dir/user_projects/domains/domain_name/server_name/
.wlnotdelete/extract/server_name_n_n/public/btmstorage/*
```

For example, the btmcentral deployment is located in this directory:

```
WL_install_dir/user_projects/domains/domain_name/server_name/
.wlnotdelete/extract/server_name_btmcentral_
btmcentral/public/btmstorage/btmcentral
```

To relocate persistent storage directories:

1. Shut down your Business Transaction Management deployments.

2. Backup the contents of the current default persistent storage directories and place them in a safe place.

3. Modify persistent storage directory locations in each deployment's web.xmlfile:

   a. Locate the exploded war file for the deployment whose storage directory you want to change.

   b. From the exploded war file, open the deployment's WEB-INF/web.xml file in a text or XML editor.

   c. Set the new location for the persistent storage directory by setting the storageDirectory parameter value for that deployment as follows:

   Edit the *AmberPointDefault* value in the following lines and set it to the location of the new storage directory:

```
<!-- PERSISTENT STORAGE DIRECTORY To set the persistent storage area to
some value, change the value of param-value to some EXISTING directory
where you want things stored. -->
<context-param>
<param-name>com.amberpoint.storageDirectory</param-name>
<param-value>AmberPointDefault</param-value> </context-param>
```

> **Note:** You must not change the names of the persistent storage directories. You may change only the path to the directories.

Examples:

- **On Windows systems** – if you want the persistent storage directory for btmcentral to be in `C:\btm_data\btmcentral`, change the default entry within your btmcentral web.xml file to the following:

```
<context-param>
<param-name>com.amberpoint.storageDirectory</param-name>
<param-value>C:\btm_data\btmcentral</param-value> </context-param>
```

- **On Unix-like systems** – if you want the persistent storage directory for btmcentral to be in opt/webserviceapplogs/btm_data/btmcentral, change the default entry within your btmcentral web.xml file to the following:

```
<context-param>
<param-name>com.amberpoint.storageDirectory</param-name>
<param-value>/opt/webserviceapplogs/btm_data/btmcentral</param-value>
</context-param>
```

4. Create the new empty persistent storage directory in the new location (if you want to start from scratch), or move the backup copy of the original persistent storage directory to the new directory location.

5. If required, undeploy and redeploy each deployment whose persistent storage directories you want to relocate as follows:

> **Note:** When repackaging system deployments, make sure to include the manifest file associated with the deployment, as this file contains important information required for deployment.

   a. Package the new deployment that includes the edited web.xmlfile into a new application war file.

   b. Undeploy the existing deployment using the WebLogic Console.

   c. Shut down the WebLogic server and delete the original persistent storage directory.

   > **Note:** You must delete the persistent storage directories from their default locations. If the deployments find persistent storage directories in their default locations, they will ignore the new directory locations.

   d. Restart WebLogic and redeploy the new system deployment using the WebLogic Console.

6. Restart your deployments.

7. Confirm new system output log (logdir) entries in the new locations.

   Data should now be written to the persistent storage directory locations you defined in each deployment's web.xml file. Check to make sure new system service log files (logdir) and other directories have been created in the new location upon container startup.

   If you use the `logMerger` tool to merge system service logs, make sure that you refer to the new persistent storage directory locations when merging log files.

12.3 Business Transaction Management System Security

This section provides information to help you administer Business Transaction Management system security and includes the following subsections:

- Authentication and Role Mapping

- Business Transaction Management Application User Roles

12.3.1 Authentication and Role Mapping

Business Transaction Management relies on the WebLogic server in which it is deployed for authentication and association of roles with users. By default,

authentication is enabled for the Management Console. To disable authentication, use whatever tool or procedure is appropriate for the application server you are using.

> **Note:** In order to log into the Management Console, you must use credentials that are mapped to at least one of these Business Transaction Management user roles: btmAdmin, btmUser, or btmObserver.

If you disable authentication, users of the Management Console must still log in. However, they can log in using any user name and are not required to provide a password. Note that all UI personalizations, such as edits to the Navigator, filters, and column sets are stored as preferences and associated with the user name.

This topic describes how the supported application servers authenticate users and map them to Business Transaction Management application roles.

12.3.1.1 Summary of Initial Application Role Mapping

Business Transaction Management Role	WebLogic Group
btmAdmin	WebLogic Administrators
btmUser	WebLogic Operators & Monitors
btmObserver	Everyone
btmInspector (Note that the role name is singular while the group name is plural.)	btmInspectors

> **Note:** The role btmInspector is, by default, mapped to a group named btmInspectors, but the application server administrator must create a group named btmInspectors and assign it to the appropriate users.

12.3.2 Business Transaction Management Application User Roles

Business Transaction Management uses roles to authorize access to various parts of the user interface.

12.3.2.1 Primary Roles

Each user must be assigned at least one primary role. The primary roles are:

btmAdmin – users with this role are granted all privileges. These users can use all tools and facilities provided by the Business Transaction Management Console, including the ability to view and create sensitive properties and to view all message content.

btmUser – users with this role have most privileges needed to configure basic monitoring. For example, they can configure monitors; create, edit, and delete policies (does not include system policies); register services; set registry attributes on services and endpoints; and create and edit transactions and conditions. They also have all the

privileges of btmObserver. This role does not grant the privilege to modify the Business Transaction Management environment, access message content, or view or edit sensitive properties.

btmObserver – users with this role have privileges to use most of the basic monitoring facilities. They can view summary, dependency, and administrative information about the monitoring system, but are not allowed to configure any of the policies or settings related to it. They can also view transactions and conditions, but are not allowed to create or edit them. This role does not allow users to modify the Business Transaction Management environment, access message content, or view or edit sensitive properties.

> **Note:** All navigation and views in the Management Console are available to all primary roles. However, some roles cannot access certain menus and menu items and the tools associated with them.

12.3.2.2 Auxiliary Role

In addition to the primary roles, Business Transaction Management defines an auxiliary role. The auxiliary role provides additional privileges that you might want to assign certain users. For example, you might want to let a user access message content but not want to give that user full administrative privileges. You could do this by assigning the user a primary role of btmUser and an auxiliary role of btmInspector. The auxiliary role is:

btmInspector – users with this role can view message content and view and create properties, including sensitive properties.

> **Note:** The btmAdmin role has all of the privileges of btmInspector.

12.4 Backing up and Restoring Business Transaction Management

The following sections explain how you back up and restore your system. The topics covered include the following:

- About the Back up and Restore Process
- Backing up Business Transaction Management
- Restoring Business Transaction Management
- Data Storage Reference

12.4.1 About the Back up and Restore Process

Oracle Business Transaction Management stores a large amount of data. This data describes the system's configuration, what the system is monitoring, and the current and past states of monitored applications. All of this data is needed for the operation of the system; if something happens that causes this data to be lost or damaged, the system can no longer perform as you expect. This is why it is important to create a backup of the system's data and to be able to recover this data.

