

HP TransactionVision

Software Version: 7.50

Sensor Installation and Configuration Guide

Software Release Date: May 2008



Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

© Copyright 2008 Hewlett-Packard Development Company, L.P.

Trademark Notices

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle® is a registered trademark of Oracle Corporation and/or its affiliates.

UNIX® is a registered trademark of The OpenGroup.

Documentation Updates

This guide's title page contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

<http://h20230.www2.hp.com/selfsolve/manuals>

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP Software Support Online web site at:

<http://www.hp.com/go/hpsoftwaresupport>

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software Support Online provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the HP Software Support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

Contents

1 TransactionVision Sensor Overview	9
Available Sensor Types	9
WebSphere MQ (WMQ) Sensors	10
Java Sensors	11
CICS Sensor	12
Installation Overview	12
Upgrading from Previous Sensor Releases.....	12
Additional TransactionVision Resources	12
2 Installing and Configuring Java Agent	15
About Installing and Configuring the Java Agent.....	15
Installation Files.....	16
Installing and Configuring the Java Agent on Windows.....	16
Launching the Installer on Windows.....	16
Running the Installation on Windows.....	17
Configuring the Java Agent to Work as a TransactionVision Java Sensor on Windows	18
Post Configuration Options.....	24
Enable Java Agent in Applications on Windows	24
Enable Java Agent in Applications on UNIX.....	25
Installing and Configuring the Java Agent on UNIX	26
Downloading the Installer on UNIX	26
Running the Installation on UNIX	26
Configuring the Java Agent to Work as a TransactionVision Java Sensor on UNIX.....	27
Configuring the Java Agent on UNIX in Graphical Mode	27
Configuring the Java Agent on UNIX in Console Mode	28
Silent Installation of the Java Agent	32
Running the JRE Instrumenter	33
JRE Instrumenter Processing	34
Running the JRE Instrumenter Manually	34
Running the JRE Instrumenter on a Windows Machine	34
Running the JRE Instrumenter on a UNIX Machine	37
Configuring the Application Servers.....	40
About Configuring the Application Server	40
Configuring WebSphere Application Servers	41
WebSphere 5.x and 6.0	41
WebSphere 6.1	44
Running the JRE Instrumenter for WebSphere IDE	45
Using the JMS Sensor with the WebSphere Application Server	46
Adding Interceptors for Sensors	46

Configuring WebLogic Application Servers	46
WebLogic 8.1	46
WebLogic 9.2	50
Configuring Remote-Started WebLogic Managed Servers	51
Configuring Messaging System Providers	52
IBM WebSphere MQ	52
TIBCO EMS	53
Progress SonicMQ	53
BEA WebLogic JMS	53
3 Installing WebSphere MQ and User Event Sensors on UNIX Platforms	55
Installing Sensors	55
Installation Files	55
Installation Steps	55
Rebinding the WebSphere MQ Sensor on AIX	56
Uninstalling Sensors	57
4 Installing WebSphere MQ and User Event Sensors on Windows	59
Initial Installation	59
Upgrade Installation	60
Modifying the Installation	62
Uninstalling Sensors	63
5 Installing Sensors on i5/OS	65
6 Installing Sensors on z/OS	67
Installing the CICS, WebSphere MQ Batch, and WebSphere MQ IMS Sensors on IBM z/OS	68
Installing the WebSphere MQ CICS and WebSphere MQ IMS Bridge Sensors on IBM z/OS	74
Before You Install the WebSphere MQ Sensor for CICS	74
SMPE Installation Procedure	74
Configuring SLMC for CICS	81
Additional Setup for the WebSphere MQ IMS Bridge Sensor	83
7 Configuring WebSphere MQ Sensors	85
Configuring the WebSphere MQ Sensor Library	85
Distributed Platforms	85
z/OS Batch, IMS, and WebSphere MQ-IMS Bridge	86
z/OS CICS	87
i5/OS	87
Configuring Sensor Logging	88
Setting the Configuration Queue Name	88
UNIX, Windows, and i5/OS	88
IBM z/OS Batch, IMS and WebSphere MQ-IMS Bridge	88
Setting the Configuration Queue Check Interval	91
Configuring the WebSphere MQ Messaging System Provider	91
Configuring the WebSphere MQ API Exit Sensor	91
Configuring the API Exit Sensor on Distributed and i5/OS Platforms	92
Linking the WebSphere MQ API Exit Sensor	92

New Stanzas	93
Stanza Attributes and Values	93
Configuring the API Exit Sensor on Windows Platforms	94
Identifying Programs to Monitor	94
Discarding WebSphere MQ Events on TransactionVision Queues	95
WebSphere MQ Sensors and FASTPATH_BINDING	96
Using Sensors with WebSphere MQ Samples	96
WebSphere MQ Client Application Monitoring	96
Distributed Monitoring	96
Centralized Monitoring	97
Installation and Configuration Considerations	99
Using the WebSphere MQ-IMS Bridge Sensor	99
Sensor Setup	100
WebSphere MQ-IMS Bridge Sensor Operation	100
The TVISIONB Buffer Queue	101
Event Data	101
Beans.xml	102
IMSBridgeObject.xml	102
Data Collection Filters and Queries	103
Using the WebSphere Business Integration Sensor	103
Message Brokers Toolkit for WebSphere Studio Integration	104
TransactionVision User-Defined Node Installation for WBIMB	104
Node Insertion	104
8 Configuring the Proxy Sensor	105
Application Requirements	105
Enabling the Proxy Sensor	105
Configuring the Proxy Definition File	106
Subelements	106
Optional Attributes for the Proxy Element	107
Configuring the User Interface	107
9 Configuring Sensor Logging	109
Log Files	109
Java Sensors	109
WebSphere MQ Sensors	109
Circular Logging	109
Maximum Log File Size	110
Maximum Number of Backup Log Files	110
Changing from Circular to Linear Logging	110
Trace Logging	111
Configuring Separate Log Files for Multiple Sensor Instances	111
Using Windows and UNIX System Logs	112
Windows Event Appender	112
UNIX Event Appender	113

A Utilities Reference	115
SetupModule	115
Description	115
Options	115
MigrateConfig	116
Location	116
Description	116
rebind_sensor	117
Location	117
Description	117
Syntax	117
Options	117
B Configuration Files	119
License.properties	119
Performance.properties	119
Sensor.properties	119
SensorConfiguration.xml	120
Setup.properties	120
tvision-wl-sensorconfig.properties	121
tvision-ws-sensorconfig.properties	121
C Additional z/OS Settings	123
RACF Authorizations	123
For the TransactionVision CICS Sensor	123
For the TransactionVision CICS WebSphere MQ (WMQ) Sensor	124
For the TransactionVision IMS WebSphere MQ (WMQ) Sensor	124
Firewall Settings	125
MIPS Required	125
Index	127

1 TransactionVision Sensor Overview

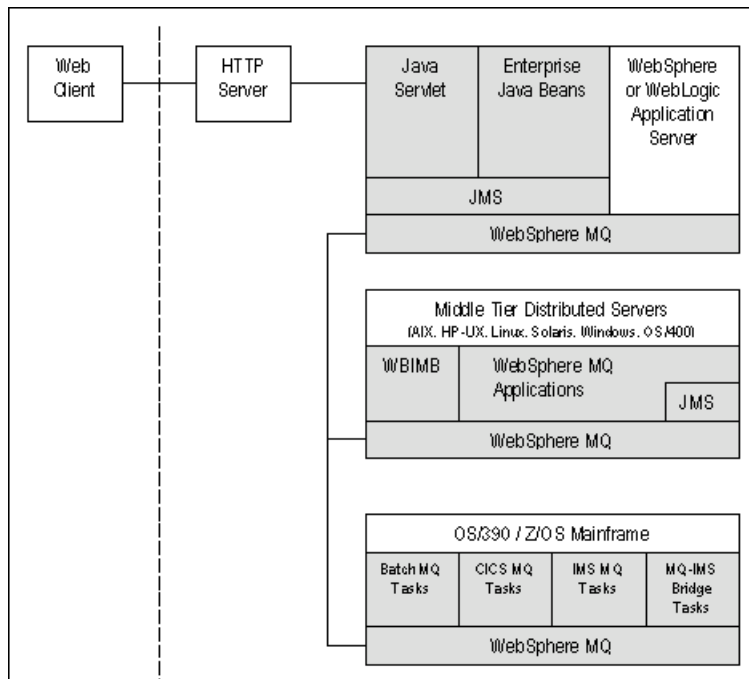
TransactionVision Sensors collect transactional events from the various applications involved in your distributed transactions. Sensors are lightweight libraries or exit programs that are installed on each computer in your environment. Each Sensor monitors calls made by supporting technologies on that system and compares them against filter conditions. If the call matches the filter conditions, the Sensor collects entry information about the call, then passes the call on to the appropriate library for processing. When the call returns, the Sensor collects exit information about the call. It then combines the entry and exit information into a TransactionVision event, which it forwards to the Analyzer by placing it on a designated event queue.

Available Sensor Types

TransactionVision provides the following types of Sensors:

- [WebSphere MQ \(WMQ\) Sensors](#)
- [Java Sensors](#)
 - [Servlet Sensor](#)
 - [JMS Sensor](#)
 - [EJB Sensor](#)
 - [JDBC Sensor](#)
- [CICS Sensor](#)

In the following diagram, shaded areas represent the parts of a web application for which TransactionVision can track events:



WebSphere MQ (WMQ) Sensors

- The **WebSphere MQ Sensor** tracks MQ API calls. These API calls include the entire MQ API set, the major APIs being MQPUT, MQGET, MQCONN, MQDISC, MQOPEN, MQCLOSE, etc. There are two types of WebSphere MQ Sensors provided by TransactionVision on distributed platforms: the WebSphere MQ Library Sensor and the WebSphere MQ API Exit Sensor. Both of these Sensors report the same information from an MQ API call. They differ primarily in the mechanism by which they intercept MQ API calls, their usage, and the amount of data they collect from the system.
- The **WebSphere MQ Library Sensor** intercepts a WebSphere MQ API call by the shared library (or DLL) interception method on distributed platforms. This involves placing the TransactionVision Sensor libraries before the WebSphere MQ libraries in the application library path. This method is useful if you need to track MQ APIs for specific applications.
- The **WebSphere MQ API Exit Sensor** uses the WebSphere MQ API exit support available on distributed platforms in WebSphere MQ v5.3 and later. This Sensor is registered as an exit to the queue manager and invoked when any program connecting to the queue manager invokes a WebSphere MQ API. This method is recommended to collect MQ events from all applications on a queue manager and in particular the listener and the channel agents.
- **z/OS WebSphere MQ Sensors** are provided for tracking MQI API calls in the CICS, batch and IMS environments on the IBM z/OS system. In the CICS environment, the API crossing exit provided by the CICS adapter for WebSphere MQ is used to intercept the MQ API. In the batch and IMS environments, the application has to be re-bound with the Sensor to intercept MQ API calls.

The following supplemental Sensors are available for WebSphere MQ:

- The **Proxy Sensor** correlates business transactions into processes that are not monitored using the TransactionVision Sensor libraries (for example, events between a Sensored application and an application running on a system where no Sensor is installed such as an external partner system).
- The **WebSphere Business Integration Sensor** (previously known as the MQSI Sensor) distinguishes the various message flows and identifies individual logical transaction paths within WBI. This Sensor is a WBI plugin that provides a trace node, which is inserted into the normal execution path of a message flow, and a failure node, that is inserted into the failure path of a message flow. These nodes generate a MQSI2TRACE event that allows tracking of the message flow within WBI.
- The **WebSphere MQ-IMS Bridge Sensor** tracks WebSphere MQ-IMS bridge messages rather than the WebSphere MQ API calls made by the calling applications. The MQ-IMS Bridge is a component that enables WebSphere MQ applications to invoke IMS transactions and receive their reply messages. The MQ-IMS Bridge Sensor tracks MQ messages coming into the bridge and correlates them with the reply received from IMS. Two events, MQIMS_BRIDGE_ENTRY and MQIMS_BRIDGE_EXIT are generated for every message coming in and going out of the bridge. These events contain the MQ message header and information about which IMS transaction is invoked.

Java Sensors

The capabilities of the TransactionVision Java Sensors (JMS, Servlet, EJB and JDBC) and the Diagnostics Java Probe are combined into a single component, HP Diagnostics/TransactionVision Java Agent. The Java Agent instruments and captures events from applications and sends the information to a Diagnostics Server and/or to a TransactionVision Analyzer. In this release the Java Agent can be configured to serve as a Java Probe in a Diagnostics environment or as a Java Sensor in a TransactionVision environment. For combined environments, the agent can simultaneously serve as both the Probe and the Sensor. See [Chapter 2, Installing and Configuring Java Agent](#) for details.

- The **Servlet Sensor** tracks servlet methods in a J2EE application server. This Sensor tracks HTTP calls such as HTTP_POST, HTTP_GET, HTTP_PUT, etc., which result in method calls into the J2EE container. The Servlet Sensor tracks these method invocations by instrumenting the servlet to collect events at the entry and exit of each call.
- The **JMS Sensor** tracks WebSphere MQ Java Message Service or TIBCO EMS events from standalone Java applications as well as from J2EE application servers. This Sensor tracks JMS interface methods such as send, receive, etc. These methods are tracked by instrumenting the JMS library to collect events at the entry and exit of each call.
- The **EJB Sensor** tracks transactions through business logic within a J2EE application server. This Sensor tracks all public business methods in an entity bean, session bean or message driven bean. In addition to the business methods, this Sensor tracks the ejbCreate, ejbPostCreate, ejbRemove, ejbLoad, ejbStore and onMessage methods. These methods are instrumented by the Sensor to collect events at the entry and exit of each call.
- The **JDBC Sensor** allows users to collect and analyze API and timing information on SQL calls and transactions made to a relational database through the JDBC API.

CICS Sensor

The CICS Sensor collects non-WebSphere MQ CICS events to track transactions in a mainframe environment. The CICS Sensor collects data for five types of events: file control, temporary storage, transient data, interval control, and program control. For all types, it tracks information such as Transaction ID, User ID, Terminal ID and SYSID. Other information collected depends on the event type.

Installation Overview

The TransactionVision installation consists of the following three TransactionVision packages:

- The Analyzer binaries and configuration files. Install this package on all hosts where you want the Analyzer service to run. For instructions on installing and configuring the Analyzer, see the *TransactionVision Analyzer Installation and Configuration Guide*.
- The TransactionVision Web User Interface, which consists of WebSphere or WebLogic related files such as the .ear file, JSPs, servlets, and so on. Install this package on the hosts that users and administrators will access via web browsers to use and manage TransactionVision. For instructions on installing and configuring the Web User Interface, see the *TransactionVision Web Application Installation and Configuration Guide*.
- TransactionVision Sensors. Install these packages on all hosts running applications that you wish to collect information about. This guide provides instructions for installing Sensors.

Installation steps vary for the different TransactionVision components on different platforms.

Before installing Sensors, make sure the systems you are installing on meet the software requirements for TransactionVision. Requirements vary for each package (Sensors, Analyzer, and Web User Interface). See the *TransactionVision Release Notes* for detailed software requirements.

Upgrading from Previous Sensor Releases

The Java Servlet, JMS and EJB Sensors from TransactionVision 5.00 can be used with the TransactionVision 7.50 Analyzer. TransactionVision 7.50 Sensors may not be used with older versions of the TransactionVision Analyzer.

Additional TransactionVision Resources

The following documents are provided with TransactionVision:

- The *TransactionVision Planning Guide* provides information to help you plan the TransactionVision implementation in your environment.
- The *TransactionVision Web Application Installation and Configuration Guide* provides instructions for installing and configuring the TransactionVision web user interface. This file is also available from the TransactionVision Help menu.

