BEA TSAM™

Administration Guide

Version 1.1
Document Released: September 28, 2007

Contents

Introduction to BEA TSAM

This topic contains the following sections:

Overview

BEA TSAM (Tuxedo System and Application Monitor), is a BEA Tuxedo add-on product. Tuxedo is widely used by enterprises that develop and use in mission-critical applications. It acts as the infrastructure layer in distributed computing environments. The complexity of Tuxedo and the applications running on top of it makes performance measurement extremely complex.

BEA TSAM monitors the major performance sensitive areas of a Tuxedo-supported enterprise computing environment. It can be used to monitor *real-time* performance bottlenecks and business data fluctuations, determine service models, and provide notification when pre-defined thresholds are violated.

BEA TSAM Features

The following is a list of BEA TSAM features:

- Tracking a Tuxedo system call transmissions. Each monitored call is assigned a unique ID and is propagated along a call path tree. TSAM is able to track calls across multiple machines and domains.
- *Real-time* call path tree tracking of a monitored request is displayed and the performance metrics for each step are available.
- Call pattern summarization based on historical call tracking data.
- Monitoring a particular Tuxedo service, checking its response time, IPC queue length and execution status. The data can be queried using recent or historical data.
- Gathers Tuxedo GWTDOMAIN and BRIDGE overall throughput, graphically displaying the business data flow curve.
- Tracking transactions with XA API specifications. Displays execution status and time used on each XA call. TSAM helps diagnose global distributed transactions.
- The BEA TSAM Manager console provides the capability to create "Alert" definitions that generate events when predefined thresholds are reached. The events can be posted to Tuxedo and received by Tuxedo Event Broker subscribers.
- Programming APIs that retrieve meta data packaged in a monitored call. Helps developers make application decisions dynamically.
- Flexible monitoring controls. The sampling can be based on interval or ratio and the monitoring can be turned on or off dynamically without restarting application.
- Plug-in mechanism for performance metrics collection at the Tuxedo infrastructure level. It provides great integration capability between TSAM and other third-party products.
- Powerful plug-in-level event triggers without sending raw metrics data to the BEA TSAM Manager. It supports flexible FML boolean expression to achieve advanced event trigger conditions. Events can be posted to the Tuxedo Event Broker and/or the BEA TSAM Manager.
- Scalable Tuxedo- side server monitoring design to meet small, middle and large Tuxedo runtime environments.
- J2EE based solution. Easy to deploy, configure and use. It is a pure Web-based solution. The TSAM Console can be accessed anywhere using a compatible Web browser.

BEA TSAM Components

BEA TSAM includes two components:

BEA TSAM Agent: Performs Tuxedo-side data collection.

BEA TSAM Manager: Performs data storage, aggregation, computing and representation.

BEA TSAM Agent

The BEA TSAM Agent handles all Tuxedo-side back-end logic. It works in conjunction with the BEA TSAM Manager, and includes the following sub-components:

- **BEA TSAM Framework**: A Tuxedo-side facility that defines and controls performance metrics collection behavior. It uses the Tuxedo traditional interface and can be easily integrated into an existing Tuxedo management suite.
- **BEA TSAM Plug-in**: An extensible mechanism invoked by the BEA TSAM Framework**.** The BEA TSAM Agent provides default plug-ins to send data to the LMS (Local Monitor Server), and then to the BEA TSAM Manager. The default plug-in also checks event triggers, and generates events if needed.

  You can develop your own plug-ins for additional data processing. A customized plug-in can be linked to an existing plug-in chain, or replace the default plug-in.
- **LMS (Local Monitor Server)**: The LMS is a Tuxedo system server. The BEA TSAM default plug-in sends data to the LMS. The LMS then passes the data to the BEA TSAM Manager in HTTP/XML message format.

For more information, see BEA TSAM Agent.

BEA TSAM Manager

The BEA TSAM Manager is built on standard J2EE technology. It includes following components:

- **BEA TSAM Data Server:** Data server that accepts data from the LMS and stores the data in the database. It is a standard J2EE application.
- **BEA TSAM Console**: The BEA TSAM presentation layer. It is a standard J2EE Web application and can be accessed via a compatible Web browser. After logging on to the BEA TSAM Console, you have access to full BEA TSAM functionality.

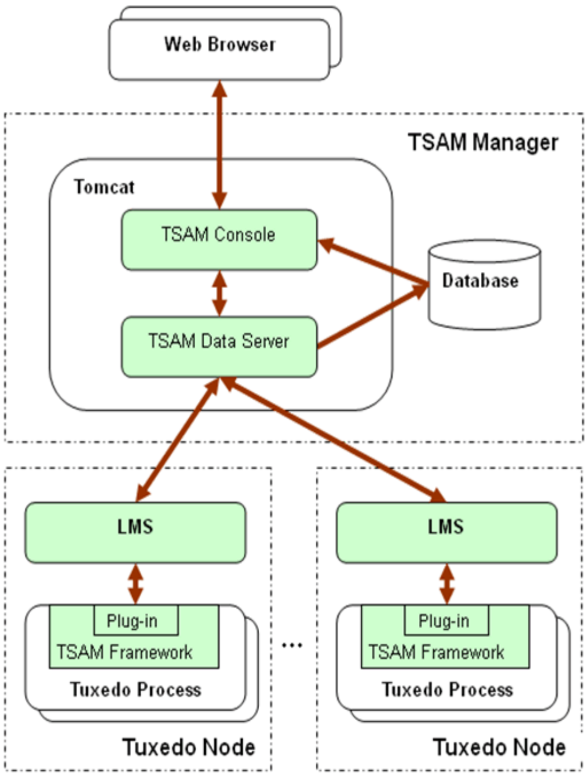
- **Apache Derby Database**: The BEA TSAM Manager ships with an Apache Derby evaluation copy as the default database solution. Oracle is also supported using a pre-packaged SQL script (Oracle databases are not included with the BEA TSAM Manager).

  Note: The BEA TSAM Manager ships with Apache Tomcat as the J2EE application server used by the BEA TSAM Data Server and BEA TSAM Console.

For more information, see BEA TSAM Manager.

TSAM Architecture Diagram

Figure 1-1 TSAM Architecture



BEA TSAM Concepts

Call Path Monitoring

Tuxedo is typically used by a client program (not necessarily a Tuxedo client process) that calls a service to perform a business computing logic scenario. The service implementation is completely transparent to the caller. This type of middleware transparency provides many benefits for development, deployment, and system administration. However, from a monitoring perspective, it is difficult for the end user or administrator to figure out what happens "behind the scene". BEA TSAM call path monitoring helps to alleviate this problem.