You might need to back up Business Transaction Management for different reasons:

- on a regular basis to enable recovery from unforeseen events
- before migrating to a new sphere

- before upgrading an application server in the Business Transaction Management environment or adding an application server

- before installing a new version of Business Transaction Management

This section offers general guidelines for backup and recovery, and suggests milestones for testing the process you have defined. How often you create a checkpoint by backing up your data depends entirely on the lifecycle stage of your application and on business requirements.

Backing up and restoring Business Transaction Management does not include the backup and recovery of the hosting application server and its configuration settings, some of which Business Transaction Management needs to function properly: JVM settings, Java System parameters, and so on. You should already have processes in place for backing up your application servers and their configurations.

12.4.1.1 Before You Back Up

Business Transaction Management operates in a complex environment. For this reason, before backing up, it is important to make sure that you can isolate Business Transaction Management components and that you can identify any other systems that might be affected by the backup and recovery process. Consider issues like the following:

- Databases might be shared with components other than Business Transaction Management. Unless the problem is database failure itself, it is important to restore only those database instances that are used by Business Transaction Management.

- Recovery might affect other systems. For example, if Business Transaction Management shares JDBC drivers with other applications, recovery might restore a driver to a previous version and cause other applications using the driver to fail.

12.4.1.2 Testing the Backup and Recovery process

You should test your backup process periodically by attempting a recovery and making sure the system can be brought up to the desired state with no side effects. Identifying and resolving problems with the backup process will ensure successful recovery when recovery matters. Your backup verification checklist should include things like the following:

- database and file system structure, and permissions are as expected

- Business Transaction Management is functional and in the expected state: the console shows everything is running, services are reachable, traffic flows normally, and so on.

- no application sharing the same resources is adversely affected.

12.4.2 Backing up Business Transaction Management

This section describes how Business Transaction Management data is organized, explains how you back up each type of data, and discusses timing issues related to backups.

12.4.2.1 How Business Transaction Management Data Is Organized

The next figure shows the various kinds of Business Transaction Management data and the Business Transaction Management system services that rely on this data.
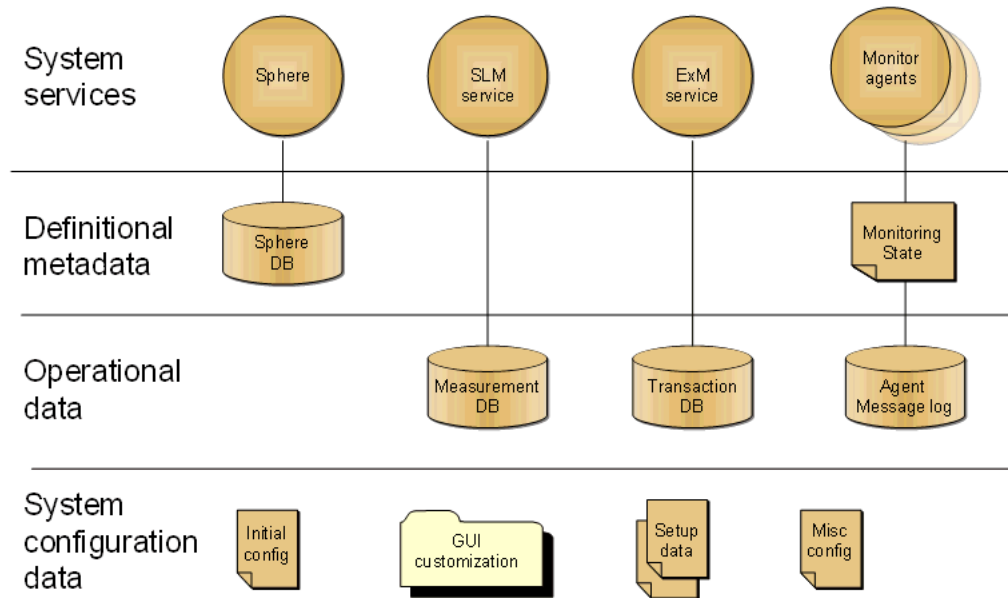
With reference to the figure, the basic principle of backing up data is as follows:

- All data contained in databases is backed up by backing up the database.

- All data contained in files or directories is backed up by backing up the `btmstorage` directory, which can be found on every host where one of the Business Transaction Management system services or monitors is deployed. The location of this directory for your server is specified in Section 12.4.2.2, "Backing up Business Transaction Management Data."

The rest of this section provides more information about elements shown in the previous figure. You do not need to know this level of detail just to do backup and recovery. But this detail might be helpful in troubleshooting and in understanding the resources used by Business Transaction Management. If you want, you can skip ahead to Section 12.4.2.2, "Backing up Business Transaction Management Data."

As the figure shows, Business Transaction Management is composed of multiple system services:

- The sphere, responsible for the overall operation of Business Transaction Management and coordination of its member services

- The SLM service, responsible for gathering performance measurements

- The ExM service, responsible for transaction management

- Monitor agents, responsible for collecting data from observers

Each of these services depends upon data that specifies the system's configuration, describes what it is monitoring, and records the state of monitored applications. This data can be grouped into the three categories shown in the figure.

- **Definitional metadata** is stored in two places and contains the following information:

  The Sphere database contains data that describes Business Transaction Management as well as the monitored user systems. It includes a description of the users' applications, the policies used to monitor them, and transaction definitions.

Monitor agent configuration files contain data that describes whether and how each user endpoint is being monitored.

- **Operational data** is the information Business Transaction Management gathers about user applications: performance and behavioral metrics, logged messages, transaction instances, and generated alerts. This information is stored in the Measurement, Transaction, and Agent Message log databases shown in the figure.

- **System configuration data** controls the basic behavior of Business Transaction Management: what databases it connects to, the address a container should use to connect to the sphere, default GUI views and layout. This information is saved in various configuration files: initial configuration data, GUI customization, setup data, container registration, and miscellaneous configuration files.

12.4.2.2 Backing up Business Transaction Management Data

Backing up Business Transaction Management is fairly simple: you back up data contained in databases by backing up the respective database; you back up data contained in files or directories by backing up the `btmstorage` directory.

The `btmstorage` directory can be found on every host where one of the Business Transaction Management system services or monitors is deployed at this location:

WebLogic_InstllDir/user_projects/domains/*MyDomain*/servers/*MyServer*/btmstorage

Once you have backed up the databases and the `btmstorage` directory, you are done with the backup process.

In general it is best to back up and recover all data, even if only a subset of your data has been damaged or lost. However, if you would like a more detailed understanding of the individual components used by Business Transaction Management, see Section 12.4.4, "Data Storage Reference."

12.4.2.3 Timing Backups

The timing of backups is important: you should back up the databases and the `btmstorage` directory as close together in time as possible. If possible, follow these guidelines:

1. Quiesce the system if possible.

2. Back up the `btmstorage` directories.

3. Back up the databases, with the Sphere data last.

12.4.3 Restoring Business Transaction Management

The goal of restoring Business Transaction Management is to bring it back to the desired state with no side effects. Before you start this process, make sure that you have complete and accurate information about the Business Transaction Management system you are trying to restore.

It is assumed that you are restoring Business Transaction Management to the same environment from where it was backed up. If you need to recover to a different environment, for example, in the case of hardware failure; you will need to change the host name of the machine where you restore to (at the operating system level) to the host name of the machine that failed. You will also need to make sure that Business Transaction Management services hosted on the new machine can run on the same ports as on the old machine. It will then be possible to recover services to the new machine without disruption.