- The *TransactionVision Analyzer Installation and Configuration Guide* provides instructions for installing and configuring the TransactionVision Analyzer, and setting up your database for the Analyzer. This file is also available from the TransactionVision Help menu.
- The *TransactionVision Administration Guide* provides instructions managing user accounts and communication links, configuring projects and data collection filters, and managing services and schemas. This file is also available from the TransactionVision Help menu.
- The *TransactionVision User's Guide* provides instructions using TransactionVision analysis views. This file is also available from the TransactionVision Help menu.
- The *TransactionVision Advanced Customization Guide* provides information for creating custom beans and reports for use with TransactionVision.
- The *TransactionVision Security Guide* provides an overview of the security features and setup procedures of TransactionVision. These features and procedures ensure that data collected by TransactionVision is secure and accessible to the appropriate people.

2 Installing and Configuring Java Agent

This chapter provides instructions on installing and configuring the HP Diagnostics/TransactionVision Java Agent on Windows and UNIX.

This chapter includes:

- [About Installing and Configuring the Java Agent](#)
- [Installing and Configuring the Java Agent on Windows](#)
- [Installing and Configuring the Java Agent on UNIX](#)
- [Silent Installation of the Java Agent](#)
- [Running the JRE Instrumenter](#)
- [Configuring the Application Servers](#)
- [Configuring Messaging System Providers](#)

About Installing and Configuring the Java Agent

The Java Agent combines the capabilities of the TransactionVision Java Sensors (JMS, Servlet, JDBC and EJB) and the Diagnostics Java Probe into a single component. The Java Agent can be configured to serve as a J2EE Probe in a Diagnostics environment or as a Java Sensor in a TransactionVision environment and for combined environments, the agent can simultaneously serve as both the Probe and the Sensor.

To use the Java Agent as a TransactionVision Java Sensor, you need to perform the following operations:

- 1 Install the HP Diagnostics/TransactionVision Java Agent.

The Java Agent is installed on the machine hosting the application that you want to monitor. See [Installing and Configuring the Java Agent on Windows](#) on page 16 and [Installing and Configuring the Java Agent on UNIX](#) on page 26.

- 2 Configure the Java Agent.

The Java Agent is configured to function as a TransactionVision Java Sensor, a J2EE Probe or both. This guide provides instructions for configuring the Java Agent as a TransactionVision Java Sensor. See [Configuring the Java Agent to Work as a TransactionVision Java Sensor on Windows](#) on page 18 and [Configuring the Java Agent to Work as a TransactionVision Java Sensor on UNIX](#) on page 27.

- 3 Configure the application server.

To allow the Java Sensor to monitor an application, you need to instrument the JRE (see [Running the JRE Instrumenter](#) on page 33) and configure the application server (see [Configuring the Application Servers](#) on page 40).

Installation Files

The following table shows the installation file names for the TransactionVision Java Agent for each platform.

Platform	Files
Windows	JavaAgentSetup_win_7_50.exe
AIX	JavaAgentSetup_ibm_7_50.bin
Linux	JavaAgentSetup_linux_7_50.bin
Solaris	JavaAgentSetup_sol_7_50.bin

Installing and Configuring the Java Agent on Windows

The following steps provide detailed instructions for installing the Java Agent on a Windows machine. These instructions also apply when you are installing the Java Agent on a UNIX machine using the graphical installer.

If there is a pre-existing installation of the Java Agent, the legacy J2EE Probe, or the legacy TransactionVision 5.0 Sensors on the host machine, you must uninstall it before you install the Java Agent.

This section includes:

- [Launching the Installer on Windows](#)
- [Running the Installation on Windows](#)
- [Configuring the Java Agent to Work as a TransactionVision Java Sensor on Windows](#)

Launching the Installer on Windows

You may launch the Java Agent installer from the HP Software web site, from the Diagnostics or TransactionVision product disk, or from the Downloads page in Business Availability Center.

You must be a user in the Administrators group to install the Java Agent.

To launch the installer from the HP Software Web site:

- 1 Go to the HP Software Web site's Download Center.
- 2 In the Quick Search section, in the Products list, click Diagnostics and click Search.
- 3 Under Software Trial, select the appropriate link.
- 4 Follow the download instructions on the Web site.
- 5 Continue with [Running the Installation on Windows](#) on page 17.

To launch the installer from the Diagnostics product installation disk:

- 1 Run the **setup.exe** file in the root directory of the installation disk. The Diagnostics Setup program begins and displays the installation menu page.


- 2 From the installation menu page, select **Diagnostics/TransactionVision Agent for Java** to launch the installer.

To launch the installer from the TransactionVision product installation disk:

- 1 From the <**HP TransactionVision Installation Disk**>\TransactionVision directory, run the executable file **JavaAgentSetup_win_750.exe**.
- 2 Continue with “Running the Installation on Windows” on page 17.

To launch the installer from Business Availability Center:

- 1 Select **Admin > Platform** from the top menu in Business Availability Center, and click the **Setup and Maintenance** tab.
- 2 On the Downloads page, click the appropriate link to download the Java Agent installer for Windows.

 The Java Agent installers are available in Business Availability Center only if you provided the path to the Java Agent installers directory during the installation of the Diagnostics Server in Command mode.

- 3 Continue with [Running the Installation on Windows](#) on page 17.

Running the Installation on Windows

After you have launched the installer, the software license agreement opens and you are ready to run the installation.

To install the Java Agent on a Windows machine:

- 1 **Accept the end user license agreement.**

Read the agreement and select **I accept the terms of the license agreement**.

Click **Next** to proceed.

- 2 **Specify the location where you want to install the agent.**

Accept the default directory or select a different location either by typing the path to the installation directory into the Installation Directory Name box, or by clicking **Browse** to navigate to the installation directory.

Click **Next** to proceed.

- 3 **Review the summary information.**

The installation directory and size requirement are listed.

Click **Next** to proceed.

- 4 **Review the installation summary information. If the summary information panel indicates no errors, click Next to proceed.**

The Java Agent Setup Module is started. This begins the Java Agent configuration.

Configuring the Java Agent to Work as a TransactionVision Java Sensor on Windows

This section provides detailed instructions on how to configure the Java Agent to work as a TransactionVision Java Sensor using the Java Agent Setup Module user interface.

The Java Agent Setup Module starts automatically at the end of the Java Agent Installation. You can start the setup module at any time by choosing **Start > All Programs > HP Java Agent > Setup Module**.

Perform the following steps to configure the Java Agent to work with a TransactionVision Java Sensor:

- 1 **Select the TransactionVision Java Agent working with an HP TransactionVision Server option.**

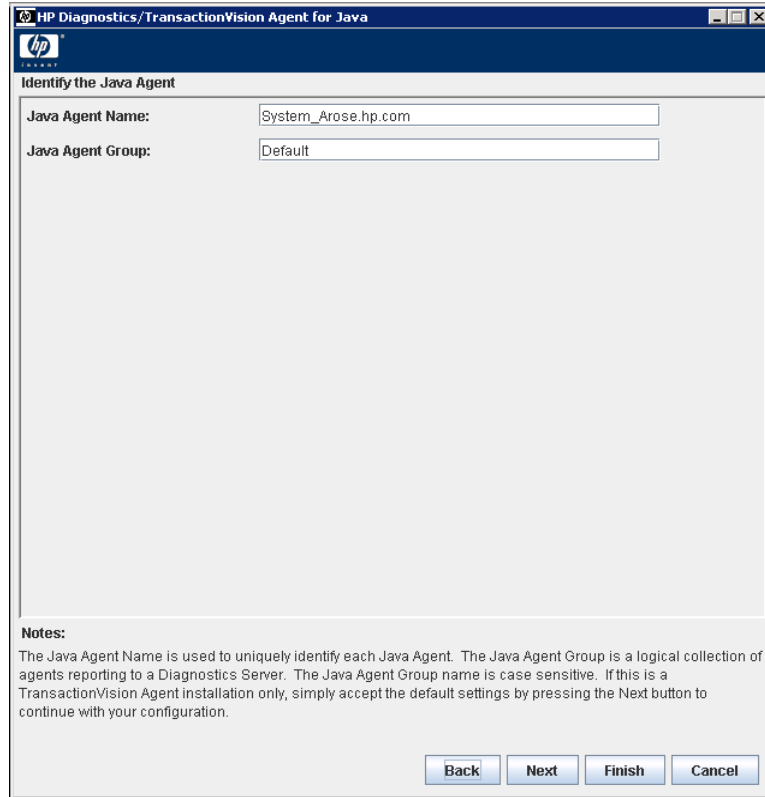


You can also choose from the following options:

- **Configure the Java Agent to work as a Diagnostics Profiler only.** If you are configuring the Java Agent as a Diagnostics Profiler only, see “Configuring the Java Agent as a Profiler Only” in the *HP Diagnostics Installation and Configuration Guide*.
- **Configure the Java Agent as a Diagnostics Java Agent working with an HP Diagnostics Server.** If you are configuring the Java Agent to work as a J2EE Probe with a Diagnostics Server, See “Configuring the Probe to Work with a Diagnostics Server” in the *HP Diagnostics Installation and Configuration Guide*.
- **Configure the Java Agent as both a Diagnostics Java Agent and a TransactionVision Java Agent.** If you are configuring the Java Agent to work as a J2EE Probe with a Diagnostics Server and also as a TransactionVision Java Sensor, select both check boxes and continue to step2. After this step, you first configure the Java Agent as the J2EE Probe (described in “Configuring the Java Agent to Work with a Diagnostics Server” in the *HP Diagnostics Installation and Configuration Guide*), then you configure the Java Agent as the J2EE TransactionVision Java Sensor starting with step 3.

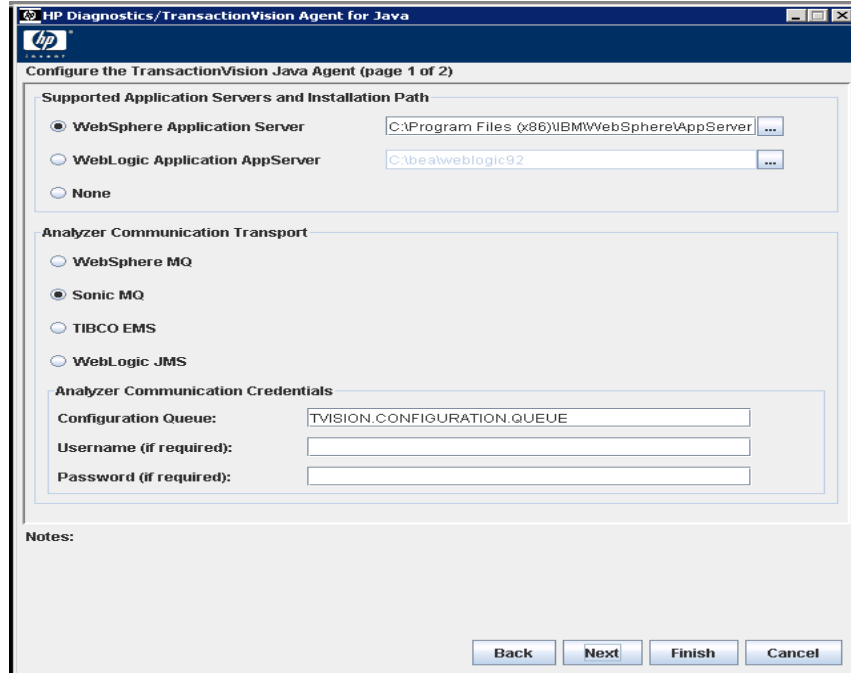
Click **Next** to proceed.

2 **Assign a name to the Java Agent and specify the group to which it belongs.**



- For the Java Agent name, enter a name that uniquely identifies the agent within TransactionVision. The following characters can be used in the name: - , _ , and all alphanumeric characters. The agent name is assigned to be the Java Sensor name.
When assigning a name to an agent, choose a name that will help you recognize the application that the agent is monitoring, and the type of Java Sensor.
- For the Java Agent group name, enter a name for an existing group or for a new group to be created. The agent group name is case-sensitive.
Click **Next** to proceed.

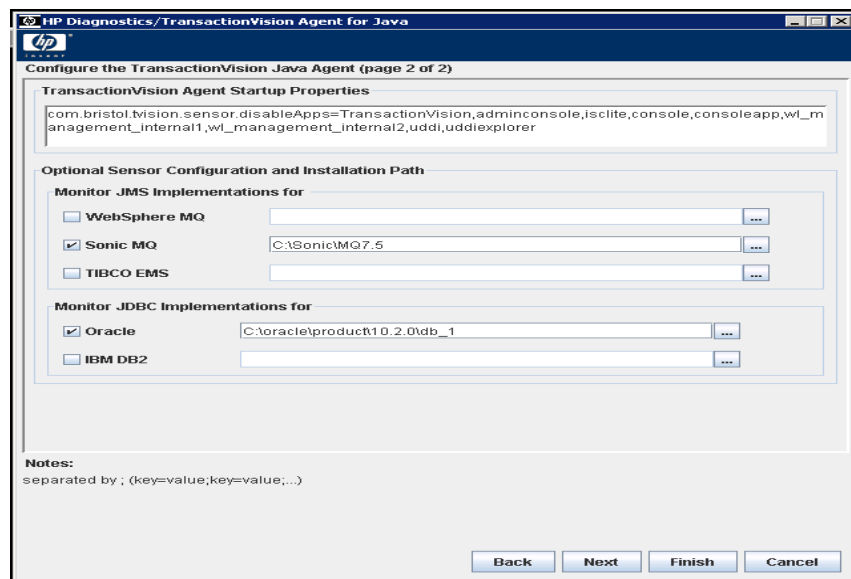
3 **Set the application server to be monitored and its installation directory.
Choose the JMS vendor to be used for the communication link transport.**



- You can right-click inside the text box to open a file selection dialog.
- Change the name of the configuration queue if you use a different queue.
- Enter the user name and password for the JMS provider if they are required.
- Click **Next** to continue.

4 Configure what JMS and JDBC implementation you want to monitor.

Enter the installation directory path of the corresponding implementation. You can right-click inside the text box to open the file selection dialog,



- ▶ BEA JMS for WebLogic 8.1.x is monitored automatically, and no additional configuration is necessary since it is integrated with WebLogic Application Server

5 Configure your JMS transport settings.

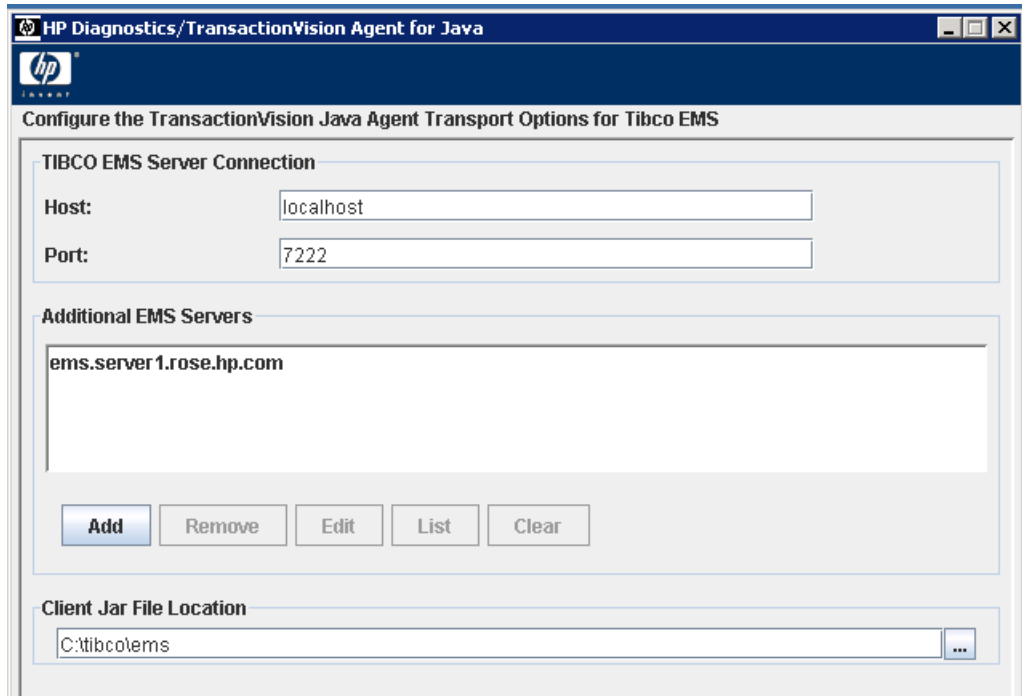
- If you choose WebSphere MQ as your communication link transport, configure the transport settings using WebSphere MQ JMS server binding or client connection.
- ▶ On 64-bit Windows with WebSphere MQ 6.0, if you want to monitor a 64-bit JVM, you must choose client instead of server. WebSphere MQ 6.0 does not support server binding on 64-bit Windows platforms.