Call Path Tree Definition

A simple Tuxedo application call triggers a set of service invocations. The involved services constitute a tree (call path tree"). A call path tree strictly defines the following factors:

- What type of services are involved to perform the initial service request.
- The service invocation depth (that is, the depth of the call path tree).
- The service invocation sequence. For example, client A calls SVC1. SVC1 calls SVC2 and SVC3.
- Call transportation. The *edge* (how information is sent and received) of a call path tree represents the transportation information from caller to service provider. It could be an IPC queue, BRIDGE connection or DOMAIN connection. The elapse time used for each transportation is also recorded.

Monitoring Initiator

A "**monitoring initiator**" is a process that "initiates" tracking a call path tree. The process can be a Tuxedo client, application server, or client proxy server (WSH/JSH). A typical scenario is when a `tpcall`/`tpacall` is invoked by the monitoring initiator; call path monitoring begins. All the back-end services involved in this call are displayed on the call path tree representation in the BEA TSAM Console.

Note: Currently only `tpcall`/`tpacall` can trigger a call path monitoring. Other communication models are not supported.

A Tuxedo application server performs two functions:

- All sub-calls made in the service implementation are a part of the call path tree started by the original monitoring initiator (if the incoming request is already monitored).

- It is a monitoring initiator with calls made in the service routine according to the monitoring policy definition.

Service Monitoring

Service monitoring focuses on pure Tuxedo service execution status. It does not care about call correlation, as call path monitoring does. Service monitoring can be used with call path monitoring together or performed independently.

System Server Monitoring

Tuxedo has two important system servers: BRIDGE and GWTDOMAIN. BRIDGE connects multiple Tuxedo machines in a Tuxedo domain. GWTDOMAIN connects one Tuxedo domain with another. The system server monitoring tracks message throughput, pending sent messages, and awaiting reply messages on each network link.

Transaction Monitoring

A critical use of Tuxedo is transaction monitoring. Tuxedo coordinates activities in a distributed transaction with an XA compliant resource manager, such as a database. BEA TSAM transaction monitoring tracks each XA call triggered in a transaction allowing you to clearly identify where a global distributed transaction is bottle necked.

Monitoring Policy

Monitoring policy controls monitoring behavior.

- **On or Off**. BEA TSAM monitoring can be dynamically turned on or off (for a specific type or several kinds of monitoring) for a particular Tuxedo component. The Tuxedo component can be a particular server, group, or machine.
- **Interval-Based Monitoring.** Monitoring is initiated based on specific time intervals. For example, call path monitoring. An interval-based monitoring policy can specify that the call path is tracked in 60-second intervals.
- **Ratio-Based Monitoring**. Monitoring is initiated by the number of executions. For example, service monitoring. A ratio that is set to 5 indicates that every 5 executed services are monitored. For call path monitoring, a ratio set to 5 indicates that every 5 `tpcall`/`tpacall` calls are monitored.

- **Flexibility to Reduce Monitoring Performance Impact**. BEA TSAM monitoring control enables you to configure the monitoring policy based on your application size, load and network activity.

Performance Metrics

BEA TSAM performance metrics are listed as follows:

- **Correlation ID:** A unique identifier that represents a call path tree. It is generated by the monitoring initiator plug-in. It uses the following format:

```
DOMAINID:MASTERHOSTNAME:IPCKEY LMID PROCESSNAME PID TID COUNTER
```

Listing 1 shows an example of a Correlation ID. The monitored call is started by the program “`bankclient`” with process **ID 8089** and thread `ID 1` on machine “`SITE1`” on Tuxedo domain “`TUXDOM1`”. The master is “`bjsol18`” and `IPCKEY` in TUXCONFIG is “`72854`”.

Listing 1 Correlation ID Example

```
TUXDOM1:bjsol18:72854 SITE1 bankclient 8089 1 99
```

- **Service Name:** The name of a Tuxedo Service.
- **Location:** The string to identify the process who sends out the performance metrics.
- **IPC Queue Length:** The message number in an IPC queue.
- **IPC Queue ID:** Tuxedo identifier of an IPC queue.
- **Execution Time:** The time used in a Tuxedo service or XA call execution, it is in millisecond level.
- **Wait Time:** The time used of a message in the transportation stage.
- **Message Size:** The Tuxedo message size.
- **Execution Status:** The `tpreturn` service return code. It is defined by Tuxedo ATMI interface.
- **Call Flags:** The flags passed to `tpcall/tpacall` in Tuxedo ATMI interface.
- **Call Type:** `tpcall`, `tpacall`, or `tpforward`.

- **Elapse Time:** The time elapsed sine the a call is monitored.
- **GTRID:** Tuxedo global transaction ID.
- **Pending Message Number:** The number of messages which are delivered to Tuxedo network layer and waiting for being sent.
- **Message Throughput:** The total message number and volume accumulated in system server monitoring intervals.
- **Waiting Reply Message Number:** The number of requests in GWTDOMAIN are waiting reply from remote domain.
- **XA Code:** The XA call return code in transaction monitoring.
- **XA Name:** The XA call name.

BEA TSAM Use Cases

BEA TSAM is built on top of Tuxedo and has unique service, call, and transaction tracking capabilities. Enterprise organization usually have many widely distributed services deployed and one client request that requires complex back-end service coordination to perform the processes.

It can be difficult for an administrator to figure out what exactly is happening during these interactions. BEA TSAM call path monitoring helps to alleviate this problem.

The followings are FAQs will help you to better understand how BEA TSAM works with your applications:

- Understanding Your Applications
- Solving Application Performance Problems
- Improving Application Performance

Understanding Your Applications

What happens behind a simple call?

Enabling call path monitoring for a Tuxedo client or application server allows you to find out all the information behind a simple `tpcall`/`tpacall`. The tracking points span multiple machines and multiple domains. You can clearly see the following information in the call path tree:

- The service invocation hierarchy that supports your call

- The transmission cost for each message flow step, from IPC Queue to Network
- The execution status of each service involved
- The call type and call flags of all the intermediate calls
- The waiting time in queue and response time for each service
- The end-to-end response time

What about my services?

Service monitoring enables you to measure your service response time, IPC queue length, and execution status. Service monitoring provides the following information:

- Service Execution Status Summary.

  BEA TSAM tells you how many service executions succeeded or failed recently or during a period of time. BEA TSAM also computes the average response time. These are important factors in measuring the quality of your services.

- Service Activity Trends.

  BEA TSAM also displays your services activity trends. It tells you what the peek time is and when the services requests are low.

Is my network busy?

BEA TSAM allows you to monitor the network connection attached to your local domain gateways. You can easily find which link is busy and its data fluctuation trend. You have more in-depth understanding of the business data flow model between departments and organizations.

Who participates in my transaction?

BEA TSAM monitors the transaction XA calls. Transaction participants are listed on the transaction monitoring page. For a large distributed transaction, a slow branch can result in the entire transaction being slowly completed. BEA TSAM lets you know who the transaction participants are, and how much time is used during XA calls.

Solving Application Performance Problems

Why is the service response time is slow recently?

Turn on the call path monitoring for a particular call to investigate the following:

- How much network-side time is used
- Which services are the most time-consuming point in the call path tree
- Is the service routed to a remote machine or a domain
- Is it client wait time a reply problem?