The restore procedure recovers the whole system to the last checkpoint created by the backup process.

> **Note:** After the restore, the database schema and the file system must reflect the state they were in at the time of the backup. To make sure this happens, before you restore, check that the existing database and storage directory is completely clean. Because the data in the two storage locations are connected in various ways, problems can arise if either holds data that is newer than the backed up data. Thus, you should never restore a backup on top of an existing `btmstorage` directory. Most database restores take care of this issue; be sure yours does.

The restore procedure consists of two steps:

- Restore databases
- Restore the `btmstorage` directory on each server hosting a system service or monitor.

In the case where there is some damage to the Business Transaction Management software itself because something has damaged or corrupted the installed instance, we recommend that you do the following.

1. Reinstall the Business Transaction Management software.
2. Restore the `btmstorage` directory on each server hosting a system service or monitor agent.
3. Restore the databases.

> **Note:** If the damage affects only the EAR, WAR, or JAR files themselves, a simple re-installation of the Business Transaction Management software is all that is required

12.4.4 Data Storage Reference

The following table offers some additional detail about the Business Transaction Management components. This detail might be helpful to understand the role of each component or to locate specific information.

Data	Contents	Backup Procedures
Sphere database	Description of Business Transaction Management system, monitoring and logging policies, transaction definitions, user application definitions.	Use the backup features of the Oracle database to create a backup.
Monitoring state	Information about whether and how each user endpoint is being monitored. This data is also replicated in the sphere database; however the monitor agent's configuration file is considered the master source for this information. Although monitoring state data is backed up when you back up the sphere database (and restored when you restore it), that copy does not count, and if you recover an agent without capturing its original monitoring state, your endpoints will end up unmonitored.	Back up the `btmstorage` directory.
Operational data	Information Business Transaction Management gathers about user applications. This data is stored in the Performance Manager's database, the Transaction Manger's database, and the message log database. These might be located on the same physical database, but they are considered to be distinct databases.	Use the backup features of the Oracle database to create backups.

Data	Contents	Backup Procedures
Initial configuration data	By default, information gathered from the user's initial configuration of Business Transaction Management is saved in the file `essentialConfiguration.xml` in the directory *WebLogic_InstallDir*/user_projects/domains/*MyDomain*/servers/*MyServer*/btmstorage/globalPreferences *WAS_InstallDir*/profiles/*MyProfile*/btmstorage/*MyNode*/*MySrvr*/*DeployEarFileName*/globalPreferences This information includes the location of databases used by Business Transaction Management, deployment credentials, and database type.	Back up the `btmstorage` directory.
UI customization	Information about customizations done by the administrator and preferences and views created by the user. By default, this information is stored in files in the following directories: *WebLogic_InstallDir*/user_projects/domains/*MyDomain*/servers/*MyServer*/btmstorage/btmui/userPreferences *WAS_InstallDir*/profiles/*MyProfile*/btmstorage/*MyNode*/*MySrvr*/*DeployEarFileName*/btmui.war/userPreferences	Back up the `btmstorage` directory.
Monitor registration	Registration information about monitor agents that you have added to the system.	Back up the `btmstorage` directory.
System Service setup	Setup data for each of the Business Transaction Management system services.	Back up the `btmstorage` directory.
Miscellaneous scripts and configuration	In the course of configuring Business Transaction Management, you might create various configuration scripts: for example, scripts to configure email subscriptions known to the notifier service or scripts to set up baseline performance values.	Back up the `btmstorage` directory. If you have stored scripts anywhere else, back up that directory as well.

12.5 Migrating Data

This section summarizes the steps required to migrate data from one environment to another. For example, you might need to do this when you migrate from a testing to a production environment.

To migrate data from one environment to another, do the following:

1. Always backup your system before you migrate data. For information about backing up your system, see Section 12.4, "Backing up and Restoring Business Transaction Management."

2. Use CLI commands to export your data as follows. It does not matter in which order you do the export:

   - Use the `exportProfile` command to export property definitions. If you have no properties defined, you do not need to do this.

   - Use the `exportTransactionDefns` command to export transaction definitions.

   - If you do not use autocreation of business objects, use the `exportBusinessObjects` command to export consumer data.

   - If you have created downtime schedules you plan to use in your new environment, use the `exportSchedules` command to export these.

   - Use the `exportPolicies` command to export policy definitions.

     You only need to export user-defined policies. These are most likely to be SLA policies. You do not need to, and should not, export system policies or policies

created for the transaction. System policies relate to the configuration of your environment, which is likely to differ in the new environment. System-generated policies for your transaction will be automatically re-created when you import the transaction definition.

3. Install and configure your new BTM environment.

   Make sure that the new system has an opportunity to observe everything in the production environment that you used to create the transactions you exported. If you know what services your transactions include, you can use the BTM console to look for the services you need and to make sure that they have been observed.

4. Use CLI commands to import your data in the order shown:

   a. Use the `importProfile` command if you have exported any profiles.

   b. Use the `importTransactionsDefns` command to import transaction definitions.

      If an import fails, this is most likely due to the fact that the system has not observed a service that is included in your transaction. You can run more traffic, and then try the import again.

   c. Use the `importBusinessObjects` command if needed.

   d. Use the `importSchedules` command if needed.

   e. Use the `importPolicies` command to import policy definitions.

Your transactions should now be functional in your new environment. You should run some traffic to make sure that transaction monitoring is working as expected.

12.6 Setting up Load Balancers

Setting up load balancers allows Business Transaction Management to model the flow of traffic correctly and allows you to access the load balancer's administrative console from the Business Transaction Management console.

This section also explains how you set up an F5 device to load balance messages from one observer to multiple monitors. It includes the following sections:

- Setting up a Load Balancer
- Configuring an F5 Device to Work with Replicated Monitors

12.6.1 Setting up a Load Balancer

If you deploy a service in more than one container, Business Transaction Management understands these replicated endpoints are part of the same service, and it can infer the existence of a load balancer that routes messages to these replicated endpoints. That is, Business Transaction Management can model the flow of traffic correctly in dependency diagrams even though it does not monitor the flow of traffic through the load balancer itself. However, without your help, Business Transaction Management cannot provide more detailed information about the inferred load balancer. Setting up a load balancer means giving Business Transaction Management information that allows it to do the following:

- provide information about the load balancer in the management console; for example, the name you want displayed for it and the vendor associated with it
- identify the device hosting the load balancer
- give you easy access to the load balancer's administrative console

■ specify the lifecycle phase of the load balancer device and all the endpoints that are created within it

Setting up a load balancer starts with registering it, which you can do using either the CLI command registerDevice or using the management console. In some cases, you might also need to specify an entry point to the load balancer and define target entry points that correspond to the destinations where messages are being routed.

This section explains some basic terms related to load balancing, describes the devices that Business Transaction Management supports, and explains the following user tasks:

■ Registering a load balancer

■ Modifying information about a load balancer

■ Adding entry points to show routing relationships

■ Unregistering a load balancer

Business Transaction Management supports a variety of load-balancing devices. It provides the greatest support for F5 load balancers, but it can also recognize and model other hardware and software load balancers.