The screenshot shows a configuration window titled "HP Diagnostics/TransactionVision Agent for Java" with the HP logo. The main heading is "Configure the TransactionVision Java Agent Transport Options for WebSphere MQ". The window is divided into four sections:

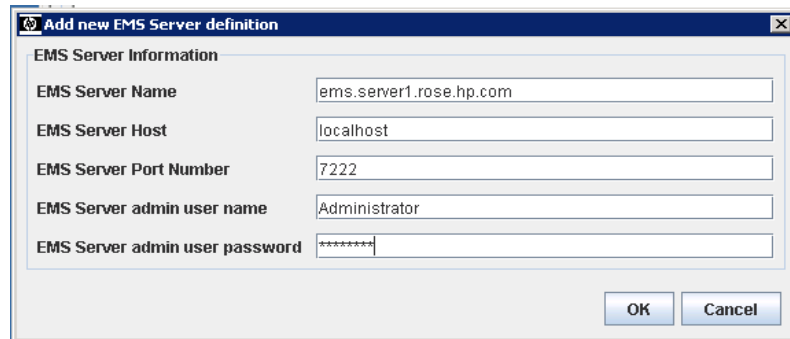
- Queue Connection:** Contains a text field for "Connection Queue Manager" with the value "TRADING" and a dropdown menu for "Connection Type" set to "client".
- Client Configuration:** Contains three text fields: "Queue Manager Host" with "localhost", "Queue Manager Port" with "1421", and "Queue Manager Channel" with "TRADING.CHL".
- Client Jar File Location:** Contains a text field with the path "C:\Program Files\IBM\WebSphere MQ" and a browse button (three dots).

- Enter the configuration queue manager name.
- If you choose the client connection, enter the client queue manager configuration information: host, port and channel.
- Enter or browse for the WebSphere MQ installation location.

- If you choose TIBCO EMS as your communication link transport, configure the transport settings for TIBCO EMS.

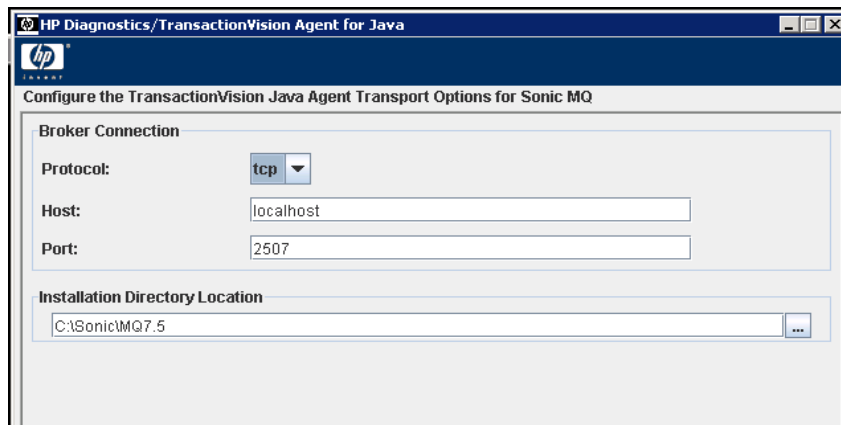


- Enter the host name. You can change the default port if you wish.
- Enter or browse for the TIBCO EMS installation location.
- You can add and define EMS servers. Click **Add** to open the add dialog.

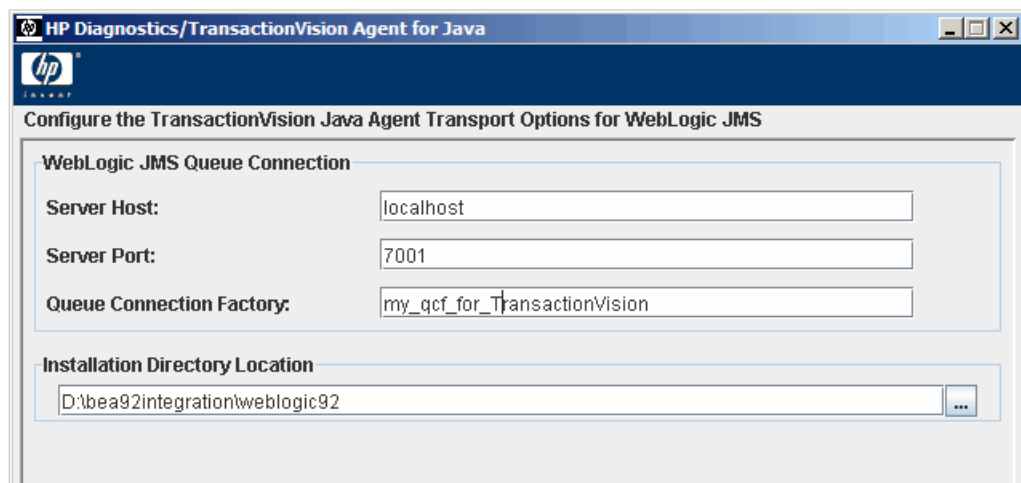


Enter the server definitions and click **OK**. To change the definitions, select the server and click **Edit**. To see what the definitions are for a selected server, click **List**.

- If you choose SonicMQ as your communication link transport, configure the transport settings for SonicMQ.



- Use the default tcp protocol unless a different protocol is used.
 - Enter the host name. You can change the default port if you wish.
 - Enter or browse for the SonicMQ installation location.
- If you choose WebLogic JMS as your communication link transport, configure the transport settings for WebLogic JMS.



- Enter the host name. You can change the default port if you wish.
- Enter the queue connection factory.
- Enter or browse for the WebLogic JMS installation location. It is typically the same as your WebLogic application server installation location.

You can go to any page to make changes any time you want.

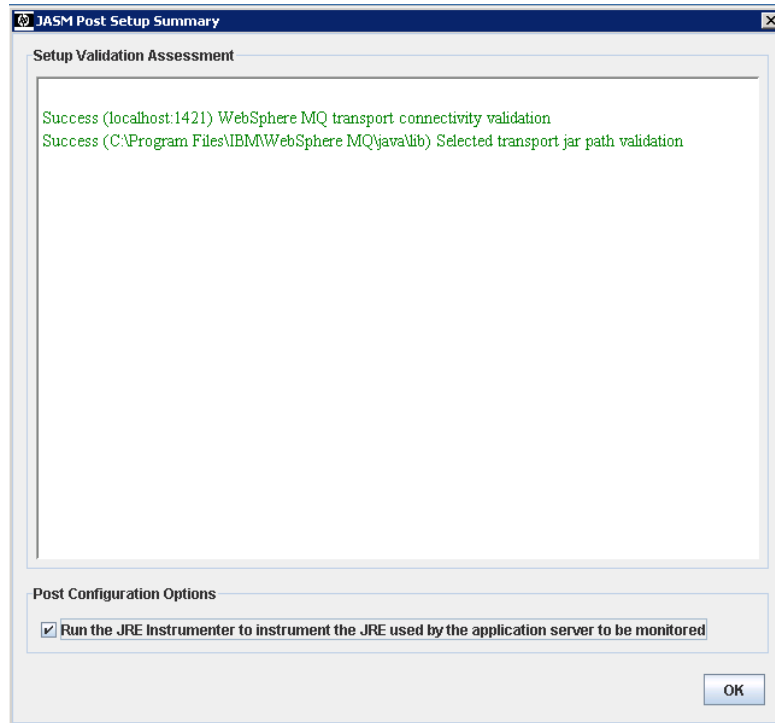
6 Save the configuration.

When you finish all the settings, click **Finish** to save the configuration. This action modifies both Diagnostics and TransactionVision configuration files.

Post Configuration Options

After modifying the configuration files, the Java Agent Setup Module automatically generates a master instrumentation file based on the version of various software installed on your system. This process takes several minutes. A wait dialog displays during the process.

At the end, the Java Agent Setup Module tests if the transport settings are valid.



If any validation fails, check your transport settings and make sure that your JMS server or queue manager is running with proper settings.

If validation is successful, you can choose to run the JRE Instrumenter automatically by checking **Run the JRE Instrumenter to instrument the JRE used by the application server to be monitored**. By default, the JRE Instrumenter option is not selected. If your JRE version is prior to 1.5, you must select this check box or run the JRE Instrumenter manually. For a complete description of the JRE Instrumenter and how to run it manually, see [Running the JRE Instrumenter](#) on page 33.

Enable Java Agent in Applications on Windows

For Java 1.5+

- To enable the Java Agent to monitor an application running on JRE 1.5 +, add the following JVM option to the java command line that starts the application:

```
java -javaagent:<java_agent_install_dir>\DiagnosticsAgent\lib\probeagent.jar
```

where <java_agent_install_dir> refers to the path of your Java Agent installation directory. The default path is C:\MercuryDiagnostics\JavaAgent.
- To enable Java Agent for application servers, see [Configuring the Application Servers](#) on page 40.

For Java 1.4

For any applications or application servers running with JRE version 1.4 (such as WebSphere 5.1, 6.0 or WebLogic 8.1), you need to run Java Agent's JRE Instrumenter tool to instrument the JRE that your application or application server is using. See [Running the JRE Instrumenter](#) on page 33 for complete details.

Enable Java Agent in Applications on UNIX

For Java 1.5+

- To enable the Java Agent to monitor an application running on JRE 1.5 +, add the following JVM option to the java command line that starts the application:

```
java -javaagent:<java_agent_install_dir>/DiagnosticsAgent/lib/  
probeagent.jar
```

where <java_agent_install_dir> refers to the path of your Java Agent installation directory. The default path is /opt/MercuryDiagnostics/JavaAgent.

- To enable Java Agent for application servers, see [Configuring the Application Servers](#) on page 40.

For Java 1.4

For any applications or application servers running with JRE version 1.4 (such as WebSphere 5.1, 6.0 or WebLogic 8.1), you need to run Java Agent's JRE Instrumenter tool to instrument the JRE that your application or application server is using. See [Running the JRE Instrumenter](#) on page 33 for complete details.

Regarding WebSphere MQ

If you use WebSphere MQ as your communication transport and you choose the connection type as server (default) in the Java Agent setup, you also need to add the path to WebSphere MQ java/lib to your system's library path environment variable. For example:

On AIX, add:

```
set LIBPATH=$LIBPATH:/usr/mqm/java/lib  
export LIBPATH
```

On Solaris or Linux, add:

```
set LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/mqm/java/lib  
export LD_LIBRARY_PATH
```

Replace lib with lib64 if your JVM is 64-bit.

On Windows, you typically do not need such settings because this path has been added to your PATH environment variable when you installed WebSphere MQ.

Installing and Configuring the Java Agent on UNIX

Java Agent installers have been provided for several UNIX platforms. The following instructions provide you with the steps necessary to install the Java Agent in most UNIX environments using either a graphical mode installation or a console mode installation.

The instructions and screen shots that follow are for an agent installation on an AIX machine. These same instructions should apply for the other certified UNIX platforms.

If there is a pre-existing installation of the Java Agent, the legacy J2EE Probe, or the legacy TransactionVision 5.0 Sensors on the host machine, you must uninstall it before you install the Java Agent.

Downloading the Installer on UNIX

You may download the Java Agent installer from the HP Software web site, from the Diagnostics or TransactionVision product disk, or from the Downloads page in Business Availability Center.

You must be the root user to install the Java Agent.

To copy the installer from the product installation disk:

- 1 From the <HP TransactionVision Installation Disk>/TransactionVision_Installers directory, copy the installer JavaAgentSetup<platform>_7_50.bin to the machine where the TransactionVision Server is to be installed.
- 2 Continue with [Running the Installation on UNIX](#) on page 26.

To download the installer from the Downloads page (for Business Availability Center users):

- 1 Select **Admin > Platform** from the top menu in Business Availability Center, and click the **Setup Maintenance** tab.
- 2 On the **Downloads** page, click the link to the installer that is appropriate for your environment and save it to the machine where the agent is to be installed.

Running the Installation on UNIX

After you have copied the installer to the machine where the Java Agent is to be installed, you are ready to run the installation.

To install the Java Agent on a UNIX machine:

- 1 Run the installer.

Where necessary, change the mode of the installer file to make it executable.

- Ensure that you are logged in as a root user.
- To run the installer in console mode, enter the following at the UNIX command prompt:

```
./JavaAgentSetup_<platform>_7_50.bin -console
```

The installer displays the installation prompts in console mode as shown in the steps that follow.

- To run the installer in the graphical mode enter the following at the UNIX command prompt:

```
xhost + #allows you to display the UI on the console
export DISPLAY=<hostname>:0.0
./JavaAgentSetup_<platform>_7_50.bin
```

The installer displays the same screens that are displayed for the Windows installer, as shown in [Installing and Configuring the Java Agent on Windows](#) on page 16.

2 Accept the end user license agreement.

The end user software license agreement is displayed.

Read the agreement. As you read, you can press **Enter** to move to the next page of text, or type **q** to jump to the end of the license agreement.

Accept the terms of the agreement by typing the number **1** and pressing **Enter**.

Type **0** (zero) and press **Enter**, then type the number **1** and press **Enter** to continue with the installation.

3 Specify the location where you want to install the agent.

At the **Installation Directory Name** prompt, accept the default installation location shown in brackets, or enter the path to a different location.

Type the number **1** and press **Enter** to continue with the installation.

4 Verify the installation location.

The installation location and the estimated size are listed.

If these are acceptable, type the number **1** and press **Enter** to start the installation.

The installation may take a few minutes.

The Java Agent Setup Module is launched.

Configuring the Java Agent to Work as a TransactionVision Java Sensor on UNIX

The following instructions provide you with the steps necessary to configure the Java Agent as a TransactionVision Java Sensor in most UNIX environments using the Java Agent Setup Module in either a graphical mode or a console mode.

The Java Agent Setup Module starts automatically at the end of the Installation program. You can start the Java Agent Setup Module at any time by running:

```
<java_agent_install_dir>/DiagnosticsAgent/bin/setupModule.sh
```

<java_agent_install_dir> refers to the path of your Java Agent installation directory. The default path is /opt/MercuryDiagnostics/JavaAgent.