My back-end services failed, but I don't know which one.

Turn on call path monitoring. You can find the service execution status for this call.

How many kinds of call paths are in my application?

Turn on call path monitoring using an adequate sampling policy. BEA TSAM will tell you how many call paths (a "call pattern") exist in your application.

Why is my global distributed transaction completed slowly?

Turn on BEA TSAM transaction monitoring. You can see the execution time used by the transaction participants.

I want to correlate local transactions with remote transactions.

Turn on BEA TSAM transaction monitoring for all involved processes and GWTDOMAIN. The BEA TSAM Console shows you the transaction mapping between local and remote transactions.

I want to know what is the peak time that my local domain uses resources from the remote domain, and how busy it is.

Use BEA TSAM system server monitoring on the GWTDOMAIN. BEA TSAM records the information for you, and shows you the throughput trends.

Can I check program request information?

Turn on call path monitoring with the proper monitoring policy and then use "`tpgetcallinfo`". The following information is provided.

- The timestamp when the request leaves the caller
- The timestamp when the request comes into to the server IPC queue
- The client IP address (workstation client, GWWS client)

The monitoring initiator process, `tpgetcallinfo()`, can also tell you the total time used.

Improving Application Performance

Are my services too fine grained?

In some cases, too many services supporting a request may add to performance overhead. Use call path tree to investigate. The service number and the tree depth are key analysis factors.

Are my services deployed properly?

Some services are called more frequently than others. Use call path monitoring to gather the information, and re-consider the service deployment. It is best to have the most used services located on the local machine and LAN. Services across domain services should be used carefully.

Do I have too many servers configured?

BEA TSAM provides a central view of your Tuxedo applications with multiple domain support. Using BEA TSAM Console allows you to easily see how many domains, machines, servers and services are configured.

I want to be notified when a service execution fails or the response time exceeds a pre-defined threshold

You have two ways to do this. One is using BEA TSAM plug-in level event trigger; the other is to define alerts using the BEA TSAM Manager console.

- BEA TSAM Plug-in Event Trigger.
- It supports the FML boolean expression for what Tuxedo components you want to check and what metric conditions you want to trigger an event. It does not require performance metrics to be sent to the BEA TSAM Manager because the evaluation is done bythe BEA TSAM default plug-in.
- BEA TSAM Manger Console Alert Management.

  The BEA TSAM Manger Console allows you to define an alert with required conditions. When the threshold is reached, BEA TSAM generates the events. The events can also be posted to Tuxedo Event Brokers.

Quick Start

To add BEA TSAM functionality to an existing Tuxedo application, do the following steps:

1. Install BEA TSAM

   Install the BEA TSAM Agent and the BEA TSAM Manager. For more information, see the BEA TSAM Installation Guide.

2. Deploy BEA TSAM Manager

   For more information, see the BEA TSAM Deployment Guide.

3. Deploy TSAM Agent

   For more information, see the BEA TSAM Deployment Guide

4. Find Your Tuxedo Configuration

   Login to the BEA TSAM Manager Console (for example: **http://localhost:8080/tsam)**.

   Note: The first time you login to the TSAM console you must set the admin password.

   Go to **BEA TSAM** > **Administration** > **Tuxedo Configuration** to find your Tuxedo configuration. For more information, see the BEA TSAM Console User Guide.

5. Configure Monitoring Policy

   Monitoring policy defines what and how you want to monitor. The monitoring policy can be configured at either TSAM Manager side or TSAM Agent side.

   For more information, see Monitoring Policy Management and Configuring Monitoring Policies in the "*BEA TSAM Agent User Guide*."

   For more information, see Policy Management in "*BEA TSAM Console User Guide*."

   - **Typical Monitoring Policy**

     Go to **BEA TSAM > Administration > Policy Management,** click "**Create**" to enter the "**Policy Specification**" page. Input "tsampolicy" in the "**Name:**" input field.

   - **Monitor call path initiated from a particular server**

     a. Click "**New**" button to show the "**Policy Definition**" page.

     b. In the left panel "**Tuxedo Component**", select the "**Domain**", "**Machine**","**Group**" and "**Server**" in the drop down lists.

     c. In the right panel "**Policy Management Definition**" select the "**Enable**" checkbox of "**Call path**"

     d. Click "**Add**" button

     e. Select that policy entry just created in "**Monitoring Policy Set**" table and click "**Enable**" button.

- **Check BEA TSAM Agent results from Tuxedo**

  If you want to monitor a call path from a particular client process, you must use the TSAM Agent TMMONITOR environment variable for that client.

- **Monitor services of a particular server**

  Same steps as call path monitoring policy set, except you must select the **Services** "**Enable**" check box.

- **Monitor a Domain Gateway**

  Same steps as call path monitoring policy set but

  - Select GWTDOMAIN you want in the "**Tuxedo Component**" panel

  - Select **System Servers** "**Enable**" check box.

- **Monitor XA calls in transaction for a particular group**

  Same steps as call path monitoring policy set but

  - Select the GROUP you want in the "**Tuxedo Component**" panel

  - Select the **Transaction** "**Enable**" check box

6. Start to Monitor Tuxedo

   Login to TSAM Console, and start to monitor Tuxedo system and application.

   Go to **TSAM** > **Call Path** to monitor call path.

   Go to **TSAM** > **Service** to monitor service.

   Go to **TSAM** > **System Server** to monitor system server.

   Go to **TSAM** > **Transaction** to monitor transaction.

   For more information, see the BEA TSAM Console User Guide.

CHAPTER 2

BEA TSAM Agent

This topic contains the following sections:

- Prerequisites
- Monitoring Policy Management
- Configuring Monitoring Policies
- tpcallinfo API
- LMS (Local Monitor Server)
- Plug-in Level Events

Prerequisites

To effectively and correctly use the BEA TSAM Agent, please note the following prerequisites:

- The system clocks for all monitored Tuxedo machines and the BEA TSAM Manager are synchronized. A uniform time server is recommended.
- Set each Tuxedo domain to a unique DOMAINID in the UBBCONFIG file.

Monitoring Policy Management

BEA TSAM provides flexible and comprehensive monitoring management functionality. It provides the following features:

- Monitoring can be turned on or off dynamically without rebooting Tuxedo applications
- Monitoring can be applied to Tuxedo components at the server, group or machine level
- Multiple monitoring management interfaces:
  - `tmadmin` command
  - environment variable
  - MIB
  - BEA TSAM Console
- Fine grained monitoring properties based on time intervals and ratios

Concepts

Before monitoring starts, you must specify what Tuxedo system components and applications you want monitored, and how you want to monitor them by configuring respective monitoring policies. A monitoring policy defines the monitored Tuxedo component, monitoring categories, and monitoring properties.