12.6.1.1 Basic Terms

The figure below illustrates the use of a load balancer to route messages A and B to three replicated endpoints (E1, E2, E3). Note the elements marked: *routing entry point* and *target entry point*. The load balancer receives messages at the routing entry point and forwards them to the target entry points. There are situations in which you might have to supply entry point information after registering the load balancer, as described in the next section.
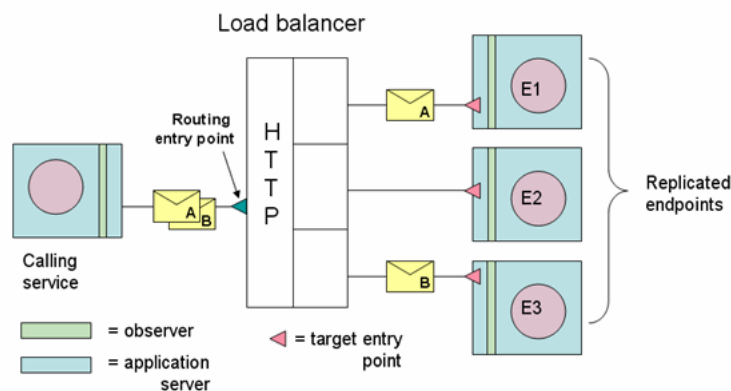
12.6.1.2 Supported Devices

Business Transaction Management can work with three kinds of load balancers: F5 devices, other hardware devices, and software load balancers. The work you need to do to help Business Transaction Management model message traffic varies with each case:

■ Business Transaction Management knows the most about the F5 device and requires only a single registration step to derive the information it needs. Business Transaction Management might perform additional discovery passes to complete its picture of an F5's role in message routing.

- For hardware load-balancers, Business Transaction Management can usually detect and model any routing entry points automatically, based on the information in the HTTP Host headers of the observed messages. However, explicitly registering the device (using the CLI command `registerDevice` or using the management console) allows Business Transaction Management to display information about the device's location and other attributes.

  If you do not register any device, Business Transaction Management automatically registers a *default load balancer* and is able to model the flow of messages through this device. In this case, you can still edit the Profile page for the default load balancer to specify a name, the base address, admin UI, and vendor.

  If the observed messages do not carry information about their original recipient (the load balancer) in the HTTP Host headers, you will need to register the device and specify routing and target entry points in the same way as you do for software balancers, described next.

- For load balancers implemented by software that make a separate HTTP connection to the back-end servers (rather than just forwarding HTTP messages), you need to describe the routing relationships in order for Business Transaction Management to model them correctly. To do this you must register the device, add an entry point for the load balancer, and specify target entry points that correspond to the destinations where messages are being routed.

12.6.1.3 Default Load Balancer

The default load balancer is either the first load balancer registered or the one you set to be the default using the `setDefaultLoadBalancer` command.

The calling service uses a routing entry point to communicate with the load balancer. Business Transaction Management discovers the routing entry point by observing messages. If no load balancer has yet been registered, Business Transaction Management creates a default load balancer and assigns the discovered routing entry port to it. Any newly discovered routing entry points will be modeled as part of the default device unless they belong to a registered F5 load balancer.

You can edit the profile of the default load balancer that was created for you, to provide additional details or, if you prefer, you can unregister it and register your actual load balancer explicitly.

12.6.1.4 Registering a Load Balancer Using the Management Console

You can register a hardware or software load balancer using the CLI command `registerDevice` or using the console.

After you complete the registration, the device is listed in the summary pane when you select **Explorer > Devices** in the navigator.

To register a hardware or software load balancer using the console:

1. Select **Admin > Register > Device** and then choose **F5 Networks** to register an F5 load balancer, or choose **Other** to register any other load balancer.

2. In the ensuing dialog, specify values as shown in the following table.

Field	Description
Vendor	Not editable for F5; required for other load balancers.
	If you chose **F5 Networks** in Step 1, this field is set to the non-editable value of **F5**.
	If you chose **Other**, specify the name of the vendor of your load balancer. This field is purely descriptive. You can specify any value (except F5). The specified value is displayed in the Management Console.
Device Name	Required.
	Specify the friendly name for your load balancer. This name is displayed in the Management Console.
Notes	Optional. Add any notes to remind you of the nature or purpose of this load balancer.
Lifecycle phase	Select the lifecycle phase from the drop-down list. Available values are `deprecated`, `development`, `production`, `staging`, and `test`. These are case insensitive.
Configuration URL	Required and displayed only if you chose F5.
	Specify the URL of the F5 console in the following format:
	https://*managementPortIP*/iControl/iControlPortal.cgi
	Replace *managementPortIP* with the appropriate host name and port number. This URL normally ends with iControl/IControlPortal.cgi.
Username and Password	This value is required and displayed only if you chose **F5 Networks**.
	Specify the user name and password of an account on your F5 load balancer. A user role of Guest provides sufficient privileges.
	You can encrypt passwords using the `encryptPassword` CLI command, for example:
	`btmcli encryptPassword -password "myPassword"`
Base Address	Required and displayed only if you chose **Other**.
	Specify the base address of the URL for your load balancer, for example:
	https://myLoadBalancer:443/
Administrator URL	Optional and displayed only if you chose **Other**.
	Specify the URL of your load balancer's HTML administrative console. A link to this URL is displayed in the Business Transaction Management Management Console to provide easy access to your load balancer's console.
	This flag is not needed for F5 load balancers because Business Transaction Management obtains the URL automatically.

3. Click **Apply**.

4. If needed, assign routing entry points and target entry points as described in the .

12.6.1.5 Modifying Information About a Load Balancer

You can modify information about a device you have already registered or about a default device.

To modify information about a device:

1. Select **Explorer > Devices** from the navigator.

2. In the summary area, select the device whose attributes you want to specify or edit.

3. Select **Modify > Edit Profile for** *deviceName***.**

4. Modify the fields of interest (as described when registering the device) in the ensuing dialog.

5. Click **Apply**.

12.6.1.6 Adding Entry Points to Show Routing Relationships

In most cases, Business Transaction Management automatically detects and models routing relationships by observing message traffic and reading destination information from the message headers. However, if the observed messages do not carry information about their original recipient (the load balancer) in the HTTP Host header, you will need to manually create a routing entry point to the load balancer. You will also need to add target entry points to indicate where the messages are being routed.

If you do not specify routing relationships, Business Transaction Management will not be able to draw contiguous dependency flows. In the case of transactions, you could still connect these disjoint flows by linking related services using manual keys.

To add routing entry points and target entry points:

1. Select **Explorer > Devices** from the navigator.

2. In the summary area, select the device whose routing relationships you want to clarify.

3. Select **Create > Entry Point for** *deviceName***.**

4. In the ensuing **Create Entry Point** tool, specify the following information: --In the **Hosted on** section, specify the IP address and port number where the load balancer is receiving observation messages (the HTTP port). -- Click the **Add Target Entry Point** and choose a destination from the drop down list. Each destination refers to a target entry point where the load balancer is routing messages. Do this for each potential destination.**Note**: There will be more entry points on the drop-down list than the router is using. Some of these might be addresses to which another load balancer is sending messages. (Basically, the drop down list shows every entry point known to the sphere.)