Configuring the Java Agent on UNIX in Graphical Mode

To configure the Java Agent with a Java Sensor in graphical mode:

1 Set up the graphical interface.

Export your display back to your terminal.

```
xhost + #allows you to display the UI on the console
```

```
export DISPLAY=<hostname>:0.0
```

2 Run the Java Agent Setup Module.

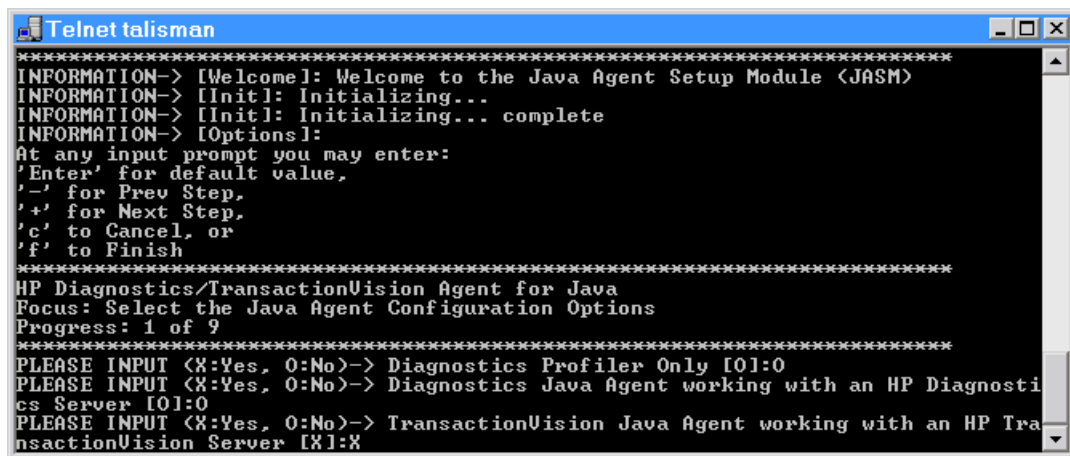
```
<java_agent_install_dir>/DiagnosticsAgent/bin/setupModule.sh
```

The Java Agent Setup Module displays the same screens that are displayed for the Windows Java Agent Setup Module, as shown in [Configuring the Java Agent to Work as a TransactionVision Java Sensor on Windows](#) on page 18.

Configuring the Java Agent on UNIX in Console Mode

To configure the Java Agent with a Java Sensor in console mode:

1 Select the TransactionVision Java Agent working with an HP TransactionVision Server option by entering X for this option.

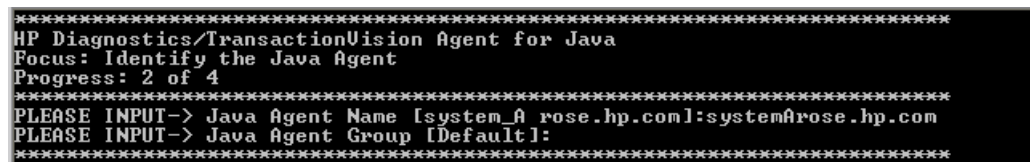


```
Telnet talisman
*****
INFORMATION-> [Welcome]: Welcome to the Java Agent Setup Module (JASM)
INFORMATION-> [Init]: Initializing...
INFORMATION-> [Init]: Initializing... complete
INFORMATION-> [Options]:
At any input prompt you may enter:
'Enter' for default value,
'-' for Prev Step,
'+' for Next Step,
'c' to Cancel, or
'f' to Finish
*****
HP Diagnostics/TransactionVision Agent for Java
Focus: Select the Java Agent Configuration Options
Progress: 1 of 9
*****
PLEASE INPUT (X:Yes, 0:No)-> Diagnostics Profiler Only [O]:0
PLEASE INPUT (X:Yes, 0:No)-> Diagnostics Java Agent working with an HP Diagnosti
cs Server [O]:0
PLEASE INPUT (X:Yes, 0:No)-> TransactionVision Java Agent working with an HP Tra
nsactionVision Server [X]:X
*****
```

- Enter O (capital letter O) to skip the Diagnostics Profile Only option and again to skip the Diagnostics Java Agent working with an HP Diagnostics Server option.
- If you want to configure the Java Agent to work as both a J2EE Probe with a Diagnostics Server and as a TransactionVision Java Sensor, enter O for both options and continue to step2.

Press **Enter** to continue.

2 Assign a name to the Java Agent and specify the group to which it belongs.



```
*****
HP Diagnostics/TransactionVision Agent for Java
Focus: Identify the Java Agent
Progress: 2 of 4
*****
PLEASE INPUT-> Java Agent Name [system_rose.hp.com]:system_rose.hp.com
PLEASE INPUT-> Java Agent Group [Default]:
*****
```

- For the Java Agent name, enter a name that uniquely identifies the agent within TransactionVision. The following characters can be used in the name: -, _, and all alphanumeric characters. The agent name is assigned to be the Java Sensor name. When assigning a name to an agent, choose a name that will help you recognize the application that the agent is monitoring, and the type of Java Sensor.

- For the Java Agent group name, enter a name for an existing group or for a new group to be created. The agent group name is case-sensitive.

Press **Enter** to continue.

3 Set the application server to be monitored and enter its installation directory. Choose the JMS vendor to be used for the communication link transport.

S

```
Focus: Configure the TransactionVision Java Agent (page 1 of 2)
Progress: 3 of 4
*****
PLEASE MAKE SELECTION-> Please select Application Server:
Selection 1. WebSphere
Selection 2. WebLogic
Selection 3. None
SELECT-> Please type in corresponding Number or
SELECT->      press Enter for default [WebSphere]:
1
PLEASE INPUT-> Please installation type in path for AppServer WebSphere [/usr/We
bSphere60/AppServer1:/usr/WebSphere60/AppServer
PLEASE MAKE SELECTION-> Please select Analyzer Communication Transport:
Selection 1. WebSphere MQ
Selection 2. Sonic MQ
Selection 3. TIBCO EMS
Selection 4. WebLogic JMS
SELECT-> Please type in corresponding Number or
SELECT->      press Enter for default [Sonic MQ]:
1
PLEASE INPUT-> Configuration Queue [TVISION.CONFIGURATION.QUEUE]:
PLEASE INPUT-> Username (if required) []:
PLEASE INPUT-> Password (if required) []:*
*****
```

- Change the name of the configuration queue if you use a different queue.
- Enter the user name and password for the JMS provider if they are required.
- Press **Enter** to continue.

4 Configure what JMS and JDBC implementation you want to monitor.

```
*****
HP Diagnostics/TransactionVision Agent for Java
Focus: Configure the TransactionVision Java Agent (page 2 of 2)
Progress: 4 of 5
*****
PLEASE INPUT-> Additional TransactionVision Properties: [com.bristol.tvision.sen
sor.disableApps=TransactionVision.adminconsole,isc-lite_console,consoleapp,wl_man
agement_internal1,wl_management_internal2,uddi,uddiexplorer]:
PLEASE INPUT (X:Yes, O:No)-> Monitor JMS Implementations for WebSphere MQ [O]:X
PLEASE INPUT-> Installation path for JMS transport WebSphere MQ [/usr/mqm:/usr/
mqm
PLEASE INPUT (X:Yes, O:No)-> Monitor JMS Implementations for Sonic MQ [O]:O
PLEASE INPUT (X:Yes, O:No)-> Monitor JMS Implementations for TIBCO EMS [O]:O
PLEASE INPUT (X:Yes, O:No)-> Monitor JDBC Implementations for Oracle [O]:O
PLEASE INPUT (X:Yes, O:No)-> Monitor JDBC Implementations for IBM DB2 [O]:X
PLEASE INPUT-> Installation path for JDBC transport IBM DB2 [/usr/opt/db2_08_01]
:/usr/opt/db2_08_01
*****
```



Use the TransactionVision Startup Properties field to define additional properties for TransactionVision Java Sensor. The `com.bristol.tvision.sensor.disableApps` property disables the Sensor for any application name listed. Delimit multiple applications with a comma. The default value is shown in the screen shot.

- Press **Enter** at the Additional TransactionVision Properties prompt.
- Select one or more JMS properties by entering capital X for yes or capital O for no next to each transport (if you wish, you can enter no for all properties).
- If you specify yes for any transport, you need to enter the installation path for that transport.

- Select the JDBC database if you wish.
- ▶ BEA JMS for WebLogic 8.1.x is monitored automatically, and no additional configuration is necessary since it is integrated with WebLogic Application Server

5 Configure your JMS transport settings.

- If you choose WebSphere MQ as your communication link transport, configure the transport settings using WebSphere MQ JMS server binding or client connection.

```

*****
HP Diagnostics/TransactionVision Agent for Java
Focus: Configure the TransactionVision Java Agent Transport Options for WebSphere MQ
Progress: 5 of 5
*****
PLEASE INPUT-> Connection Queue Manager [configuration_qm_name]:TRADING
PLEASE MAKE SELECTION-> Please select Connection Type:
Selection 1. server
Selection 2. client
SELECT-> Please type in corresponding Number or
SELECT->      press Enter for default [server]:
2
PLEASE INPUT-> Host []:localhost
PLEASE INPUT-> Port []:1421
PLEASE INPUT-> Channel []:TRADING.CHL
PLEASE INPUT-> Install Path [/usr/mqm]:/usr/mqm
PLEASE INPUT->
CONFIRM-> Would you like to save your changes now? Enter Y or N: [Y]:N
*****

```

- Enter the configuration queue manager name and press **Enter**.
- If you choose the client connection, enter the client queue manager configuration information: host, port and channel.
- Enter the WebSphere MQ installation location and press **Enter**.
- If you choose SonicMQ as your communication link transport, configure the transport settings for SonicMQ.

```

*****
HP Diagnostics/TransactionVision Agent for Java
Focus: Configure the TransactionVision Java Agent Transport Options for Sonic MQ
Progress: 5 of 5
*****
PLEASE MAKE SELECTION-> Please select Broker Protocol Type:
Selection 1. tcp
SELECT-> Please type in corresponding Number or
SELECT->      press Enter for default [tcp]:
1
PLEASE INPUT-> Host [localhost]:
PLEASE INPUT-> Port [2506]:
PLEASE INPUT-> Install Path [/opt/Sonic75/MQ7.5]:/opt/Sonic75/MQ7.5
PLEASE INPUT->
CONFIRM-> Would you like to save your changes now? Enter Y or N: [Y]:

```

- Enter the name of the protocol and press **Enter**.
- Enter the host name and press **Enter**.
- Press **Enter** to choose the default port, or enter a different one and press **Enter**.
- Enter the SonicMQ installation location and press **Enter**.
- If you choose TIBCO EMS as your communication link transport, configure the transport settings for TIBCO EMS.

```

*****
HP Diagnostics/TransactionVision Agent for Java
Focus: Configure the TransactionVision Java Agent Transport Options for Tibco EMS
Progress: 5 of 5
*****
PLEASE INPUT-> Host []:localhost
PLEASE INPUT-> Port [7222]:
PLEASE INPUT-> Install Path [!:/opt/tibco/ems
INFORMATION-> [Existing TIBCO EMS Server Definitions]:
PLEASE INPUT-> Option? (A:Add, Rn: Remove, En: Edit, or 0: Exit) [0]:A
PLEASE INPUT-> EMS Server Name [!:ems_server1.rose.hp.com
PLEASE INPUT-> EMS Server Host [!:localhost
PLEASE INPUT-> EMS Server Port Number [!:7222
PLEASE INPUT-> EMS Server admin user name [!:Administrator
PLEASE INPUT-> EMS Server admin user password [!:*****
INFORMATION-> [Existing TIBCO EMS Server Definitions]:
1: ems_server1.rose.hp.com
   URL: tcp://ems_server1.rose.hp.com:7222/
   Username: Administrator
   Password: *****
PLEASE INPUT-> Option? (A:Add, Rn: Remove, En: Edit, or 0: Exit) [0]:0

```

- Enter the host name and press **Enter**.
- Press **Enter** to choose the default port, or enter a different one and press **Enter**.
- Enter the TIBCO EMS installation location.
- You can add and define EMS servers:
 - Type **A** and press **Enter**. Enter each server definition and press **Enter** after each one you enter.
- If you choose WebLogic JMS as your communication link transport, configure the transport settings for WebLogic JMS.

```

*****
HP Diagnostics/TransactionVision Agent for Java
Focus: Configure the TransactionVision Java Agent Transport Options for WebLogic JMS
Progress: 5 of 5
*****
PLEASE INPUT-> Server Host []:localhost
PLEASE INPUT-> Server Port [7001]:
PLEASE INPUT-> Queue Connection Factory [!:myQueFactory
PLEASE INPUT-> Install Path [!:/opt/hea/weblogic81

```

- Enter the host name and press **Enter**.
 - Press **Enter** to choose the default port, or enter a different one and press **Enter**.
 - Enter the queue connection factory and press **Enter**.
 - Enter the WebLogic JMS installation location and press **Enter**. It is typically the same as your WebLogic application server installation location.
- 6 **When prompted to save your changes to the Java Agent Setup Module, enter Y.**
 - 7 **Instrument the JRE.**


```

CONFIRM-> Would you like to save your changes now? Enter Y or N: [Y]-Y
INFORMATION-> [Save]: Saving Dialog Select the Java Agent Configuration Options
INFORMATION-> [Save]: Saving Dialog Identify the Java Agent
INFORMATION-> [Save]: Saving Dialog Configure the Diagnostics Java Agent
INFORMATION-> [Save]: Saving Dialog Configure the TransactionVision Java Agent <
page 1 of 2>
INFORMATION-> [Save]: Saving Dialog Configure the TransactionVision Java Agent <
page 2 of 2>
INFORMATION-> [Save]: Saving Dialog Configure the TransactionVision Java Agent T
ransport Options for WebSphere MQ
INFORMATION-> [Save]: Saving Dialog Configure the TransactionVision Java Agent T
ransport Options for Sonic MQ
INFORMATION-> [Save]: Saving Dialog Configure the TransactionVision Java Agent T
ransport Options for Tibco EMS
INFORMATION-> [Save]: Saving Dialog Configure the TransactionVision Java Agent T
ransport Options for WebLogic JMS
INFORMATION-> [Save]: Saving Diagnostics Agent property files
INFORMATION-> [Save]: Saving TransactionVision Agent property files
INFORMATION-> [Merge]: Merging TransactionVision Rules files...please wait
INFORMATION-> [JASM Post Setup Summary]:
Success <localhost:2506> Sonic MQ transport connectivity validation
Success </opt/Sonic75/MQ7.5/lib> Selected transport jar path validation
INFORMATION-> [Currently Instrumented UMs]:
IBM 1.5.0 </usr/java5/jre>
IBM 1.4.2 </usr/java14/jre>
IBMJ9 1.4.2 </usr/java14/jre/lib/jc1SC14>
IBM 1.5.0 </usr/java/jre>
PLEASE INPUT-> Option? (<'Command', H: Help, or 0: Exit) [0]:

```

Enter the instrumentation commands as described in [Running the JRE Instrumenter on a UNIX Machine](#) on page 37.

8 Enter 0 (zero) to complete the setup.

Once you have installed the agent, configured it as a Java Probe and instrumented the JRE, you need to perform post-installation tasks.

9 Modify the startup script for the application server so that the probe is started together with the monitored application.

For detailed instructions, see [Configuring the Application Servers](#) on page 40.

10 Verify the Java Agent installation.

Silent Installation of the Java Agent

A *silent installation* is an installation that is performed automatically, without the need for user interaction. In place of user input, the silent installation accepts input from a response file for each step of the installation.

For example, a system administrator who needs to deploy a component on multiple machines can create a response file that contains all the prerequisite configuration information, and then perform a silent installation on multiple machines. This eliminates the need to provide any manual input during the installation procedure.

Before you perform silent installations on multiple machines, you need to generate a response file that will provide input during the installation procedure. This response file can be used in all silent installations that require the same input during installation.

The silent installation uses two response files: one for the Java Agent installation and one for the Java Agent Setup module.

To generate a response file for the Java Agent installation:

Perform a regular installation with the following command-line option:

```
<installer> -options-record <installResponseFileName>
```


This creates a response file that includes all the information submitted during the installation.

To generate a response file for the Java Agent Setup program:

Run the Java Agent Setup program with the following command-line option.

- On Windows:

```
<java_agent_install_dir>\bin\setupModule.cmd -createBackups -console  
-recordFile <JASMRresponseFileName>
```

- On UNIX:

```
<java_agent_install_dir>/bin/setupModule.sh -createBackups -console  
-recordFile <JASMRresponseFileName>
```

Either command creates a response file that includes all the information submitted during the installation.

To perform a silent installation or configuration:

Perform a silent installation or configuration using the relevant response files.