Monitoring Tuxedo Components

The monitored Tuxedo component can be a machine, group and/or server. At the machine level, all Tuxedo application processes are effected. At the group level, all the servers running in a group are effected. For a particular server, only the server instance is effected.

Note: The later setting of a monitoring policy overwrites previous settings. For example, if a server has enabled "service" monitoring, and the last group that the server belonged to is set to "call path" monitoring, the "call path" attribute overwrites the existing "service" monitoring setting.

Monitoring Categories

BEA TSAM Agent has 4 monitoring areas:

- Call Path Monitoring
- Service Monitoring
- System Server Monitoring
- Transaction Monitoring

Call Path Monitoring

When call path monitoring is enabled, the BEA TSAM framework tracks a requesting calls (using `tpcall` or `tpacall`) until a reply is received by the initiating caller. All back-end services track and compose the call path tree nodes. The edge connecting service nodes carry the transport information

Call path monitoring can be initiated from a client, server or WSH/JSH. When a reply is received from the initiating caller, the monitoring activity for the call is complete.

Key Word: “app”.

If “app” is specified for a process, the process becomes the “monitoring initiator”. A call from this process can be tracked throughout the Tuxedo system until a reply is received. The BEA TSAM console tracks the call path tree in real time. Monitoring initiator processes are limited to the following:

- client process (ATMIs)
- WSH/JSH
- application server processes

Once monitoring for a particular call is started by the “initiator,” monitoring attributes are propagated with this call along its call path. All processes handling this call recognize monitoring characteristics and update related metrics.

An application server has two specific and important functions:

- If the incoming request already indicates it is being monitored, then any call made in the service routine is deemed as part of the original call path tree, whether “app” is set for the server or not.
- If the incoming request does not have monitoring attribute, and “app” is set for the current process, then any service implementation call will start a new monitoring process. The application server becxomes the “monitoring initiator”.

Note: BEA TSAM uses the “monitoring initiator” as part of the correlation ID. For a server process, it is the process name. For a client process, you should specify the name the same as specified using `userlog(3c)`. If no name is given, BEA TSAM uses “client” as the fixed name.

Service Monitoring

Unlike call path monitoring (which focuses on message correlation triggered by a particular call), service monitoring focuses on pure service execution. It records service execution status including the request waiting time, service routine execution time, execution status and buffer size, and so on.

Key Word: “svc”.

> Enables service monitoring. It applies to normal Tuxedo application services and services imported by GWTDOMAIN.

System Server Monitoring

The Tuxedo framework has two key servers, BRIDGE and GWTDOMAIN, that connect machines and domains in a distributed computing environment. BEA TSAM also monitors the activity on the these servers. System server monitoring tracks the overall data for each network connection, and does not differentiate what service or call is passed through.

Key Word: “sys”.

> Enables system server monitoring for the current process. It only applies to Tuxedo BRIDGE and GWTDOMAIN servers. When `sys` is set, BEA TSAM periodically reports the data on the connected network link .

Transaction Monitoring

A major Tuxedo function is transaction monitoring (following XA specifications). A global transaction may have multiple resource managers involved. Because the XA interface is implemented by the vendor, it is difficult for Tuxedo users to measure how much time is used by the XA calls. BEA TSAM provides transaction monitoring functionality.

Besides transaction call time tracking, BEA TSAM agent also reports the return code and transaction ID. If a transaction is sent across domains, BEA TSAM also reports mapping between local transactions and remote transactions.

Key Word: “tran”.

> Enables transaction monitoring. Each transaction call is measured and the execution status sent to the BEA TSAM manager.

All the four monitoring types can be used together or individually. The BEA TSAM console organizes the functionality pages based on the defined monitoring type(s).

Monitoring Properties

BEA TSAM Agent has 4 monitoring properties areas:

- Call Path Monitoring Properties

- Service Monitoring Properties
- System Server Monitoring Properties
- Transaction Monitoring Properties

Call Path Monitoring Properties

Call path monitoring properties use the following keywords:

`appratio`

Enables monitoring frequency. Its range is [0-65535]. "0" indicates that monitoring is stopped. If not specified, the default value is 1 which specifies that every request is monitored.

`appratio` only applies to the "monitoring initiator" and controls the request monitoring frequency. For example, if a client process is set to `appratio=3` and issues 10 tpcalls, then request calls 1, 4, 7, 10 are tracked.

`appinterval`

Initiates the time interval (in seconds) for a monitored action. Its range is [0-65535]. If not specified, the default value is "0" which specifies that there is no time limitation.

Similar to `appratio`, `appinterval` only applies to the "monitoring initiator." For example, if a client process is set to "`appinterval=10`" and continually issues calls for 60 seconds, then calls are tracked at the following time intervals: 0, 10, 20, 30, 40, 50, 60.

Note: "`appratio`" and "`appinterval`" are exclusive. If they are not specified, then every call is monitored.

`appnolog`

For monitored call path requests, by default every process that participates on a call path tree invokes the monitoring plug-in. In some instances, you may not want to trigger the plug-in. Here are twoscenarios:

- An application server or domain gateway administrator may want to disable the plug-in (to increase performance) even though the pass-through message indicates it is being monitored.
- The monitoring initiator does not want trigger the plug-in invocation on its call path tree, but the monitoring attributes are preferred in the message so a program can use `tpgetcallinfo()` to retrieve information.

Using "`appnolog`" allows the BEA TSAM framework to track the calls in Tuxedo system, but without initiating the plug-in invocation. "appnolog" default value is 0 which permits the plug-in to be invoked If "appnolog" is set to 1, the plug-in is not invoked.

Here are two `appnolog` usage examples:

- If “appnolog” is specified by the monitoring initiator, the plug-in invocation is not allowed on its call path tree
- If “appnolog” is specified by a “pass-through” process, the plug-in invocation is not allowed in this process.

Plug-in invocation is only allowed when “`appnolog`” is *not* set in both examples.

Note: If “`appnolog`” is used in part of process on the call path tree, the call path tree on BEA TSAM Web console is not complete.

`appdecode`

Used only for Tuxedo BRIDGE call path monitoring. By default, messages passing through BRIDGE are encoded. This means the BEA TSAM framework cannot extract the monitoring attributes from the message. If BRIDGE is monitored, `appdecode` must be set for BRIDGE processes. The default value is 0, which means BRIDGE will not decode the message and will not appear in the call path tree. If “`appdecode`” is set to 1, BRIDGE decodes the “**monitored message**” and composes the transportation information on the call path tree.

Note: If “BRIDGE” is set with “`appdecode=1`”, it will not decode *non-monitored* messages.

Service Monitoring Properties

`svcratio`

Controls service execution monitoring frequency. Similar to `appratio`.

`svcinterval`

Controls service time interval monitoring. Similar to `appinterval`.

Note: `svcratio` and `svcinterval` are exclusive. If both of are not specified, all services are monitored.

System Server Monitoring Properties

`sysinterval`

Controls the plug-in invocation interval of the monitored BRIDGE or GWTDOMAIN. Its range (in seconds) is [30-65535]. The default value is 300.