5. Click **Apply**.

12.6.1.7 Unregistering a Load Balancer

You can only use the management console to unregister a load balancer.

To unregister a load balancer:

1. Select **Explorer > Devices** from the navigator.

2. In the summary area, select the name of the device you want to unregister.

3. Select **Modify > Delete** *deviceName* **Registration**

4. Confirm deletion action by clicking **Delete** in the next dialog.

12.6.2 Configuring an F5 Device to Work with Replicated Monitors

Monitoring in Business Transaction Management relies on the communication that takes place between an *observer* that monitors message traffic through a given service and a *monitor* that analyzes and stores the data obtained by the observer.

To scale your system and make it fault tolerant, you can associate several replicated monitors (monitor group) with observers. Replicated monitors require a third-party load balancer that can route messages from observers to the monitors. This section explains how you set up an F5 device to load balance messages from an observer to two or more monitors.

In order to understand F5 setup, it is helpful to review the mechanism that allows an observer to communicate with a single monitor; this is illustrated in the following figure:
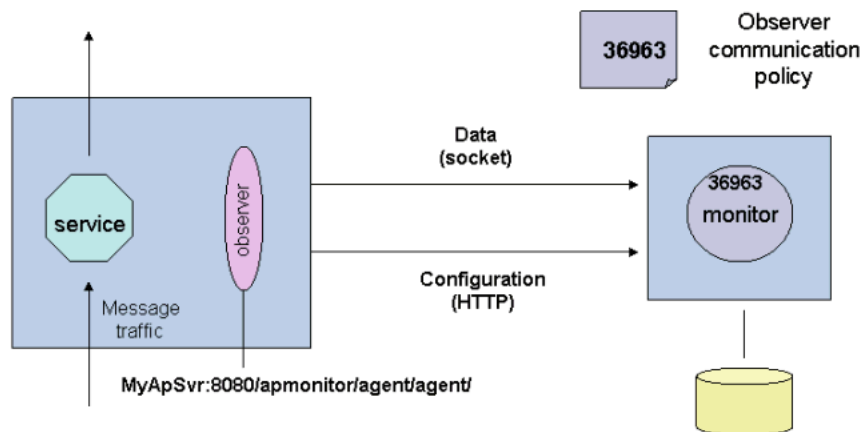
As shown in the figure, communication between the observer and the monitor proceeds by means of two paths:

■ The observer queries the monitor for configuration information and receives that information from the monitor's HTTP port. (Although configuration data is mostly flowing from the monitor to the observers, the connection is made by the observer.) This port is specified using the AP_NANO_CONFIG_URL Java system property or using the `AmberPoint:NanoConfigRL` Windows key, depending on the platform. The sample URL shown in the figure is `MyApSvr:8080/apmonitor/agent/agent`. When the observer starts up, it sends a request to this URL to get configuration information, which tells it what it should measure and how often. In this way, the observer can be reconfigured dynamically as your need for different kinds of information changes.

■ The observer sends measurement data to the monitor at the socket port specified in the **Observer communication policy**. By default, this port is **36963**.

When you set up communication between your observers and replicated monitors by way of an F5 device, the device must be configured to include these same two (configuration and data) paths for every replicated monitor you add. The next figure shows how the F5 device connects an observer with the replicated monitors.
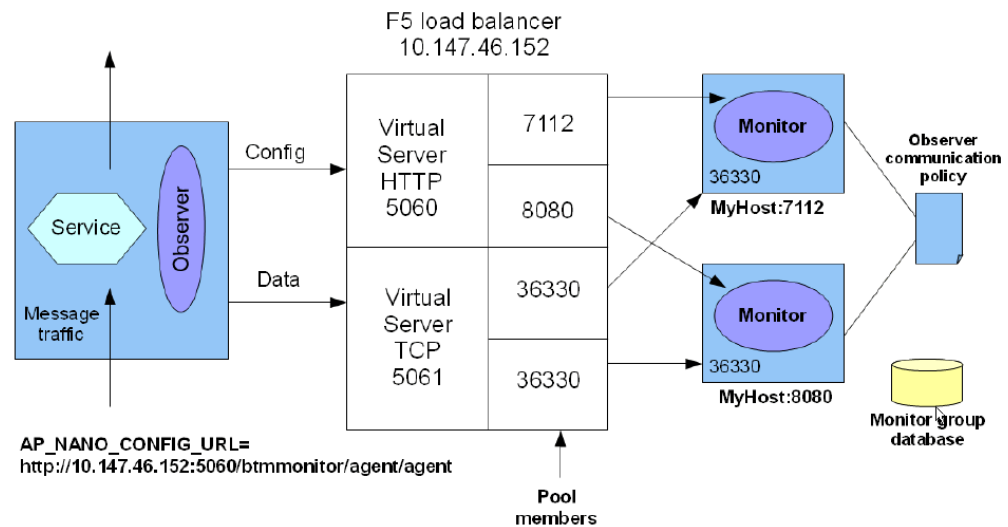
Creating a scheme like the one above involves configuring the F5 device, setting the Java system property or Windows key, and defining the observer communication policy for the replicated monitors.

When you set up the F5 device, you must use the admin console for that device to do the following:

■ Create an HTTP virtual server to be used by the observer to get configuration information. This is shown at port 5060 above.

Assign a pool to the HTTP virtual server with member port numbers that correspond to the HTTP ports of the monitors to which you are connecting. As illustrated, the pool for the HTTP virtual server includes ports 11080 and 11081.

■ Create a socket virtual server to be used by the observer to send data to the socket ports of the monitors. This is at port 5061 above. Assign a pool to the socket server with member port numbers that correspond to the socket (data) ports of the monitors to which you are connecting. As illustrated, the pool for the socket virtual server includes the ports at 36330 on each host machine.

When you set the AP_NANO_CONFIG_URL Java system property or the AmberPoint:NanoConfigURL Windows key, you must provide a value like the following:

http://**10.147.46.152:506**0/btmmonitor/agent/agent/

Note the bold portion of the URL: it is the IP address of the F5 device (host) and the virtual sever HTTP port. (Of course, these numbers will be different for your deployment.)

The values you specify for the observer communication policy correspond to the values defined for the F5 device as follows:

Observer communication policy (Through router to monitor group)	F5 device values
Router IP address	The IP address of the F5 device. With reference to the figure, this would be 10.147.46.152.
Router port number	The virtual server socket number. With reference to the figure, this would be 5061.

Observer communication policy (Through router to monitor group)	F5 device values
Monitor port number	One of the pool member ports. With reference to the figure, this would be 36630.
	If, for some reason, the replicated monitors are located on the same machine, the port numbers for each monitor would be different, and you would need a different observer communication policy for each monitor.

It does not matter whether you define the observer communication policy first and the F5 second. What matters is that the socket ports assigned to the monitors correspond to those defined for the virtual server socket pool in the F5 device.

This section assumes that a certain amount of work has already been done to deploy and register the replicated monitors and to create the monitor group. Consult the *Business Transaction Management Installation Guide* for information on how to do this, and on how to define the observer communication policy.