You set an environment variable and use the `-silent` command-line option as follows.

```
set HP_JAVA_AGENT_SETUP=-DoNotRun  
<installer> -options <installResponseFileName> -silent
```

Followed by:

```
set HP_JAVA_AGENT_SETUP=  
cd <setupModule> -createBackups -console -installFile <JASMRresponseFileName>
```

Note that on UNIX systems you need quotes around "-DoNotRun."

Running the JRE Instrumenter

The JRE Instrumenter instruments the `ClassLoader` class for the JVM that the application is using and places the instrumented `ClassLoader` in a folder under the `<java_agent_install_dir>/DiagnosticsAgent/classes` directory. It also provides you with the JVM parameter that must be used when an application or application server is started so that the application server will use the instrumented class loader.

If the JDK (`java.exe` executable) used by the application server changes, you must run the JRE Instrumenter again so that the Java Agent can continue to monitor the processing.

NOTES:

- If you want to instrument IBM's 1.4.2 J9 JRE, you must instrument the correct `ClassLoader` and add the `-Xj9` option on the application's command line. The correct `ClassLoader` is located in the `<java_dir>\jre\lib\jclSC14` directory (for example, `jreinstrumenter.sh -i \usr\java14_64\jre\lib\jclSC14`).
- If the Java Agent is being used to monitor multiple JVMs, the JRE Instrumenter must be run once for each JVM so that the Java Agent can be prepared to instrument the applications that are running on each JVM. For details, see "Configuring the Probes for Multiple Application Server JVM Instances" in the *HP Diagnostics Installation and Configuration Guide*.

JRE Instrumenter Processing

The JRE Instrumenter performs the following functions:

- Identifies JVMs that are available to be instrumented.
- Searches for additional JVMs in directories that you specify.
- Instruments the JVMs that you specify and provides the parameter that you must add to the startup script for the JVM to point to the location of the instrumented ClassLoader class.

The Instrumenter puts the instrumented ClassLoader in different places depending on how it is executed.

- When the Instrumenter is run from the Java Agent installer, the Instrumenter places the instrumented ClassLoader in a folder under the `<java_agent_install_dir>/DiagnosticsAgent/classes/boot` directory.
- When the Instrumenter is run using the graphical interface in a Windows or UNIX environment, the Instrumenter places the instrumented ClassLoader in a folder under the `<java_agent_install_dir>/DiagnosticsAgent/classes/<JVM_vendor>/<JVM_version>` directory.
- When the Instrumenter is run in a UNIX environment in console mode, the Instrumenter places the instrumented ClassLoader in either a folder under the `<java_agent_install_dir>/DiagnosticsAgent/classes/boot` directory or the `<java_agent_install_dir>/DiagnosticsAgent/classes/<JVM_vendor>/<JVM_version>` directory depending on the processing option specified. For more information on the UNIX processing options see [Instrumenting a Listed JVM](#) on page 39.

Running the JRE Instrumenter Manually

Instructions for running the JRE Instrumenter in a Windows environment and in a UNIX environment in console mode are provided below.

This section includes:

- [Running the JRE Instrumenter on a Windows Machine](#) on page 34
- [Running the JRE Instrumenter on a UNIX Machine](#) on page 37

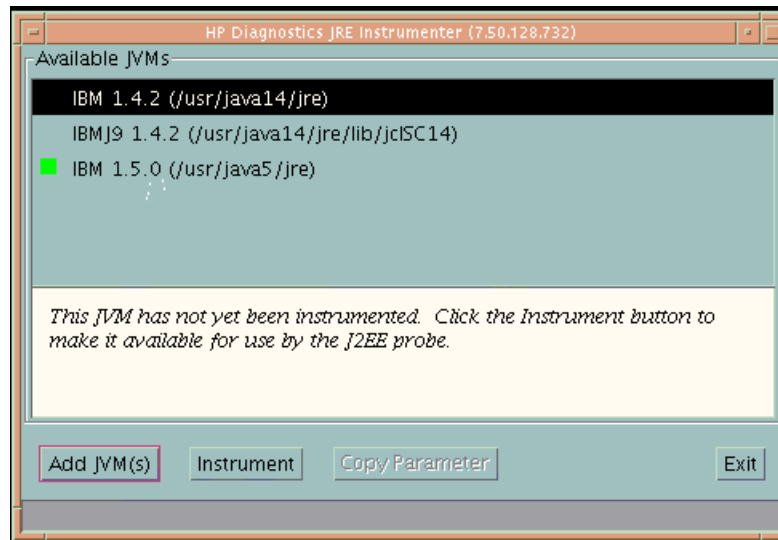
Running the JRE Instrumenter on a Windows Machine

When the JRE Instrumenter is run in a Windows environment, the Instrumenter displays the dialogs of its graphical user interface. The same dialogs are displayed when running the installer on a UNIX machine when the Instrumenter is running in graphical mode.

Starting the JRE Instrumenter on a Windows Machine

- 1 Go to `<java_agent_install_dir>\DiagnosticsAgent\bin` to locate the JRE Instrumenter executable.
- 2 Run the command:
`jreinstrumenter.cmd`

When the Instrumenter starts, it displays the JRE Instrumentation Tool dialog.



The Instrumenter lists the JVMs that were discovered by the Instrumenter and are available for instrumentation. The JVMs that have already been instrumented are listed with a green square preceding the name of the JVM.

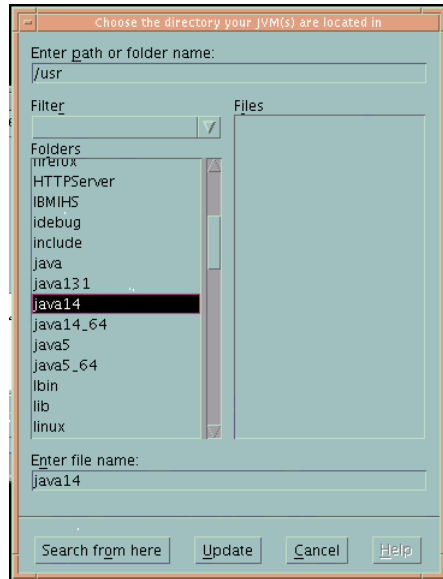
From the JRE Instrumentation Tool dialog you can perform the following tasks:

- If the JVM that you want to instrument is not listed in the Available JVMs list in the dialog, you can add JVMs to the list as described in [Adding JVMs to the Available JVMs List](#) on page 35.
- If the JVM that you want to instrument is listed but has not yet been instrumented, you can instrument the JVM as described in [Instrumenting a Selected JVM](#) on page 37.
- If the JVM that you want to instrument is listed and has already be instrumented, you can copy the JVM parameter that must be inserted into the start script for the JVM to activate the Probe's monitoring as described in [Including the JVM Parameter in the Application Server's Startup Script](#) on page 37.
- If you have finished using the JRE Instrumenter you can click Exit to close the JRE Instrumentation Tool.

Adding JVMs to the Available JVMs List

- 1 In the JRE Instrumentation Tool dialog, click **Add JVM(s)** to search for other JVMs and add them to the Available JVMs list.

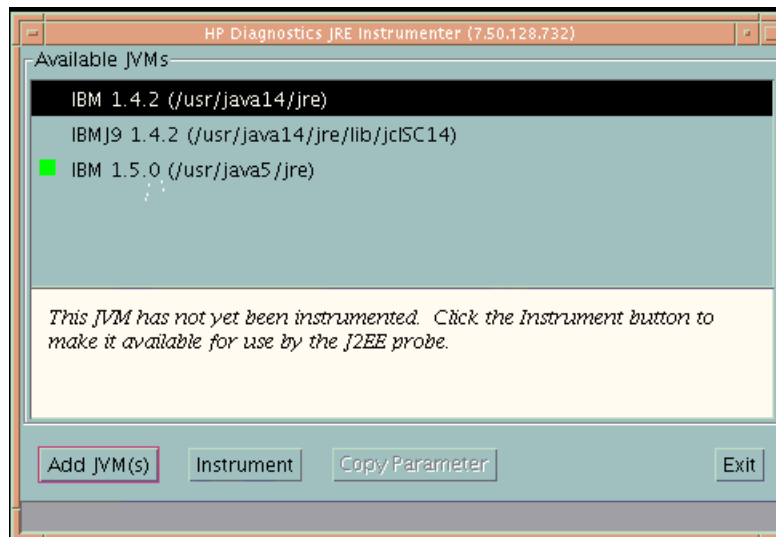
The Instrumenter displays the Choose the Directory dialog box.



- 2 Enter the directory location where you would like the Instrumenter to begin searching for JVMs.
- 3 Click **Update** to list all the folders in this directory in the **Folders** list.
- 4 Select the folder where you would like to begin the search so that its name appears in the **File name** box.
- 5 Click **Search from here** to start searching for JVMs.

The Instrumenter closes the dialog box and displays the JRE Instrumentation tool dialog box once more. The command buttons on the dialog are disabled while the Instrumenter searches for JVMs. A progress bar at the bottom of the dialog indicates that the Instrumenter is searching and shows how far along it is in the search process.

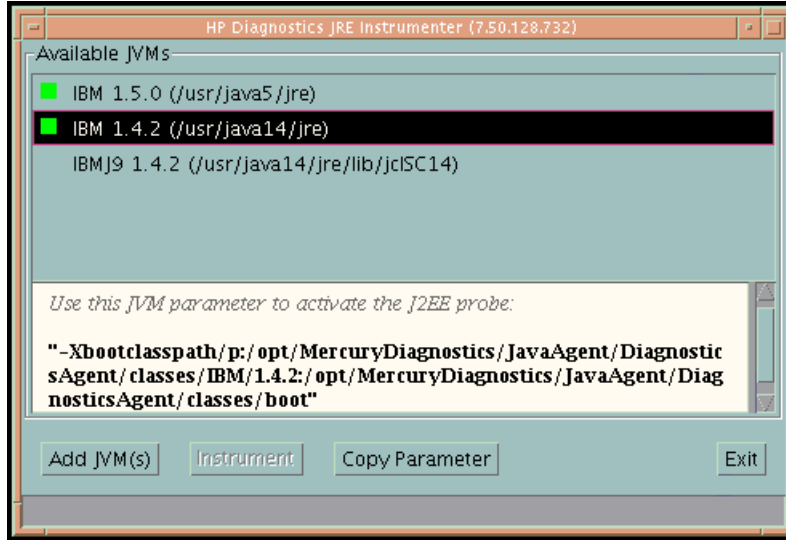
As the tool locates JVMs, it lists them in the **Available JVMs** list.



When the Instrumenter has completed the search, it enables the command buttons on the dialog. If the selected row is a JVM that has already been instrumented, the Instrument button is disabled. The green square indicates that the JVM has been instrumented.

Instrumenting a Selected JVM

Select a JVM that has not been instrumented from the **Available JVMs** list and click **Instrument**.



The JRE Instrumenter instruments the ClassLoader class for the selected JVM and places the instrumented ClassLoader in a folder under the <java_agent_install_dir>/DiagnosticsAgent/classes directory. It also displays the JVM parameter in the box below the Available JVMs list, which must be used when the application server is started.

Including the JVM Parameter in the Application Server's Startup Script

When the JRE Instrumenter instruments a JVM it also creates the JVM parameter that you must include in the startup script for the application server in order to cause your application to use the instrumented class loader. When you select an instrumented JVM from the Available JVMs list the JVM parameter is displayed in the box below the list. You can copy and paste one or more instrumented JVMs (one at a time) to the appropriate location for the application server to pick up.

To copy the JVM parameter displayed in this box to the clipboard, click **Copy Parameter**. The JVM parameter is copied to the clipboard so that you can paste it into the location that allows it to be picked up when your application server starts.



When all the JVM parameters have been copied and pasted to the appropriate application server location, be sure to stop and restart the application server for the settings to take affect. If copy does not work, manually type them in. WebLogic JVM is usually located at the %bea_home% directory. WebSphere JVM is usually in the %WebSphere_home%\java directory.

Running the JRE Instrumenter on a UNIX Machine

The following instructions provide you with the steps necessary to run the JRE Instrumenter using either a graphical mode installation or a console mode installation.

The JRE Instrumenter screens that are displayed in a graphical mode are the same as those documented for Windows installer in [Running the JRE Instrumenter on a Windows Machine](#) on page 34.

- ▶ If you are installing this agent to work with WebSphere in an Integrated Development Environment (IDE), you must run the JRE Instrumenter using a slightly different procedure than is described here. See [Running the JRE Instrumenter for WebSphere IDE on page 45](#).

Starting the JRE Instrumenter on a UNIX Machine

Open `<java_agent_install_dir>/DiagnosticsAgent/bin` to locate the JRE Instrumenter executable. Run the following command:

```
./jreinstrumenter.sh -console
```

When the Instrumenter starts, it displays a list of the processing options that are available as shown in the following table:

Instrumenter Function	Description
<code>jreinstrumenter -l</code>	Display the list of known JVMs. See Displaying the List of Instrumented JVMs on page 38 for details.
<code>jreinstrumenter -a DIR</code>	Look for JVMs below the DIR directory. See Adding JVMs to the Available JVMs List on page 38 for details.
<code>jreinstrumenter -i JVM_DIR</code>	Instrument the JVM in JVM_DIR. See Instrumenting a Listed JVM on page 39 for details.
<code>jreinstrumenter -b JVM_DIR</code>	Instrument the JVM in JVM_DIR and put the ClassLoader in <code><java_agent_install_dir>/DiagnosticsAgent/classes/boot</code> . See Instrumenting a Listed JVM on page 39 for details.

You can redisplay the list of options by specifying the `-x` option when you run the `jreinstrumenter.sh` command:

```
./jreinstrumenter.sh -x
```

Displaying the List of Instrumented JVMs

To display a list of the JVMs that are known to the JRE Instrumenter enter the following command:

```
./jreinstrumenter.sh -l
```

The Instrumenter lists the JVMs that it is aware of in rows containing the JVM vendor, JVM version, and the location where the JVM is located.

Adding JVMs to the Available JVMs List

To search for JVMs within a specific directory and add any JVMs that are found to the list of the JVMs that are known to the JRE Instrumenter enter the following command:

```
./jreinstrumenter.sh -a DIR
```

Replace DIR with the path to the location where you would like the Instrumenter to begin searching.

The Instrumenter searches the directories from the location specified including the directories and subdirectories. When it has completed its search, it displays the updated list of Available JVMs.

Instrumenting a Listed JVM

To instrument a JVM listed in the Available JVMs list, use one of the following two commands:

- Explicit path to ClassLoader

```
./jreinstrumenter.sh -i JVM_DIR
```

Replace JVM_DIR with the path to the location of the JVM as specified in the Available JVM list.

This command instructs the JRE Instrumenter to instrument the ClassLoader class for the selected JVM and places the instrumented ClassLoader in a folder under the <java_agent_install_dir>/DiagnosticsAgent/classes/<JVM_vendor>/<JVM_version> directory.

This is the command that you should use; especially if you want to instrument multiple JVM to be monitored by a single Java Agent.

- Generic path to ClassLoader

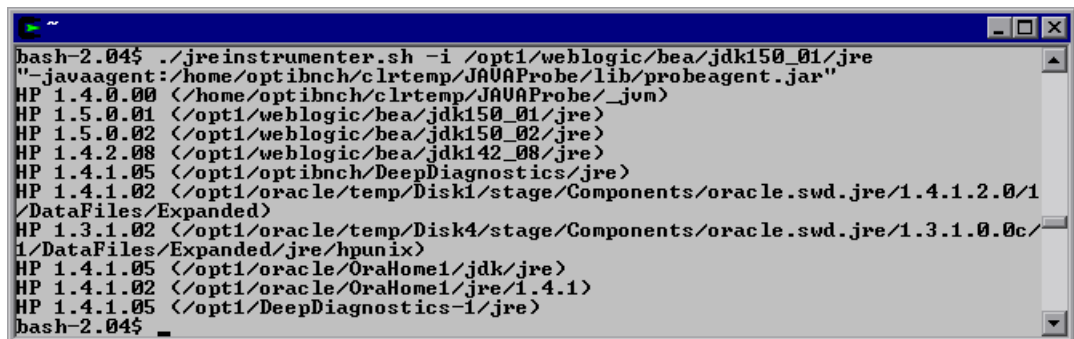
```
./jreinstrumenter.sh -b JVM_DIR
```

Replace JVM_DIR with the path to the location of the JVM as specified in the Available JVM list.