Transaction Monitoring Properties

`tranratio`

Controls the transaction call monitoring process-level frequency. Its range is [0-65535]. The default value is 1. A 0 value stops transaction monitoring.

Note: It is recommended that you use the default value since other values may cause a loss of transaction monitoring data.

Configuring Monitoring Policies

BEA TSAM provides four policy monitoring configuration interfaces:

- TMMONITOR Environment Variable
- changemonitor Command
- MIB Interface
- BEA TSAM Console

Policy monitoring information must use the following format:

`monitoring category:monitoring properties:required fields`

- `monitoring category`**:** defines the monitoring category
- `monitoring properties`**:** defines the monitoring properties
- `required fields:` reserved for BEA TSAM plug-in development. It should not be configured for the BEA TSAM default plug-in.

Table 2-1 lists the corresponding keywords

Table 2-1 BEA TSAM Monitoring String Specification Keywords

Monitoring Category Keywords	Monitoring Property Keywords
`app`	`appratio, appinterval, appnolog, appdecode`
`svc`	`svcratio, svcinterval`
`sys`	`sysinterval`
`tran`	`tranratio`

TMMONITOR Environment Variable

The `TMMONITOR` environment variable enables monitoring for required processes. It can be defined in `ENVFILE` parameter in the UBBCONFIG file `*MACHINES` section or in Tuxedo application startup scripts. Usually Tuxedo client programs use environment variables to control the behavior.

Note: BEA TSAM does not restore the original process monitoring settings if the process is restarted unless `TMMONITOR` is used.

Listing 2-1 provides a `TMMONITOR` example with all four monitoring areas turned on. The policies are using default values.

Listing 2-1 TMMONITOR Environment Variable: Example1

```
TMMONITOR=app,svc,tran,sys::
```

Listing 2-2 provides a `TMMONITOR` example with call path and service monitoring turned on. `appratio` is set to 10 and `svcinterval` is set to 30.

Listing 2-2 TMMONITOR Environment Variable: Example2 pro

```
TMMONITOR=app,svc:appratio=10,svcinterval=30:
```

changemonitor Command

Using the `tmadmin changemonitor` command allows you to dynamically change monitoring settings. Listing 2-3 provides a `changemonitor` usage example.

Listing 2-3 Using changemonitor

```
> help chmo

changemonitor (chmo)[-m machine] | [-g groupname] | [-g groupname -i srvid]
newspec
```

Listing 2-4 provides a `changemonitor` example that enables all the service monitoring for processes running on machine SITE1.

Listing 2-4 changemonitor: Example 1

```
tmadmin

chmo -m SITE1 svc::
```

Listing 2-5 provides a `changemonitor` example that enables service and transaction monitoring for all GROUP1 servers. `svcinterval` is set to 30 seconds.

Listing 2-5 changemonitor: Example 2

```
tmadmin

chmo -g GROUP1 svc,tran:svcinterval=30:
```

Listing 2-6 provides a `changemonitor` example that enables GWTDOMAIN system monitoring with `sysinterval` set to 30 seconds. The GWTDOMAIN is located at group GWGRP, server ID 10.

Listing 2-6 changemonitor: Example 3

```
tmadmin

chmo -g GWGRP -i 10 sys:sysinterval=30:
```

MIB Interface

BEA TSAM framework also opens the Tuxedo MIB interface for developers. The `TA_TMMONITOR` attribute set in the following MIB(5) classes can be used to control BEA TSAM monitoring. `TA_TMMONITOR` accepts the same string format used in the command line and environment variables.

- T_CLIENT Class
- T_SERVER Class
- T_MACHINE Class

For more information, see MIB(5) in reference in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Note: `TA_TMMONITOR` is a local MIB attribute, so when using the MIB for set operation, the `MIB_LOCAL` field must be set. The MIB "`get`" operation is not supported in the current release.

BEA TSAM Console

The monitoring policy can also be configured using the BEA TSAM Console policy management page. The BEA TSAM Console policy management page allows you to:

- create policy definitions
- group policies in a monitoring policy set
- track policy usage

Note: It is not recommended that you use Tuxedo-side control mechanisms and BEA TSAM console-side policy management together; monitoring consistency and accuracy may be affected.

For example, a monitoring policy is created using the BEA TSAM console and applied to Tuxedo components, but is monitoring policy is changed later using Tuxedo-side command line settings. The BEA TSAM is not aware of the Tuxedo-side changes and still shows its original settings.

Cancel Monitoring

If you use the TMMONITOR, changemonitor command, or MIB, cancel monitoring is initiated in the same way as you enable monitoring. Setting the string specification to "::" cancels monitoring on the effected process. The BEA TSAM Console allows to cancel using the GUI interface.

tpcallinfo API

User application can take advantage of the monitoring attributes by using `tpgetcallinfo`. `tpgetcallinfo` is designed for call path monitoring. Using `tpgetcallinfo`, allows applications to make dynamic decisions based on application performance metrics.

For more information, see `tpgetcallinfo()` in the *BEA TSAM Reference Guide*.

LMS (Local Monitor Server)

LMS (Local Monitor Server) is a component of BEA TSAM Agent. It performs the following tasks:

- Act as a data proxy server between BEA TSAM default plug-in and BEA TSAM Manager
- Accept management request from BEA TSAM Manager and apply to Tuxedo domain
- Accept Tuxedo event request from BEA TSAM Manager and post it to Tuxedo Event Broker
- Send Tuxedo domain configuration to BEA TSAM Manager
- Other help functionality

For more information, see LMS in the *BEA TSAM Reference Guide*.

Plug-in Level Events

Overview

You can to define and generate event monitoring using BEA TSAM. Event monitoring data is collected by the BEA TSAM Agent and sent to the BEA TSAM Manager. This process may increase system overhead.

The BEA TSAM Event Plug-in help minimizes system overhead. It checks monitor data against pre-defined rules at the plug-in level. If the specified rule is satisfied, the event is sent to the Tuxedo Event Broker and/or BEA TSAM Manager (specified in the rule definition).

Tuxedo Event Broker plug-in generated events are subscribed by application. The data portion of the event is an FML32 typed buffer which contains generated event information and monitoring data. BEA TSAM Manager generated events can be queried from the BEA TSAM Console `TSAM > Alerts > Events` menu (based on the plug-in contents).

Administration Tasks

Enabling the BEA TSAM Event Plug-in requires the following administrative tasks:

1. Configuring the UBBCONFIG File
2. Composing the Plug-in Rules File
3. Activating the Rules File.

Configuring the UBBCONFIG File

To configure BEA TSAM events plug-in the UBBCONFIG file, do the following two steps:

1. Define `MAXSPDATA` in the `*RESOURCE` section to reserve bulletin board space for storing plug-in event rules.

   `MAXSPDATA` is needed to setup to reserve space in BB to store Plug-in event rules if `the size of rules file` is bigger than 8192 bytes.