12.7 Resolving Discovery Issues

Creating a useful discovery configuration can be an iterative process, particularly in the early stages of using Business Transaction Management. You might find that default settings for enabling probes are turning up too much information, or that changing your deployment or the observer-monitor topology results in redundant or erroneous information. To spare you the need to reinstall the system or to manually remove all observed entities and related artifacts, Business Transaction Management provides the deleteAll command. This command deletes objects already discovered along with related artifacts such as transactions, properties, registered services, devices, and containers.

Use this command judiciously to avoid unwanted loss of data, which includes historical data related to observed objects. The command is most appropriate when you start working with Business Transaction Management and are fine tuning your discovery scheme. It should never be used in a production environment. See section 10.11, "deleteAll" for more information.

After you have fine-tuned your discovery configuration, and worked with Business Transaction Management for a while, problems might still arise. The simplest reasons for not being able to observe a service are the following:

- Traffic has not run long enough or diversely enough for Business Transaction Management to see a service. The solution here is to run more traffic and attempt to traverse all possible branches.

- The probe responsible for observing the service has not been activated. To check, select **Administration > Observers** from the Navigator and make sure that the probe appropriate for the type of service you are trying to discover is active. If it is not, edit the observer communication policy to activate it. By default all probes are activated except for the JAVA probe. To enable it, you must configure it with the names of the specific Java classes you want to observe.

More complicated issues arise in determining whether services are replicates. In the process of discovering and representing services and endpoints appropriately, Business Transaction Management needs to figure out whether a copy of a service represents a valid replicate and, conversely, whether services whose WSDL definitions are not identical actually implement the same interface. It makes these decisions by

comparing the WSDLs it discovers and by following the criteria defined by the system service versioning policy. In addition, there might be cases where you might want to separate or merge different versions of a service because of ownership or accounting issues. Business Transaction Management provides commands and tools that you can use to resolve replication and duplication problems, and to resolve cases where it cannot guess your needs or intent.

This section summarizes some of these issues and introduces the commands you use to deal with them. You can also resolve some discovery problems by using the Disambiguate endpoints tool, which you access from the management console. This section includes the following topics:

- Modifying the Service Versioning Policy

- Resolving Replication Issues

- Resolving Duplication Issues

- Using the Console to Disambiguate Endpoints

12.7.1 Modifying the Service Versioning Policy

By default, the service versioning policy sets guidelines for how Business Transaction Management should deal with new or changed WSDLs:

- It treats two endpoints as part of the same service version if their qualified service names, port type names, and port definitions match.

- When it discovers a WSDL defining a distinct version of an existing service, it creates a new service version based on the host and port of the endpoint location being registered.

- When it re-reads a WSDL, if the qualified name of a service changes, it replaces the previous endpoints, consequently losing all measurements.

It might be possible to forestall discovery problems by editing the default service versioning policy to make these criteria more or less restrictive. If modifying the policy does not suffice, there are a number of CLI commands as well as a tool you can access from the console that you can use to correct discovery results. For additional

To edit the default service versioning policy

1. In the Navigator, select **Administration > System Policies.**

2. Double click the **Service Versioning Policy** item in the summary area to display the current versioning policy in the detail area.

3. Select **Edit Definition for Service Versioning Policy** from the **Modify** menu.

4. Make the desired changes and click **Apply**.

12.7.2 Resolving Replication Issues

If you deploy a service in multiple containers, Business Transaction Management is able to understand that the same service is referenced by all the endpoints and that all endpoints share one interface. Consequently, it is able to aggregate statistics for the replicates at the service level and it allows you to define message properties on operations shared by all endpoints in the service.

For two endpoints to be treated as replicates of the same service, the following is required:

- The WSDLs for the two services have the same qualified name (target name space and a simple name)

- The endpoints implement the same interface

- The service type of the endpoints is the same

To take an example, a user web service, an OSB business service, and the OSB Proxy service that have the same name would show up as three separate services because their service types are different.

There are cases however where this information is not sufficient to make a determination, and you might have to teach the system whether two endpoints are the same or different. In each of the following cases, you will be alerted to take some action:

1. In a development environment, multiple versions of the same service might be created that are incompatible. If Business Transaction Management sees a version that looks different from what it has seen before, it will treat it as a different version. If these differences are not important, you can run the `mergeServices` command to merge two versions into one.

2. Due to a rolling upgrade, parallel versions might need to coexist temporarily until all servers are upgraded. In this case, Business Transaction Management will not generate a new service version, but it will generate an alert and allow you to move upgraded endpoints to a new service version if needed. This might be the case if existing policies are not compatible with the upgraded version. You can use the `moveEndpoints` command to move upgraded endpoints to a new service version. By the time the upgrade is done, all endpoints will wind up in the new service version.

3. Due to a failed or incomplete upgrade, a version skew arises that was not intended but results in two different versions of the services. You can choose to merge the two versions or separate them.

4. When a side-by-side upgrade results in the deployment of a new version of the application with an updated interface at new endpoint locations, Business Transaction Management will generate a new service version by default. You can accept this or merge the new version with the older version to retain the service history.

5. Two sets of WSDLs that identify the same service would normally be treated as replicates, but because different instances of the service are used by different departments in different ways, you might need to divide the endpoints between service versions manually (`moveEndpoints`). In this case Business Transaction Management will generate an alert if new replicated endpoints are discovered, to allow you to determine which service version they belong to.

6. There are cases where deploying an updated version of a service causes Business Transaction Management to delete the older version. If you wish to keep the measurements associated with the older version, you can use the `moveMeasurements` command to move the measurements from the older to the newer version.

12.7.3 Resolving Duplication Issues

Business Transaction Management attempts to resolve issues that arise as the result of changes to a machine name or to a container's listening address, or the use of multiple aliases for the same host name, without assistance from you. If the system guesses incorrectly how to handle such conditions, the most common symptom is the

discovery of duplicate services or endpoints where in reality only one exists. You can use the commands listed in the following table to help the system avoid or resolve duplication problems.

The parts to be used as WSDL, service, or endpoint identifiers are as follows (with respect to the following example):

```
http://jbujes-pc.edge.com:8080/Bookmart/Credit/CreditService?wsdl
```

The *base address* is `http://jbujes-pc.edge.com`

The *node* is `jbujes-pc.edge.com`

The *path* is `Bookmart/Credit/CreditService?wsdl`

Generally, using the `removeDuplicateEndpoint` (or `addBaseAddressAlias`) command accomplishes everything you need to do. That is, the duplicate item is removed and the appropriate alias is defined so that duplication does not recur. But note that otherwise, all alias corrections made using the commands listed below are forward looking: they do not delete duplicates that have already been mistakenly created.

Command	Balancing command	Description
addBaseAddressAlias	removeBaseAddressAlias	Teaches the system to recognize that two base addresses are the same. This prevents duplicate endpoints from being discovered.
addNodeAlias	removeNodeAlias	Teaches the system to recognize that two node addresses are the same.
addPathAlias	removePathAlias	Teaches the system to recognize that two paths are the same for a given endpoint or WSDL address.
listNodeAliases	None	Lists current node aliases.
removeDuplicateEndpoint	None	The system has discovered two endpoints with the same path but different host names. Executing this command removes the duplicate and also adds the missing alias to make sure that future discoveries do not result in recreating the duplicate endpoint.
removeNode	None	Removes all information about the node and its aliases. Removing a node is not something you would normally need to do unless aliases for different machines have gotten so mixed up that you face the choice of pruning nodes or reinstalling the system.