This command instructs the JRE Instrumenter to instrument the ClassLoader class for the selected JVM and places the instrumented ClassLoader in a folder under the <java_agent_install_dir>/DiagnosticsAgent/classes/boot directory.

You should only use this command if you are monitoring a single JVM with the Java Agent and there is some reason that you do not want to use the more explicit path generated when you use the -i command option.

When the Instrumenter has finished instrumenting the JVM, it displays the JVM parameter that must be used to activate the instrumentation and enable the Java Agent to monitor your application. Following the JVM parameter, the Instrumenter lists the Available JVM list again as shown in the following example:



```
bash-2.04$ ./jreinstrumenter.sh -i /opt1/weblogic/boa/jdk150_01/jre
"-javaagent:/home/optibnch/clrtemp/JAUAProbe/lib/probeagent.jar"
HP 1.4.0.00 </home/optibnch/clrtemp/JAUAProbe/_jvm>
HP 1.5.0.01 </opt1/weblogic/boa/jdk150_01/jre>
HP 1.5.0.02 </opt1/weblogic/boa/jdk150_02/jre>
HP 1.4.2.08 </opt1/weblogic/boa/jdk142_08/jre>
HP 1.4.1.05 </opt1/optibnch/DeepDiagnostics/jre>
HP 1.4.1.02 </opt1/oracle/temp/Disk1/stage/Components/oracle.swd.jre/1.4.1.2.0/1
/DataFiles/Expanded>
HP 1.3.1.02 </opt1/oracle/temp/Disk4/stage/Components/oracle.swd.jre/1.3.1.0.0c/
1/DataFiles/Expanded/jre/hpunix>
HP 1.4.1.05 </opt1/oracle/OraHome1/jdk/jre>
HP 1.4.1.02 </opt1/oracle/OraHome1/jre/1.4.1>
HP 1.4.1.05 </opt1/DeepDiagnostics-1/jre>
bash-2.04$
```

Including the JVM Parameter in the Application Server's Startup Script

When the JRE Instrumenter instruments a JVM it also creates the JVM parameter that you must include in the startup script for the application server in order to cause your application to use the instrumented class loader. When the Instrumenter has finished instrumenting the JVM, it displays the JVM parameter.

Copy the JVM parameter to the clipboard and paste it into the location that allows it to be picked up when your application server starts.

Configuring the Application Servers

Important: If you have the legacy TransactionVision 5.0 Sensors installed on your application server, you must remove them from the application server. You should uninstall the TransactionVision 5.0 Sensors from the host machine.

This section provides instructions on how to configure the application server to allow the Java Agent to monitor the application. This section includes:

- [About Configuring the Application Server](#)
- [Configuring WebSphere Application Servers](#)
- [Configuring WebLogic Application Servers](#)

About Configuring the Application Server

Once you have executed the JRE Instrumenter for the Java Agent, you must modify the startup script for the application so that the Java Agent that is to monitor the application will be started when the application is started.

You can configure the application servers by updating the application server startup scripts manually. The following sections provide instructions for updating the application servers manually.



Example procedures are shown for a given version of the application server. For details on what application server versions are supported on what platforms, contact HP customer support.

It is possible that your site administrator has site-specific methods for making configuration modifications. In this case, the generic procedure described in “Configuring a Generic Application Server” on page 211 should provide the information that the site administrator needs to implement the required configuration changes.

The process for configuring the Java Agent and the application servers when there are multiple JVMs on a single machine is described in For details, see “Configuring the Probes for Multiple Application Server JVM Instances” in the *HP Diagnostics Installation and Configuration Guide*.

Notes:

In this section, <java_agent_install_dir> is used to indicate the directory where the Java Agent was installed.

When modifying the -Xbootclasspath or -javaagent parameter, be sure to use quotes if there are spaces in the path that you specify.

Configuring WebSphere Application Servers

WebSphere servers are controlled using the WebSphere Application Server Administrative Console. The Console has control over the JVM command line and allows you to add classpath elements, define runtime variables (-D variables), and configure the bootclasspath or javaagent for WebSphere. You use the Administrative Console for most of the configurations.

The appearance of the Console may differ for different versions of WebSphere. The way that changes are implemented in each version of WebSphere may vary slightly. As a result, the examples shown in this section may not correspond exactly to your WebSphere version. The examples should provide the information that you need to enter the required parameters in the appropriate location in the Console.



WebSphere applies the changes that are made on each tab only when you click **Apply** on the tab. Your changes will not be applied until you click **Apply**.

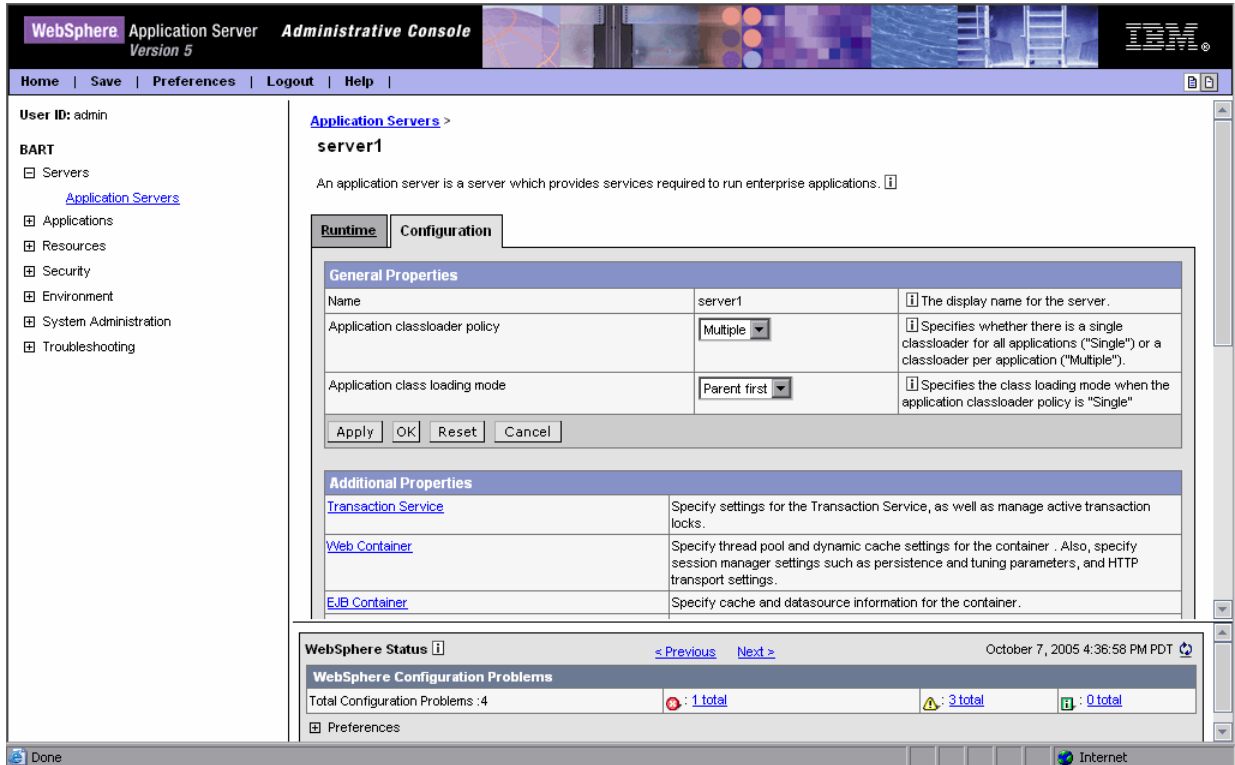
This section includes the following topics:

- [WebSphere 5.x and 6.0](#)
- [WebSphere 6.1](#)
- [Running the JRE Instrumenter for WebSphere IDE](#)
- [Using the JMS Sensor with the WebSphere Application Server](#)
- [Adding Interceptors for Sensors](#)

WebSphere 5.x and 6.0

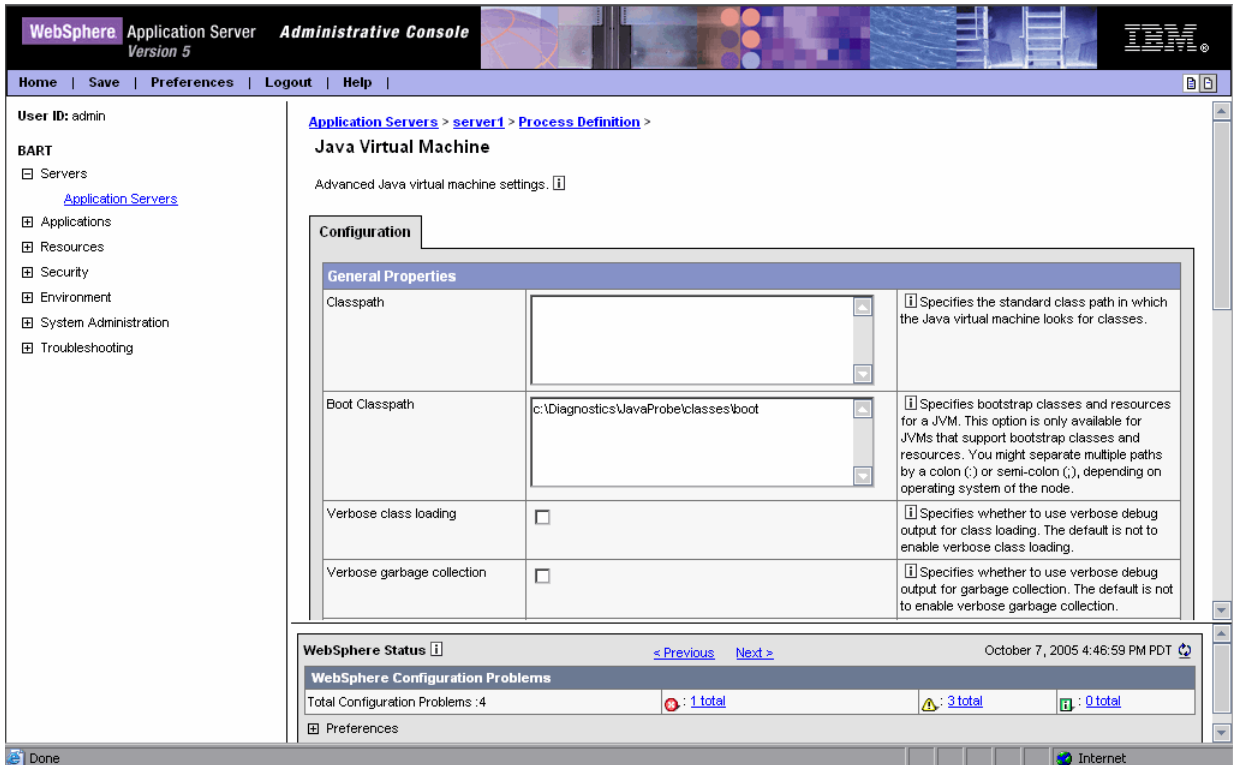
To configure a WebSphere 5.x and 6.0 application server:

- 1 Use your Web browser to access the WebSphere Application Server Administrative Console for the application server instance for which the Java Agent was installed:
http://<App_Server_Host>:9090/admin
Replace **<App_Server_Host>** with the machine name for the application server host.
The WebSphere Application Server Administrative Console opens.
- 2 In the left panel, select **Servers > Application Servers**.
- 3 From the list of application servers in the right panel, select the name of the server that you want to configure so that it will be monitored by the Java Agent.
The Configuration tab for the selected application server is displayed.



- 4 Scroll down in the Configuration tab, and in the General Properties column, look for the **Process Definition** property.
- 5 Click **Process Definition**.
- 6 Scroll down in the right panel, and look for **Java Virtual Machine**.
- 7 Click **Java Virtual Machine**.

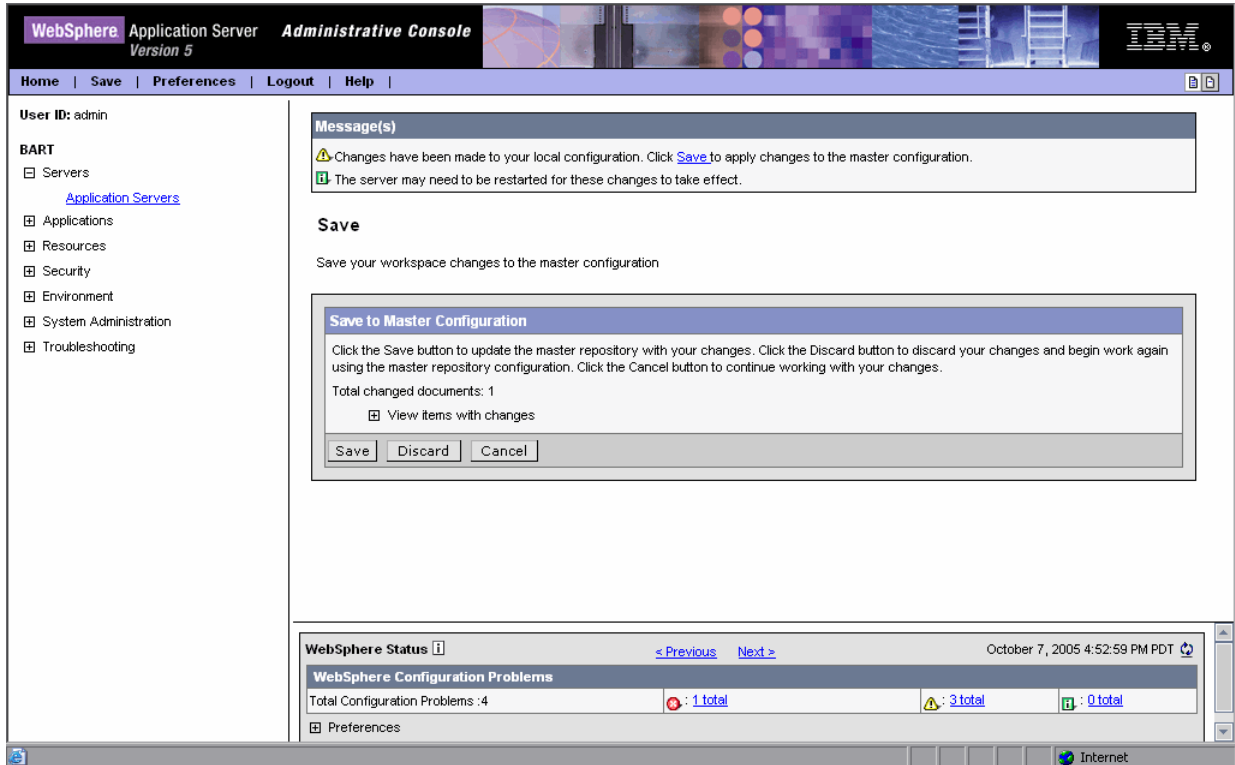
The Configuration tab for the Java Virtual Machine is displayed.



- 8 In the **Boot Classpath** box, type the path to the boot directory for the Java Agent as follows:

```
<java_agent_install_dir>\classes\IBM\1.4.2_06;<java_agent_install_dir>\classes\boot
```

 where `<java_agent_install_dir>` is the path to the location where the Java Agent was installed.
- 9 Scroll to the bottom of the Configuration tab until the command buttons are visible.
- 10 Click **Apply**.
 A message confirms that your changes have been applied.
- 11 Click **Save** to apply the changes to the master configuration.
- 12 In the **Save to Master Configuration** area, click **Save**.