   Note: More precisely, it should be *the effective size* instead of the size of rules file. The effective size is approximately equal to: `(the file size) - (size of comments) + (size of ignored optional items)`, all in bytes.

   The `MAXSPDATA` value should not be less than `MAXQUEUES*514+32+max{8192, <the size of rules file>}.`

   The `MAXSPDATA` minimum value can be retrieved using the `ctsamverify` command in `tmadmin` if the rules file exists (`tsamrules` is the rules file name) and is formatted properly.

   ```
   > ctsamverify tsamrules
   ```

   Note: The rule file `<tsamrules>` requires that `MAXSPDATA` is set to at least `<41120>`.

2. Add the Tuxedo Event Broker and/or `LMS` to the `*SERVER` section.

   The Tuxedo Event Broker and/or LMS must be setup in the `*SERVERS` section to enable the BEA TSAM events plug-in. Listing 2-7 displays an LMS setup.A snippet UBBCONFIG is as following:

Listing 2-7 LMS Set in the UBBCONFIG File

```
*SERVERS
TMSYSEVT SRVGRP=SYSGRP SRVID=1 CLOPT="-A --"
TMUSREVT SRVGRP=SYSGRP SRVID=2 CLOPT="-A --"
LMS    SRVGRP=SYSGRP SRVID=3
       CLOPT="-A -- -l <tsam-hostname>:8080/tsam/dataserver -t 60"
```

Composing the Plug-in Rules File

The rules file is a text file which contains BEA TSAM and event plug-in. For more information, see "Configuration Reference" on page 2-17. Listing 2-8 lists an example rules file.

Listing 2-8 Event Plug-in Rules File Example (tsamrules)

```
*GENERAL_RULES

SVCCHECKALL=N
CALLCHECKALL=Y

*SVC_TRIGGER

SVCTRIGGER=TA_SVCRNAM[?]=='TOLOWER':TA_MONERRNO!=0:event(name=TSAM_svcerr,
severity=Warn,destination=tsammanager+eventbroker)

*CALL_TRIGGER

CALLTRIGGER=TA_SERVERNAME=='simpserv':TA_MONELAPSETIME>=5000:event(name=TS
AM_longcall,severity=Warn,destination=eventbroker+tsammanager)

*BBL_TRIGGER

BBLTRIGGER=TA_SVCRNAM[?]=='TESTCALL':TA_MONEXECTIME>=30000:event(name=TSAM
_hang,severity=Warn,destination=tsammanager+eventbroker)

*REPORT_POLICY
```

```
SVCSENDTOLMS=Y
CALLSENDTOLMS=Y
```

Activating the Rules File

The plug-in rules file is loaded at startup if the name of the file is specified using the `TSAMPLUGINRULES` (Listing 2-9) environment variable in `MASTER` node. The loaded rules are saved to the Bulletin Board and automatically sent to all machines in MP mode.

The plug-in rules file can be loaded or reloaded later at runtime using `tmadmin` from `MASTER` node (Listing 2-10). Two `tmadmin` commands (`configtsam` and `ctsamverify`) support the event plug-in rules file operation at runtime. For more in formation, see `tmadmin(1)` in the *BEA Tuxedo Command Reference*.

Note: Please note that all previous setting will be lost after reloading rules file.

The `BBL` (or `DBBL` in MP mode) logs the messages LIBTUX_CAT:6775 or LIBTUX_CAT:6776 in ULOG if the rules file does not load successfully. New rules will not take effect if the LIBTUX_CAT:6775 message is logged since the rules file was not loaded successfully.

```
LIBTUX_CAT:6775:ERROR: Failed to load TSAM Event Trigger rules from file
<%s>. Reason:<%s>
```

```
LIBTUX_CT:6776:INFO: Load TSAM Event Trigger rules from file <%s>. Rules
size <%d>
```

Listing 2-9 Load Rules File at Startup Example

```
TSAMPLUGINRULES=$APPDIR/tsamrules; export TSAMPLUGINRULES

tmboot -y\
```

Listing 2-10 Load Rules File at Runtime Example

```
tmadmin - Copyright (c) 1996-1999 BEA Systems, Inc.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by BEA Systems, Inc.
```

```
Tuxedo is a registered trademark.

> configtsam load tsamrules
INFO: TSAM Event Trigger successfully loaded.
```

Check Plug-in Generated Events Using the BEA TSAM Console

BEA TSAM Manager plug-in generated events can be queried from the BEA TSAM console `TSAM > Alerts > Events` page. These events belong to a separate event catalog named, `Plugin`, that distinguishes it from events generated by the BEA TSAM Manager alert definition (which belongs to another catalog named Alert).

Subscribing to Plug-in Generated Events

Tuxedo Event Broker plug-in generated events are subscribed by application. The data portion of the event is an FML32 typed buffer which contains information of the generated event and monitor data.

Three trigger types are defined in the rules file:

- SVC_TRIGGER
- CALL_TRIGGER
- BBL_TRIGGER.

Table 2-2 lists the common FML 32 data fields for all events. Table 2-3, Table 2-4 and Table 2-5 show other available data fields.

Table 2-2 Common FML32 Fields for Plug-in Events

Name	Description
`TA_MONEVENTNAME`	Event name specified in event action.
`TA_MONSEVERITY`	Event severity specified in event action.
`TA_GRPNO`	Group number.
`TA_SRVGRP`	Group name.

Table 2-2 Common FML32 Fields for Plug-in Events

Name	Description
TA_SRVID	Server ID.
TA_MONLOCATION	Location where the monitor data is collected, in the format of: <DOMAINID>:<master machine name>:<IPCKEY> <Logical Machine ID> <Group Name> <Process Name> <SRVID if process is a server> <Process ID>.
TA_MONLOGTIMESEC	The second when the event is posted.
TA_MONLOGTIMEUSEC	The microsecond when the event is posted.

Table 2-3 SVC_TRIGGER FML32 fields

Name	Description
TA_MONSVCNAME	Service name
TA_MONMSGQUEUED	Number of messages queued on the IPC queue of the server which providing this service
TA_MONERRNO	Return code of the service
TA_MONURCODE	User return code of the service
TA_MONEXECTIME	Response time of the service

Table 2-4 CALL_TRIGGER FML32 fields

Name	Description
TA_MONSVCNAME	The service name
TA_MONELAPSETIME	The elapse time in milliseconds since the call started from the monitoring initiator
TA_MONDEPTH	The call path tree depth, counting from 0 (initiator).
TA_MONCORRID	The correlation id of the monitored call

Table 2-4 CALL_TRIGGER FML32 fields

Name	Description
TA_MONERRNO	Return code of the service
TA_MONURCODE	User return code of the service

Table 2-5 BBL_TRIGGER FML32 fields

Name	Description
TA_MONSVCNAME	Service name
TA_MONEXECTIME	The elapsed time in milliseconds since the service started execution.