Another apparent duplication might result when two endpoints share the same URL and are given the same port name in the WSDL that describes them. By default the port name is used as the friendly name. Although the system does not require friendly names to be unique, you will have to specify the endpoint URL (and possibly other characteristics) instead of its friendly name in any command that requires you to reference a unique endpoint. You can use the `renameEndpoint` command (or just pick the desired endpoint and modify its friendly name in its **Profile** tab) to distinguish the endpoints from one another.

12.7.4 Using the Console to Disambiguate Endpoints

You can use the **Disambiguate Endpoints** tool (from the console) to do the following:

- **Merge services**: merge the source service into the specified target service
- **Move an endpoint:** move the source endpoint to an existing or to a new service

- **Remove a duplicate endpoint**: remove the source endpoint that is a duplicate of a target endpoint

To use the **Disambiguate Endpoints** tool:

1. Select the source service or endpoint and choose **Admin > Disambiguate Endpoints**. The **Disambiguate Endpoints** tool is displayed in the console. The tool consists of three main areas: an area that compares the source and target, an area that lists available actions, and an area that allows you to preview the effect of your actions before you choose to execute them.

   The source drop list includes all possible sources, based on an internal evaluation of duplication in the system. The service or endpoint shown at the top is the one you selected when you opened the tool. You can choose another if you like.

   The target drop list includes all possible targets, given the items you have chosen for source.

   Below the drop lists is shown basic information for the selected source and target. Icons (equal/does not equal) indicate whether elements of the source and target are the same. To view differences in the WSDL's listed, click on the link to display WSDL contents.

2. Use the drop lists to select the source and target.

   If you want to create a new service as the target service, click the **New Service** check box and specify the name and version of the service. You might do this if you want to move the source endpoint to a service that does not yet exist.

3. The action part of the **Disambiguate Endpoints** tool shows you the possible actions you can take based on the selected source and target.

   Click the enabled action. If the desired action is not enabled, you might have to change your target or select **Create New Service**. For detailed information about the effect of each of these actions, please look up its command line equivalent: `mergeServices`, `removeDuplicateEndpoint`, or `moveEndpoints`. The use cases that require these actions are described above

4. After you select an action, the effects of your action are shown in the **Preview** area.

5. If the results shown are what you intended, click the OK button to execute the chosen action.

12.8 Miscellaneous Administration Topics

This section provides miscellaneous information to help you administer Business Transaction Management and includes the following subsections:

- Checking the Status of System Services
- Resynchronizing the System and Checking Aliveness
- Start and Stop Monitoring of Endpoints
- Working with Containers
- Working with System Policies

12.8.1 Unregistering a Service

Use the following procedure in the Management Console to unregister a service. You can also unregister a service using the **unregister** CLI command. For information on how to use this CLI command, refer to Chapter 10, "Commands and Scripts."

To unregister a service

1. Select the service.

2. Choose **Modify > Delete** *My_Service* **Registration**, where *My_Service* is the name of your service.

   The **Delete Service Registration** tool opens.

3. Click **Delete**.

12.8.2 Checking the Status of System Services

The following list describes the ways you can check the status of your system services. The list orders these tasks from high-level to low-level (top to bottom). Each listed task directs you to where you can perform the monitoring and provides a link to more detailed information:

To check:

- Summary of system services and system alerts.

  Navigate to **Dashboards** > **Operational Health Summary**

- Detailed view of all system alerts

  Navigate to **Alerts in the Last Hour** > **System Alerts**

- Current status of a specific system service

  Navigate to the specific system service and display the **Status** tab.

12.8.2.1 System Alerts

System alerts provide information about the health of the Business Transaction Management infrastructure. Business Transaction Management issues system alerts in situations such as the occurrence a fatal error, a container going down, the rejection of a policy, deletion of an attribute that is in use, and so forth.

12.8.2.2 Status Tab

Use the **Status** tab to check the status of the selected Business Transaction Management system service.

Service Status Table

The table at the top of the **Status** tab provides the following information about the service:

- **Url**: The location where Business Transaction Management expects to find the service uuid.

- **Id**: The uuid assigned to the service.

- **Status**: The service status can have one of the following values:

  - RUNNING - The system service is running and working correctly. For a system service to be RUNNING, all of its associated resources must be accessible and running.

  - DEGRADED - A resource may be unreachable, but that resource is not mandatory for the system service to operate. It is a good idea to investigate unreachable resources for a degraded service. A re-sync of the system may fix services that are in a degraded state. Degraded services do not trigger alerts to the system log.

- FAULTED - A required resource cannot be reached. Typically, a faulted value appears when a database required by the service is down. When a service reaches the faulted state, it is considered unusable. Access is subsequently shut down to the service to prevent errors from cascading throughout the system. A faulted service triggers an alert to the system log. This alert should provide information about how to address the root cause of the problem. A re-sync of the system will not fix services that are in a faulted state. Instead, you need to first address the root cause of the problem, and then restart the Business Transaction Management deployment that contains the faulted service.

- SETUP_REQUIRED - A required resource or piece of the system service has not been upgraded. For example, the transaction management component is not connected to a database.

- INITIALIZING - A service is in this transient state at the very beginning of its lifetime.

Resources Status Table

Below the Service Status table is the Resources Status table, which provides status information about other components in the Business Transaction Management environment that are related to the selected service. For example, several system services use databases. The resources section will confirm whether the database is running and working correctly.

The Resources Status table provides the following types of information:

- **Status**: The resource status can have one of the following values:

  - OK - The resource is running and working correctly.

  - FAULTED - A required resource cannot be reached.

  - SETUP_REQUIRED - A required resource has not been upgraded.

- **Last Access**: Time-stamp of when the service last attempted to access the resource.

- **Last Successful Access**: Time-stamp of when the service last accessed the resource. When diagnosing problems using the system log, the time-stamp of the Last Successful Access for a resource is a good starting point.

Another important piece of information included in the Resources Status table is the URL of the sphere service with which the selected system service is associated.

> **Note:** Resource URLs are displayed either with a dash "-", or with parentheses "()". URLs displayed with a dash represent a resource that can exist only once within an installation, such as the sphere service, whereas URLs displayed with parentheses represent a resource of which there can exist multiple instances, such as a monitor.

12.8.3 Resynchronizing the System and Checking Aliveness

The sphere is synchronized about every fifteen minutes. You can manually resynchronize the sphere by choosing **Admin > Resync System**. Resynchronizing ensures that the sphere reflects the current state of your system. **Resync System** also performs an aliveness check at the same time.

By default, "aliveness" is checked every two minutes to determine whether services are up or down. You can change the time interval for the automatic aliveness check, as

described below. (The CLI equivalent for checking aliveness is the
configureAlivenessCheck command.