- 13 If you use WebSphere MQ as your communication transport in the Java Agent setup, you should also change the MQ_INSTALL_ROOT variable using the Administrative Console:
 - a Navigate to the WebSphere Variables setting page as follows:
Environment > Manage WebSphere Variables > MQ_INSTALL_ROOT
 - b On the **WebSphere Variables > MQ_INSTALL_ROOT** setting page, in the Value box, enter the WebSphere MQ installation location. You may refer to the location that you entered when you installed and configured Java Agent.
 - c **Apply** and **Save** your changes.
- 14 Restart the WebSphere application server. You do not need to restart the host for the application server.
- 15 To verify that the Java Agent was configured correctly, check for entries in the <java_agent_install_dir>\log\<java_agent_id>.log file. If there are no entries in the file, this indicates that either you did not run the JRE Instrumenter or you did not enter the Xbootclasspath correctly. For details on running the JRE Instrumenter, see [Running the JRE Instrumenter](#) on page 33.

WebSphere 6.1

To configure a WebSphere 6.1 application server:

- 1 Open the WebSphere Application Server Administrative Console.
- 2 Navigate to the Java Virtual Machine page as follows:
Servers > Application servers > Application server *instance name* (for example, server1) > Process Definition > Java Virtual Machine page

- 3 On the Java Virtual Machine page, in the **Generic JVM Arguments** box, enter the following JVM parameter:

```
-javaagent:<java_agent_install_dir>\DiagnosticsAgent\lib\probeagent.jar
```

 where **<java_agent_install_dir>** is the path to the location where the Java Agent was installed.
- 4 **Apply** and **Save** your changes.
- 5 If you use WebSphere MQ as your communication transport in the Java Agent setup, you should also change the MQ_INSTALL_ROOT variable using the Administrative Console:
 - a Navigate to the WebSphere Variables setting page as follows:
Environment > Manage WebSphere Variables > MQ_INSTALL_ROOT
 - b On the **WebSphere Variables > MQ_INSTALL_ROOT** setting page, in the Value box, enter the WebSphere MQ installation location. You may refer to the location that you entered when you installed and configured Java Agent.
 - c **Apply** and **Save** your changes.
- 6 Restart the WebSphere application server. You do not need to restart the host for the application server.
- 7 To verify that the Java Agent was configured correctly, check for entries in the `<java_agent_install_dir>\DiagnosticsAgent\log\<java_agent_id>\probe.log` file. If there are no entries in the file, it indicates that Java Agent was not started correctly.

Running the JRE Instrumenter for WebSphere IDE

If you are using WebSphere IDE, you must run the JRE Instrumenter manually in order to make sure that the correct JAVA executable for the WSAD IDE has been instrumented.

The WSAD IDE has 10 different java.exe executables to choose from. You must make sure to instrument the one that is used to run the IDE.

To instrument the correct java.exe:

- 1 Determine the version of WebSphere that you are using.
- 2 Determine the location of the appropriate java.exe. See the table below.
- 3 Run the JRE Instrumenter as described in [Running the JRE Instrumenter](#) on page 33.

Version	Executable
WAS 5.0	IDE INSTALL\runtimes\base_v5\java\bin\java.exe
WAS 5.1	IDE INSTALL\runtimes\base_v51\java\bin\java.exe

Notes:

Before you can capture data, you must modify the `xbootclasspath` for the application's JVM startup parameters per your IDE's instructions.

If you are NOT running the application inside the IDE, follow the instructions below to configure the JVM parameters using the WebSphere Administrative Console.

Using the JMS Sensor with the WebSphere Application Server

The JMS Sensor does not support the WebSphere Application Server embedded JMS implementation, which is also referred to as the Default Messaging Provider. The embedded JMS implementation is only meant to be used internally by applications running on WebSphere Application Server; messaging into or out of the WebSphere environment is not supported. The full WebSphere MQ product is needed for this purpose.

Adding Interceptors for Sensors

If you want to track business transactions that include cross-application server EJB calls, you need to add the interceptors for Sensors to the application servers by adding the following lines to the server.xml file of your WebSphere application servers:

```
<interceptors xmi:id="Interceptor_X"  
name="com.bristol.tvision.sensor.ejb.ORB.TVClientInterceptor"/>  
  
<interceptors xmi:id="Interceptor_X"  
name="com.bristol.tvision.sensor.ejb.ORB.TVServerInterceptor"/>
```

In "Interceptor X," the X is a unique number that is one higher than the existing Interceptor values in your server.xml file. For example, Interceptor_18, and Interceptor_19 could be the two names that you give if 18 and 19 are not currently in use.

Configuring WebLogic Application Servers

WebLogic application servers are configured by adding the `Xbootclasspath` or `javaagent` JVM parameter to the script that is used to start the application server. WebLogic is started by running shell scripts in a UNIX environment, or command scripts in a Windows environment. Because the startup scripts that WebLogic provides are frequently customized by a site administrator, it is not possible to provide detailed configuration instructions that apply to all situations. Instead, the following sections provide instructions for each of the certified versions of the WebLogic application server for a generic implementation. Your site administrator should be able to use these instructions to show you how to make these changes in your customized environment.



Make sure you understand the structure of the startup scripts, how the property values are set, and how to use environment variables before you make any configuration changes for the Java Agent. Always create a backup copy of any file that you are going to update prior to making the changes.

This section includes the following topics:

- [WebLogic 8.1](#)
- [WebLogic 9.2](#)
- [Configuring Remote-Started WebLogic Managed Servers](#)

WebLogic 8.1

To configure a WebLogic 8.1 application server for the Sun JVM:

- 1 Run the JRE Instrumenter and add the Sun JVM that WebLogic is using.
- 2 Once the JVM is added, click on the **Copy Parameter** button. This will copy the `Xbootclasspath` parameter into the clipboard. For example:

```
JAVA_OPTIONS="-Xbootclasspath/  
p:<java_agent_install_dir>\classes\Sun\1.4.2_04;<java_agent_install_d  
ir>\classes\boot"
```

where `<java_agent_install_dir>` is the path to the directory where the Java Agent was installed.

- 3 Locate the startup script used to start WebLogic for your domain. This file is typically located in a path similar to this example:

```
D:\bea\weblogic81\config\<Dom_Name>\start<Dom_Name>.cmd
```

Replace `<Dom_Name>` by the name of the script that starts the application.

For example, if your domain name is `medrec`, the path looks like this:

```
D:\bea\weblogic81\samples\domains\medrec\startMedRecServer.cmd
```

► In rare cases, if you also deploy the TransactionVision Web Application on this WebLogic server, you should modify the `tvStartMedRecServer.cmd` instead.

- 4 Create a backup copy of the startup script prior to making any changes to the script.
- 5 Use your editor to open the startup script.
- 6 Paste the `Xbootclasspath` parameter saved in the clipboard to the Java command line that starts the application server. The parameter must be placed at the beginning of the Java parameters following any JIT options, such as `-hotspot` or `-classic`.

Following is an example of a WebLogic startup script before adding the `Xbootclasspath` parameter:

```
%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m"-Xbootclasspath/  
p:<java_agent_install_dir>\DiagnosticsAgent\classes\Sun\1.4.2_04;<java_ag  
ent_install_dir>\DiagnosticsAgent\classes\boot" -classpath "%CLASSPATH%"  
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer  
-Dbea.home="C:\bea"  
-Dweblogic.management.password=%WLS_PW%  
-Dweblogic.ProductionModeEnabled=%STARTMODE%  
-Dcloudscape.system.home=./samples/eval/cloudscape/data  
-Djava.security.policy=="C:\bea\weblogic81/lib/weblogic.policy"  
weblogic.Server
```

► The startup script examples are shown with line breaks. The actual scripts do not have line breaks and the text of the commands will wrap on your screen as necessary.

Following is an example of a WebLogic startup script after adding the `Xbootclasspath` parameter:

```
%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -Xbootclasspath/p:"C:\Program  
Files\Hewlett-Packard\common\<java_agent_install_dir>\DiagnosticsAgent\classes\boo  
t"-classpath "%CLASSPATH%"-Dweblogic.Domain=petstore  
-Dweblogic.Name=petstoreServer  
-Dbea.home="C:\bea"  
-Dweblogic.management.password=%WLS_PW%  
-Dweblogic.ProductionModeEnabled=%STARTMODE%  
-Dcloudscape.system.home=./samples/eval/cloudscape/data  
-Djava.security.policy=="C:\bea\weblogic81/lib/weblogic.policy" weblogic.Server
```

- 7 Save the changes to the startup script.

- 8 Restart the WebLogic application server. You do not need to restart the application server host machine.
- 9 To verify that the Java Agent was configured correctly, check for entries in the `<java_agent_install_dir>\log\<java_agent_id>\probe.log` file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter, or did not enter the `Xbootclasspath` correctly.

For details on running the JRE Instrumenter, [Running the JRE Instrumenter](#) on page 33.

To configure a WebLogic 8.1 application server for the JRockit JVM:

- 1 Run the JRE Instrumenter and add the JRockit JVM that WebLogic is using.
- 2 Once the JVM is added, click on the **Copy Parameter** button. This will copy the `Xbootclasspath` parameter into the clipboard.


Following is an example of the `Xbootclasspath` parameter:

```
-Xbootclasspath/p:<java_agent_install_dir>\DiagnosticsAgent\classes\boot
```

where `<java_agent_install_dir>` is the path to the directory where the Java Agent was installed.

- 3 Locate the command file that invokes the WebLogic application server (for example, `startWLS.cmd`). This file is typically located in a path similar to the following example:

```
C:\bea\weblogic81\server\bin\startWLS.cmd
```

 In rare cases, if you also deploy the TransactionVision Web Application on this WebLogic server, you should modify `tvStartWLS.cmd` instead.

- 4 Create a backup copy of the command file prior to making any changes to the script. You may want to give the new copy a name such as `startWLSWithJRockit.cmd`, and use this as the new version of the command file that will be manipulated in the following steps.
- 5 Use your editor to open the startup script.
- 6 Set the `JAVA` executable invoked by WebLogic to JRockit.
 - a Locate the line in the command file where the value of the `JAVA_VENDOR` parameter is set.
 - b Change the value of the `JAVA_VENDOR` variable to point to the JRockit folder as follows:

```
set JAVA_VENDOR=<BEA_HOME_DIR>\jrockit
```

For example:

```
set JAVA_VENDOR=BEA
```

- 7 Modify the Java command line that starts the application server.

- a Locate the line in the command file which begins as follows:

```
%JAVA_HOME%\bin\java %JAVA_VM% %JAVA_OPTIONS% .....
```

- b Indicate the JRockit management URL by specifying the `Xmanagement: class` parameter immediately following the `%JAVA_OPTIONS%` variable.

The following is an example of the `Xmanagement: class` parameter:

```
-Xmanagement: class=com.mercury.opal.capture.proxy.JRockitManagement
```


- c Allow the Java Agent to hook into the application server process by adding the `Xbootclasspath` parameter you saved in the clipboard to immediately following the `%JAVA_OPTIONS%` variable.

The following is an example of a WebLogic startup script before adding the `Xmanagement:class` and `Xbootclasspath` parameters:

- ▶ These startup script examples are shown with line breaks. The actual scripts do not have line breaks and the text of the commands will wrap on your screen as necessary.

```
"%JAVA_HOME%\bin\java" %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.management.username=%WLS_USER%
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.management.server=%ADMIN_URL%
-Dweblogic.ProductionModeEnabled=%PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
weblogic.Server
```

The following is an example of a WebLogic startup script after adding the `Xbootclasspath` parameter:

```
"%JAVA_HOME%\bin\java" %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Xmanagement:class=com.mercury.opal.capture.proxy.JRocketManagement
-Xbootclasspath/p:"C:\Program Files\Hewlett-Packard\common\JavaAgent\
classes\boot"
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.management.username=%WLS_USER%
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.management.server=%ADMIN_URL%
-Dweblogic.ProductionModeEnabled=%PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
weblogic.Server
```

- 8 If you use WebSphere MQ as your communication transport and you choose the connection type as server (default) in the Java Agent setup, you also need to add the path to WebSphere MQ `java/lib` to your system's library path environment variable. For example:

On AIX, add:

```
set LIBPATH=$LIBPATH:/usr/mqm/java/lib
export LIBPATH
```

On Solaris or Linux, add:

```
set LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/mqm/java/lib
export LD_LIBRARY_PATH
```

Replace `lib` with `lib64` if your JVM is 64-bit.

On Windows, you typically do not need such settings because this path has been added to your `PATH` environment variable when you installed WebSphere MQ.

- 9 Save the changes to the command file.
- 10 Restart the WebLogic application server (not the computer; just the application server).

- 11 To verify that the Java Agent was configured correctly, check for entries in the `<java_agent_install_dir>\log\<java_agent_id>\probe.log` file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter or did not enter the `Xbootclasspath` parameter correctly.

For details on running the JRE Instrumenter, see [Running the JRE Instrumenter](#) on page 33.

WebLogic 9.2

To configure a WebLogic 9.2 application server:

- 1 Run the JRE Instrumenter and add the JVM that WebLogic 9.2 is using.
- 2 Once the JVM is added, click on the **Copy Parameter** button. This will copy the JVM parameter into the clipboard. For example:

```
JAVA_OPTIONS="-javaagent:<java_agent_install_dir>\DiagnosticsAgent\lib\probeagent.jar"
```

where `<java_agent_install_dir>` is the path to the directory where the Java Agent was installed.

- 3 Locate the startup script used to start WebLogic for your domain. For example, if your domain name is Medrec, the path looks like this:

```
D:\bea\weblogic92\samples\domains\medrec\bin\startWebLogic.cmd
```



In rare cases, if you also deploy the TransactionVision Web Application (UI) on this WebLogic server, you should modify `tvStartWebLogic.cmd` instead.

- 4 Create a backup copy of the startup script prior to making any changes to the script.
- 5 Use your editor to open the startup script.
- 6 Paste the **JVM** parameter saved in the clipboard to the Java command line that starts the application server. The parameter must be placed at the beginning of the Java parameters following any JIT options, such as **-hotspot** or **-classic**.

Following is an example of a WebLogic startup script after adding the JVM parameter:

```
set JAVA_OPTIONS=
"-javaagent:C:\MercuryDiagnostics\JAVAProbe\DiagnosticsAgent\lib\probeagent.jar" %SAVE_JAVA_OPTIONS%
```

- 7 If you use WebSphere MQ as your communication transport and you choose the connection type as server (default) in the Java Agent setup, you also need to add the path to WebSphere MQ `java/lib` to your system's library path environment variable. For example:

On AIX, add:

```
set LIBPATH=$LIBPATH:/usr/mqm/java/lib
export LIBPATH
```

On Solaris or Linux, add:

```
set LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/mqm/java/lib
export LD_LIBRARY_PATH
```

Replace `lib` with `lib64` if your JVM is 64-bit.

On Windows, you typically do not need such settings because this path has been added to your PATH environment variable when you installed WebSphere MQ.

- 8 Save the changes to the startup script.
- 9 Restart the WebLogic application server. You do not need to restart the application server host machine.
- 10 To verify that the Java Agent was configured correctly, check for entries in the `<java_agent_install_dir>\log\<java_agent_id>\probe.log` file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter, or did not enter the JVM parameter correctly.

For details on running the JRE Instrumenter, see [Running the JRE Instrumenter](#) on page 33.

Configuring Remote-Started WebLogic Managed Servers

For a WebLogic server using Node Manager to start, follow the same steps in the previous sections to run `jreinstrumenter` to instrument the JRE your application server is using. Then copy and paste the output from the `jreinstrumenter` to corresponding `<arguments>` tag for your server in the `config.xml` file. The `config.xml` file is usually located in the `<your_domain>/config` directory.