Configuration Reference

A BEA TSAM rules file is made up of several possible specification sections. Lines beginning with an asterisk (*) indicate the beginning of a specification section. Each line contains the name of the section immediately following the *. Allowable section names are:

- GENERAL_RULES
- SVC_TRIGGER
- CALL_TRIGGER
- BBL_TRIGGER
- REPORT_POLICY

Parameters are generally specified using the following format: `KEYWORD = value`. A space (space or tab character) is allowed on either side of the equal sign (=). This format sets a value to `KEYWORD`. Valid keywords are described within each section.

The rules file may contain comments that start with a ‘#’ (pound sign). A “newline” ends a comment. Blank lines and comments are ignored.

These sections can be divided into three independent groups:

- Plug-in Event rules (the first three sections)

- BBL service execution time trigger (BBL_TRIGGER)
- BEA TSAM Manager data reporting policy (REPORT_POLICY).

GENERAL_RULES

This section provides global event plug-in settings. Lines in the `GENERAL_RULES` section use the following format: `KEYWORD=value`; where `KEYWORD` is the name of the parameter, and the value is its associated value. Valid `KEYWORD`s are as follows:

`SVCCHECKALL={ Y | N }`

Specifies whether or not all triggers defined in SVC_TRIGGER section should be evaluated. If `SVCCHECKALL` is not specified, the default is `Y`.

If `SVCCHECKALL` is set to `N`, evaluation stops when a rule is evaluated as `true`. This means that, at most, one event is generated in this case. The rules are evaluated in the same order as they appear in the rules file.

`CALLCHECKALL={ Y | N }`

Specifies whether or not all triggers defined in CALL_TRIGGER section should be evaluated. If `CALLCHECKALL` is not specified, the default is `Y`.

If `CALLCHECKALL` is set to `N`, evaluation stops when a rule is evaluated as`true`. This means that, at most, one event is generated in this case. The rules are evaluated in the same order as they appear in the rules file.

SVC_TRIGGER

This section provides service event plug-in trigger rules based on service monitor data. The check is done upon the completion of service execution.

Lines in the `SVC_TRIGGER` section use the following format: `KEYWORD=value`; where `KEYWORD` is the name of the parameter, and value is its associated value. Valid `KEYWORD`s are as follows:

`SVCTRIGGER=components filter:metrics filter:actions`

Specifies for which Tuxedo server (`components filter`, optional), under what condition (`metrics filter`) Plug-in should execute the `actions`. For the syntax for filter and action, refer to Plug-in Event Trigger Format.

There can be multiple SVCTRIGGERs defined in this section, each occupied one line.

CALL_TRIGGER

This section provides call path trigger rules for Plug-in event based on call path monitor data. The check is done in all steps of call path.

Lines in the `CALL_TRIGGER` section are of the form: KEYWORD=value where KEYWORD is the name of the parameter, and value its associated value. Valid KEYWORDs are as follows:

`CALLTRIGGER=components filter:metrics filter:actions`

Specifies for which Tuxedo server (`components filter`, optional), under what condition (`metrics filter`) Plug-in should execute the `actions`. For the syntax for filter and action, refer to Plug-in Event Trigger Format.

There can be multiple CALLTRIGGER definitions in this section, each occupying one line.

BBL_TRIGGER

This section provides trigger rules used by BBL to post events when a service execution time is longer than pre-defined value. BBL will check executing services in every `SCANUNIT`. Events only be posted once for a hang service.

This enable Tuxedo to detect potential service “hang” problem but bring no impact to the server which executing the service. On the contrary, the server will be terminated with the service time-out feature.

Lines in the `BBL_TRIGGER` section are of the form: KEYWORD=value where KEYWORD is the name of the parameter, and value its associated value. Valid KEYWORDs are as follows:

`BBLTRIGGER=components filter:metrics filter:actions`

Specifies for which Tuxedo server (`components filter`, optional), under what condition (`metrics filter`) BBL should execute the `actions`. For the syntax for filter and action, refer to Plug-in Event Trigger Format.

There can be multiple BBLTRIGGER defined in this section, each occupied one line.

REPORT_POLICY

This section specifies whether or not collected service and call path monitor data should report to BEA TSAM Manager when default Plug-in is used. Lines in the `REPORT_POLICY` section are of the form: KEYWORD=value where KEYWORD is the name of the parameter, and value its associated value. Valid KEYWORDs are as follows:

`SVCSENDTOLMS={ Y | N }`

Specifies whether or not service monitor data should report to BEA TSAM Manager by default Plug-in. The default value is `Y`.

`CALLSENDTOLMS={ Y | N }`

Specifies whether or not call path monitor data should report to BEA TSAM Manager by default Plug-in. The default value is `Y`.

Plug-in Event Trigger Format

The Plug-in Event trigger contains three fragments separated by a ':' (colon): `components filter`, `metrics filter`, and `actions`.

The `filter` fragment is a boolean expression based on FML32 fields provided by Tuxedo infrastructure. For more information, see Boolean Expressions of Fielded Buffers in the *Programming a BEA Tuxedo ATMI Application Using FML*, Field Manipulations chapter.

The events plug-in first evaluates `components filter` with the location information of the process where the monitor data is collected. If the evaluation output is `false`, this rule is ignored. Otherwise `metrics filter` will be evaluated and if the result is `true`, Plug-in will execute the `actions`.

The available FML32 fields for `components filter` is listed in Table 2-6. The available FML32 fields for `metrics filter` is listed in Table 2-7, Table 2-8 and Table 2-9, for SVC_TRIGGER, CALL_TRIGGER and BBL_TRIGGER respectively.

Each `action` is defined in the following format: `name(key1=value1,key2=value2+value3,...)`. `name` specifies the action type. Configurable parameters for the action are specified in `()` following the `name` using the following format: `KEYWORD=values`, separated by a `','`(comma). Separating each value with a `'+'`(plus) if a parameter can contain multiple values. Separating actions with `','`(comma) if there are more than one actions are defined.

Only the event action is supported, and can only be defined once. It is used to post event to Tuxedo Event Broker and/or BEA TSAM Manager, when `components filter` and `metrics filter` are all evaluated as `true`. The parameters for event is listed in Table 2-10 lists event parameters.

Table 2-6 Components Filter FML32 Fields

Name	Description
`TA_LMID`	Logical machine ID. **Note:** Not available for BBL_TRIGGER.
`TA_SERVERNAME`	Server name. **Note:** Not available for BBL_TRIGGER.
`TA_SRVGRP`	Group name

Table 2-6 Components Filter FML32 Fields

Name	Description
`TA_GRPNO`	Group number
`TA_SRVID`	Server ID
`TA_SVCRNAM`	For SVC_TRIGGER and CALL_TRIGGER, it's the service name(s) provided by the server. It will have multiple occurrences if the server provides multiple services. Since the order of service is undefined, please use `TA_SVCRNAM[?]` if you want to check for service name in the filter. For BBL_TRIGGER, it's the service name under checking.