12.8.3.1 Changing the Time Interval for the Automatic "Aliveness" Check

To change the time interval for the automatic aliveness check:

1. In the Navigator, choose **Administration > System Services**.

2. Select **AP_Sphere_Service** and then choose **Admin >Edit Setup Data for AP_ Sphere_Service**.

3. Click the **Edit XML** link.

4. Find the <AlivenessInterval> element and change its value to the appropriate number of seconds. The default value is 120, or two minutes.

5. Click **Apply**.

12.8.4 Start and Stop Monitoring of Endpoints

Business Transaction Management automatically discovers and monitors components, depending on how you have set up the Observer Communication policy. Once a component has been discovered, you can explicitly start and stop the monitoring of its endpoints.

When you start or stop monitoring an endpoint, both the monitor and the observer configurations are updated. Once monitoring stops, no performance measurements are recorded, no messages are logged, and no transactions are traced.

To stop monitoring of an endpoint

1. Select the endpoint (or the service).

2. Choose **Admin > Stop Monitoring**.

    The **Stop Monitoring** tool then opens and lets you select additional endpoints to stop monitoring.

To start monitoring of an endpoint

1. Select the endpoint (or the service).

2. Choose **Admin > Start Monitoring**.

    The **Start Monitoring** tool then opens and lets you select additional endpoints to start monitoring.

12.8.5 Working with Containers

> **Note:** Business Transaction Management can use dependency analysis to discover service endpoints running in unmonitored containers if those endpoints interact with services in monitored containers. You can use the registerExternalContainer command to define containers for such endpoints.

You can explore the registered containers by selecting **Explorer > Containers** in the Navigator. The summary area then lists all registered containers.

Select a container in the main area to display information about the container on the **Profile** tab. Open the container's administration console by clicking the URL in the

Administration UI Console field. You can edit the container's profile by choosing **Modify > Edit Profile for** *the_selected_container*.

To view the services running in a container, select the container and display the **Services** tab. You can also drill down into the service endpoints running in a container by expanding the container in the main area.

12.8.5.1 Monitoring Containers

The following list describes the ways you can monitor the health of your containers. The list orders these monitoring tasks from high-level to low-level (top to bottom). Each listed task directs you to where you can perform the monitoring and provides a link to more detailed information:

To check:

- Overall current health of all containers.

  Navigate to **Dashboards** > **Operational Health Summary**

- Recent history of all container-related problems (in addition to other system-related problems)

  Navigate to **Alerts in the Last Hour** > **System Alerts**

- Current health of a specific container

  Navigate to *specific container* > the **Up/Down Status** icon in the summary area

12.8.5.2 Unregistering Containers

There might be times when you want to unregister a container (in other words, remove it from the sphere and the Management Console). For example, if any of the following statements are true, you can, and probably, should unregister the container:

- You have registered the container with a different sphere

  This situation results if you reconfigure an observer to send observations to a monitor that is registered in a different sphere. In this case, the container in which the observer is running is automatically registered with the other sphere as soon as traffic is observed.

- You have uninstalled the observer from the container.

- You have uninstalled or physically removed the container from your system.

If none of these statements are true, and you unregister a container, the container will be automatically reregistered as soon as message traffic is observed to or from the container.

To unregister a container

1. Select the container you want to unregister.

2. Choose **Modify > Delete** *selected_container* **Registration**.

   The **Delete Registration** tool opens.

3. Click **Delete**.

12.8.6 Working with System Policies

System policies are normally used by administrators. Most system policies are not editable. The ones that you can modify allow you do define custom attributes, to configure the connection to the message logging database, to configure discovery, to

specify the interval at which measurements are aggregated and collected, and to tell the system what action to take in response to new or changed WSDLs.

This section describes available system policies and explains how you access them, how you view their definition, how you define them, and how you modify them.

12.8.6.1 Accessing System Policies

You can view system policies by selecting **Administration > System Policies** in the navigator. The following table lists and describes these policies.

Name	Description
Baseline storage	Supports the derivation and storage of baseline values for endpoints, services, and transactions.
Callout Measurement	Supports the measurement and display of average response time, throughput, and fault count for a given link. You cannot edit this policy.
Condition measurement	Provides condition alert count and condition alert count rate measurements for a given transaction. You cannot edit this policy.
Core Measurement	Supports the calculation of the following counts for a given endpoint and service: average response time, maximum response time, throughput, fault count, traffic, fault percentage, throughput rate, and fault rate. You cannot edit this policy.
Data Model Attribute Definitions	Extends custom attributes for services, endpoints, operations, containers, agents, business objects, transactions, and type domains.
Default Message Logging Database	Configures the connection to a database used for logging messages. For more information, see Section 12.2.2, "Setting up the Message Log Database."
Default Observer Communication Policy	Determines which genres are to be observed and monitored, and specifies the monitor port number.
Event Generation Policy	Configures the generation of events based on the occurrence of event notifications.
Event Notification Measurement Policy	Used by the notifier service. You can only re-define criteria for this policy.
Measurement Interval Policy	Specifies the interval at which measurements are aggregated and collected throughout Business Transaction Management.
Service Versioning Policy	Directs the system in what action to take in response to new or changed WSDLs. For more information, see Section 12.7, "Resolving Discovery Issues."
Simple Transaction Measurement Policy	Supports measurement of average response time, maximum response time, completed transaction count, started transaction count, completed transaction rate, started transaction rate for a given transaction. Business Transaction Management will use either this policy or the Transaction Measurement policy to measure transaction performance, depending on how the transaction is defined and what features are used. You cannot edit this policy.
System generated log policy for *transaction*	Specifies the location for storing transaction messages, the limits on time store, the rotation interval, and defines the scope of logging. You cannot edit this policy.

Name	Description
Transaction measurement	Supports measurement of average response time, maximum response time, completed transaction count, started transaction count, completed transaction rate, started transaction rate for a given transaction.
	Business Transaction Management will use either this policy or the Simple Transaction Measurement policy to measure transaction performance, depending on how the transaction is defined and what features are used.
	You cannot edit this policy.
Uptime Measurement	Supports the uptime measurement for a given endpoint and service. You cannot edit this policy.
Monitor Group Policy	Creates and configures a monitor agent group.

12.8.6.2 Viewing a System Policy Definition

To view a system policy definition:

1. Select **Administration > System Policies**.

2. Double click the policy of interest in the main pane.

3. Click the **Profile** tab.

12.8.6.3 Defining and Modifying System Policies

To define a system policy:

1. Select **Create System Policy** from the **Admin** menu.

2. Choose one of the following: **Message Log Database**, **Monitor Agent Group**, or **Observer Communication**.

3. Specify the information required and click **Apply**.

When you define a policy, you must supply identifying information: the policy's name and whether it is enabled. You must also supply settings that determine what the policy will accomplish, and criteria that determine to which endpoints (targets) the policy is applied.

Some system policies can only be modified and must be accessed from **Administration > System Policies** in the navigator.

To modify a system policy

1. Select **Administration > System Policies**.

2. Select the policy of interest in the main area.

3. Select **Modify > Edit definition for** *policy*. If the menu item is not available, the policy cannot be modified.

Index