Table 2-7 SVC_TRIGGER Metrics Filter FML32 fields

Name	Description
`TA_MONSVCNAME`	Service name
`TA_MONMSGQUEUED`	Number of messages queued on the IPC queue of the server which providing this service
`TA_MONERRNO`	Return code of the service
`TA_MONURCODE`	User return code of the service
`TA_MONEXECTIME`	Response time of the service

Table 2-8 CALL_TRIGGER Metrics Filter FML32 fields

Name	Description
`TA_MONSVCNAME`	The service name
`TA_MONELAPSETIME`	The elapse time in milliseconds since the call started from the monitoring initiator
`TA_MONDEPTH`	The call path tree depth, counting from 0 (initiator).
`TA_MONCORRID`	The correlation id of the monitored call

Table 2-8 CALL_TRIGGER Metrics Filter FML32 fields

Name	Description
`TA_MONERRNO`	Return code of current service. It only applies to reply stage
`TA_MONURCODE`	User return code of current service. It only applies to reply stage

Table 2-9 BBL_TRIGGER Metrics Filter FML32 Fields

Name	Description
`TA_MONSVCNAME`	The service name.
`TA_MONEXECTIME`	The elapsed time in milliseconds since the service started executing.

Table 2-10 Event Parameters

Name	Description
`name`	Event name, a string of at most 31 characters. Valid characters are ‘0’..’9’, ‘a’..’z’, ‘A’..’Z’ and ‘_’.
`severity`	Message severity level. Can be one value of `Information`, `Warn`, `Critical`, and `Fatal`. The default value is `Information`.
`destination`	There are two supported destinations: `eventbroker` (Tuxedo Event Broker) and `tsammanager` (BEA TSAM Manager). This parameter can contains multiple values (separated by `‘+’`). The default value is `tsammanager`.

BEA TSAM Manager

This topic contains the following sections:

- Overview
- Configuring BEA TSAM Manager

Overview

The BEA TSAM Manager is the data manipulation and representation component of BEA TSAM. It is a J2EE application. The BEA TSAM Manager provides the following functionality:

- Communicates with the BEA TSAM Agent for performance metrics, configuration information and other utility messages.
- Maintains persistent data storage
- Provides a Web console for BEA TSAM administration, monitored data presentation and alerts management.

The BEA TSAM Manager ships with Apache Tomcat as the Java application server. "`tsam`" is the Web application name. The BEA TSAM Manager requires JRE 1.5 or above.

BEA TSAM Data Server

The BEA TSAM Data Server is the communication interface to BEA TSAM Agent. It includes two key servlets:

- `dataserver`

Receives all messages from the BEA TSAM Agent. It extracts business data from the HTTP/XML body, collects the data, and stores it in the database.

- `requestserver`
- actively delivers requests from the BEA TSAM Manager to the BEA TSAM Agent. The `dataserver` servlet URL must be set properly in order to work with the LMS (local monitor server). AfterBEA TSAM Manager is installed the dataserver URL is: `host:port/tsam/dataserver`.

  "`host`" is the full domain name or IP address where BEA TSAM Manager installed. "`port`" is the Tomcat listening port. "`tsam/dataserver`" is the data server fixed service endpoint. The LMS distinguishes the "requestserver" URL based on the "dataserver" URL.

For more information, see the BEA TSAM Deployment Guide.

Note: From an HTTP perspective, the BEA TSAM Agent LMS is the HTTP client, and the BEA TSAM Manager is the HTTP server. If a firewall is deployed between the BEA TSAM Manager and Tuxedo applications, the firewall must allow the LMS to issue HTTP requests to the BEA TSAM Manager.

Database

BEA TSAM uses a relational database to store the following information:

- Performance metrics collected by the BEA TSAM Agent
- Tuxedo component information from the BEA TSAM Agent
- User account information
- Alerts and events

The BEA TSAM Manager includes Apache Derby as the default database (for evaluation purposes). Oracle database are also supported by using pre-built SQL script (An Oracle database is not included with BEA TSAM Manager). For more information, see the TSAM Manager Deployment Guide.

BEA TSAM Console

TSAM Console is the Web application which provides a GUI interface for administration and data presentation. For more information, see the BEA TSAM Console User Guide.

Configuring BEA TSAM Manager

All configuration parameters for the BEA TSAM Manager are located in the

`web.xml` file(short for `<TSAMDIR>/apache-tomcat-5.5.17/webapps/tsam/WEB-INF/web.xml`) and `faces-config.xml` file (short for `<TSAMDIR>/apache-tomcat-5.5.17/webapps/tsam/WEB-INF/config/faces-config.xml`).

Table 3-1 and Table 3-2 provide detailed `web.xml` and `faces-config.xml` configuration information.

The BEA TSAM Manager runtime log is based on Jarkata commons logging with Apache log4j as the logging implementation. The log setting file is located at: `apache-tomcat-5.5.17/webapps/tsam/WEB-INF/classes named log4j.properties`.

For more information, see the Apache log4j Web site.

Table 3-1 web.xml Configuration

Name	Description
`BIRT_IMAGE_LIVETIME`	Specifies how long TSAM Manager temporary image files remain in the file system (in minutes).
`tsam.jdbc.url`	JDBC connection string for TSAM Manager database. Derby Example: `jdbc:derby://localhost:1527/tmonitordb` Oracle Example: `jdbc:oracle:thin:@localhost:1521:orcl.`
`tsam.jdbc.username`	TSAM Manager database connection user name.
`tsam.jdbc.password`	TSAM Manager database connection password.
`tsam.config.windowhs`	Active interval (in seconds) for TSAM Manager house keeping thread.
`tsam.config.maxappactive`	Specifies how long active monitored application data are cached (in seconds). The default value is 600 seconds, or 10 minutes.

Table 3-1 web.xml Configuration

Name	Description
`tsam.config.maxappdone`	Specifies how long finished monitored application data are cached (in seconds). The default value is 1800 seconds, or 30 minutes.
`tsam.config.maxappsize`	Specifies the size of the cache which holds active and finished monitored application data. The default value is 1000.
`tsam.config.timeoutwithtuxedo`	Specifies the time-out value (in seconds) for communication originated from TSAM Manager to Tuxedo. The default value is 30 seconds.

Table 3-2 faces-config.xml Configuration

Name	Description
Time Zone	Specifies the TSAM Manager time zone date/time fields. Attribute: "`timeZoneID`" for "`util`" bean. If not set, the default time zone (where TSAM Manager is located) is used.

Notes: When you want to customize the BEA TSAM Manager use `web.xml` and `faces-config.xml` as templates. Remember to backup these file before changing their configurations

There are other configuration settings in `web.xml` and `faces-config.xml` that are not included in this document. It is strongly advised that you do not alter these settings. If these settings altered, BEA TSAM Manager may function incorrectly or not function at all.