Following are examples for JVM 1.5 and JVM 1.4:

JVM 1.5 Example

In this example, if your managed server is called `cluster_server1`, you would modify `config.xml` to add the `javaagent` argument (in red) as follows:

```
<server>
  <name>cluster_server1</name>
  <ssl>
    <enabled>>false</enabled>
  </ssl>
  <machine>Machine-0</machine>
  <listen-port>8001</listen-port>
  <cluster>ChenCluster</cluster>
  <web-server>
    <web-server-log>
      <number-of-files-limited>>false</number-of-files-limited>
    </web-server-log>
  </web-server>
  <listen-address></listen-address>
  <server-start>
    <arguments>-javaagent:e:/myprobe/javaprobe/build/lib/probeagent.jar</
arguments>
    <password-encrypted>{3DES}9pcg8snNNwA=</password-encrypted>
  </server-start>
  <jta-migratable-target>
    <user-preferred-server>cluster_server1</user-preferred-server>
    <cluster>ChenCluster</cluster>
  </jta-migratable-target>
</server>
```

JVM 1.4 Example

For JVM 1.4, you would copy the output from the `jreinstrumenter` to the `<arguments>` tag in `config.xml` for the server you want to monitor. In this example, the `jreinstrumenter` argument (in red) is added:

```
<server>
  <name>cluster_server1</name>
  <ssl>
    <enabled>>false</enabled>
  </ssl>
  <machine>Machine-0</machine>
  <listen-port>8001</listen-port>
  <cluster>ChenCluster</cluster>
  <web-server>
    <web-server-log>
      <number-of-files-limited>>false</number-of-files-limited>
    </web-server-log>
  </web-server>
  <listen-address></listen-address>
  <server-start>
    <arguments>-Xbootclasspath/
<java_agent_install_dir>\DiagnosticsAgent\classes\Sun\1.4.2_08;<java_agent_in
stall_dir>\DiagnosticsAgent\classes\boot</arguments>
    <password-encrypted>{3DES}9pcg8snNNwA=</password-encrypted>
  </server-start>
  <jta-migratable-target>
    <user-preferred-server>cluster_server1</user-preferred-server>
    <cluster>ChenCluster</cluster>
  </jta-migratable-target>
</server>
```

Configuring Messaging System Providers

TransactionVision uses queues for the communication between the Analyzers and the Sensors. You need at least three queues: the configuration queue, the event queue, and the exception queue. The default name of these queues are `TVISION.CONFIGURATION.QUEUE`, `TVISION.EVENT.QUEUE`, and `TVISION.EXCEPTION.QUEUE`, respectively. You must set up these queues on your message system provider in a vendor-specific way. See Chapter 5 “Managing Communication Links” in the *TransactionVision Administration Guide* for details.

IBM WebSphere MQ

You may create the queues on your queue managers using the IBM WebSphere MQ `runmqsc` utility program or the MQ Explorer graphical user interface that is available on Windows and Linux. For using the client connection type, you also need to define a server connection channel and a listener on your queue manager. See the vendor's documentation for details.

TIBCO EMS

You may create the queues using the TIBCO EMS Administration tool. See the vendor's documentation for details.

Progress SonicMQ

You may create the queues on your SonicMQ brokers using the SonicMQ Management Console. See the vendor's documentation for details.

BEA WebLogic JMS

You may use the WebLogic Administrative Console to configure a JMS server and queues. Following is a summary of the steps needed for WebLogic 8.1. See the vendor's documentation for details.

- 1 Create a Temporary template:
Choose **Services > JMS > Templates**, and click **Configure a new JMS Template....**
- 2 Create a persistent JMS store:
Choose **Services > JMS > Stores**, and click **Configure a new JMS Store....**
You need to specify a directory path for the store (such as /tmp).
Alternatively, you may use a JDBC store instead.
- 3 Create a JMS server:
Choose **Services > JMS > Servers**, and click **Configure a new JMS Server....**
On this page, you need to specify the Persistent Store and the Temporary Template created above.
- 4 Create the queues:
Choose **Services > JMS > Servers > your_server_name**, and click **Configure Destinations....**, then click **Create a new JMS Queue....**
- 5 Create a Connection Factory:
Choose **Services > JMS > ConnectionFactories** and click **Create a new JMS Connection Factory....**

3 Installing WebSphere MQ and User Event Sensors on UNIX Platforms

This chapter provides instructions on installing TransactionVision Sensors for WebSphere MQ and User Event on UNIX platforms. To install TransactionVision Sensors for Java applications and application servers, see [Chapter 2, Installing and Configuring Java Agent](#).

Note that TransactionVision provides unique packages for each Sensor.

Installing Sensors

Installation Files

The following table shows the installation file names for the TransactionVision packages for each platform. Note that if you install the User Event Sensor, the common package is installed automatically because this Sensor depends on it.

Platform	Files
AIX	tvision_common_750_aix_power.bff tvision_sensor_wmq_750_aix_power.bff tvision_sensor_userevent_750_aix_power.bff
HP-UX	tvision_sensor_wmq_750_hpux_ia64_tar.gz tvision_sensor_wmq_750_hpux_parisc_tar.gz
Linux	tvision_common_750_linux_x86.rpm tvision_sensor_wmq_750_linux_x86.rpm tvision_sensor_userevent_750_linux_x86.rpm
Solaris	tvision_common_750_solaris_sparc_pkg.gz tvision_sensor_wmq_750_solaris_sparc_pkg.gz tvision_sensor_userevent_750_solaris_sparc_pkg.gz

Installation Steps

- 1 Change to the directory location of the TransactionVision installation files (either a CD device or download directory).
 - ▶ On Solaris and HP-UX, you must copy the installation files from the CD device to a temporary directory on your host's hard drive.
- 2 Log in as superuser:

```
su
```
- 3 Enter the following command to begin the installation procedure:

```
./tvision_install_750_unix.sh
```

The following menu is displayed:

```
This script will install/uninstall different TransactionVision
components.
```

```
Unzipping/Untaring common package files ...
Unzipping/Untaring Analyzer package files ...
Unzipping/Untaring Web package files ...
Unzipping/Untaring WebSphere MQ Agent package files ...
Unzipping/Untaring User Event Agent package files ...
```

► The “Unzipping...” lines above are only for Solaris and HP-UX installations.

The following TransactionVision packages are available for installation:

```
1. TransactionVision Analyzer
2. TransactionVision Web
3. TransactionVision WebSphere MQ Agent
4. TransactionVision User Event Agent

99. All of above
q. Quit install
```

Please specify your choices (separated by ,) by number/letter:

► The actual options and numbers depend on the installation files available on your computer.

- 4 To install only a single component, type the number associated with the TransactionVision component package and press **Enter**.

To install multiple, but not all components, type the numbers associated with the components you wish to install, separated by commas, and press **Enter**. For example, to install all Sensors from the menu above, type the following and press **Enter**:

```
3,4
```

To install all available components, type **99** and press **Enter**.

The installation script installs the specified package(s), then displays the menu again.

- 5 To quit the installation procedure, type **q** and press **Enter**. To install additional components, see the installation instructions for those components.

Rebinding the WebSphere MQ Sensor on AIX

For the WebSphere MQ Sensor on the AIX platform, the installation calls the **rebind_sensor** script to relink the Sensor library. If you install a WebSphere MQ support pack that modifies the WebSphere MQ libraries (`libmqm.a`, `libmqic.a`, `libmqm_r.a`, `libmqic_r.a`), you must run this script again in order for sensed applications to run correctly. For more information about this script, see [Appendix A, Utilities Reference](#).

Uninstalling Sensors

To uninstall TransactionVision components, perform the following steps:

- 1 Log in as superuser:

```
su
```

- 2 Enter the following command:

```
./tvision_install_750_unix.sh -u
```

The following menu is displayed (note that actual options depend on the TransactionVision packages installed on your computer):

The following TransactionVision packages are installed on the system:

1. TransactionVision Web
 2. TransactionVision Analyzer
 3. TransactionVision WebSphere MQ Agent
 4. TransactionVision User Event Agent
-
99. All of above
 - q. Quit uninstall

Please specify your choices (separated by ,) by number/letter:

- 3 Type the number associated with the TransactionVision package you wish to uninstall and press **Enter**.

To uninstall all TransactionVision components, type **99** and press **Enter**.

The installation script uninstalls the specified package, then displays the menu again.

- 4 To quit the uninstall, type **q** and press **Enter**. If the common package is the only TransactionVision package still installed, it will be uninstalled automatically.

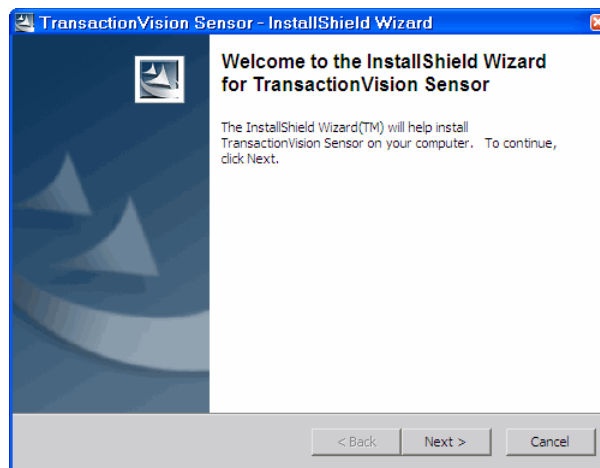
4 Installing WebSphere MQ and User Event Sensors on Windows

The TransactionVision Sensors for WebSphere MQ and User Event are installed as a single package on Windows. This chapter provides instructions for installing these Sensors. (To install TransactionVision Sensors for Java applications and application servers, see [Chapter 2, Installing and Configuring Java Agent](#).)

Note that you must be logged into the target system either as Administrator or as a user with Administrator privileges.

To install this package, perform the following steps:

- 1 Close all Windows programs currently running on your computer.
- 2 In the Windows Explorer, double-click **tvision_sensor_750_win.exe**. The InstallShield Welcome screen appears.



- 3 Click **Next>** to display the InstallShield Save Files screen.
- 4 To use the default folder for extracting installation files, click **Next>**. To choose a different folder, click **Change**, select the desired folder, then click **Next>**. InstallShield extracts the installation files.

If this is the first time installing TransactionVision Sensors on this computer, continue with [Initial Installation](#). If an earlier version of TransactionVision Sensors is installed on this computer, continue with [Upgrade Installation](#).

Initial Installation

For an initial installation, the Setup Welcome screen is displayed.

- 1 On the Setup Welcome screen, click **Next>** to display the TransactionVision license agreement.

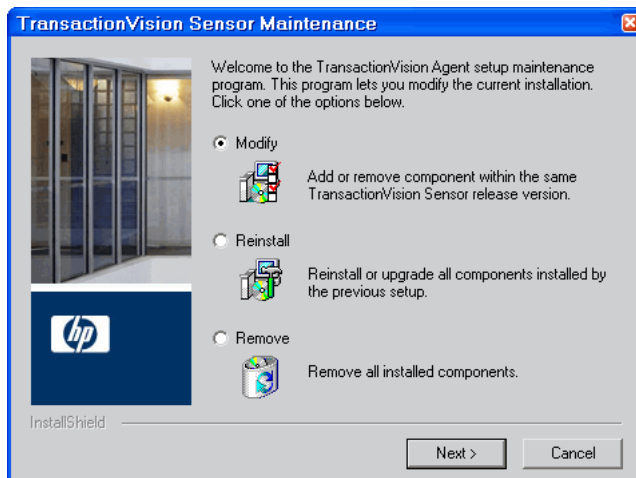
- 2 Click **Yes** to accept the license agreement. The User Information screen appears.
- 3 Enter your name and company name, then click **Next>**. The Destination Location screen appears.
- 4 To install the Sensors for WebSphere MQ and User Event, select **Complete** and click **Next>**. To install only some Sensors, select **Custom**, click **Next>**, select the desired Sensors, and click **Next>**.

The selected Sensors are installed in the specified location. The Setup Complete page appears.

- 5 Click **Finish** to complete the installation.

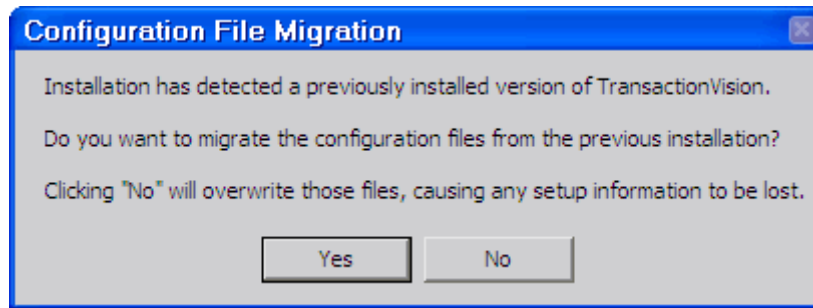
Upgrade Installation

For an upgrade installation, double-click **tvision_sensor_750_win.exe** and click **Next>** on the InstallShield Welcome screen to display the Sensor setup maintenance menu:



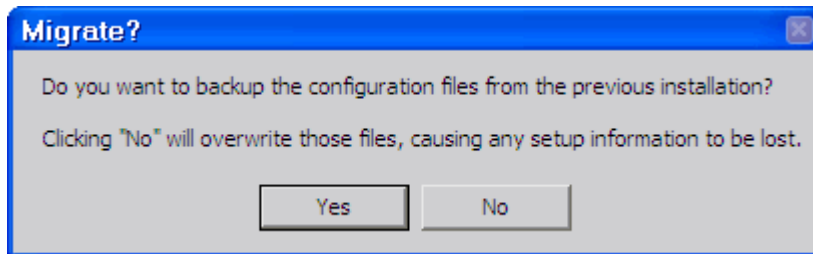
- 1 Select one of the following to upgrade the TransactionVision installation (to modify the installation, see [Modifying the Installation](#) on page 62):
 - If you wish to install TransactionVision Sensors with different settings from the previous installation, select **Remove** and click **Next>** to uninstall the previous installation, then begin the installation procedure again.

- If you are upgrading from a previous release, select **Reinstall** and click **Next>** to install TransactionVision Sensors using the settings from the previous installation. The Configuration File Migration dialog appears:



- 2 To maintain configuration information from the previous installation, click **Yes**. The installation wizard makes a backup copy of existing configuration files, installs the new version of TransactionVision, and opens an MS-DOS window to migrate existing configuration files to the new version. When complete, the Setup Complete screen appears.

To overwrite existing configuration files, click **No**. The installation wizard displays a message box asking whether you want to make a backup copy of existing configuration files before continuing the installation.



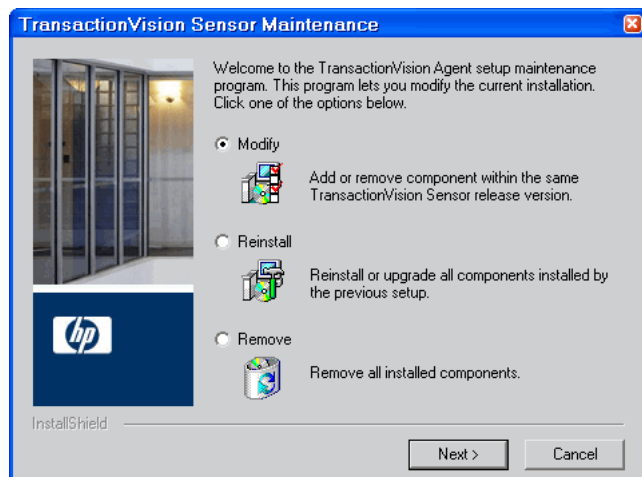
Click **Yes** to create a backup copy or **No** to continue the installation without backing up configuration files. The installation wizard then installs the new version of TransactionVision, overwriting existing configuration files, and displays the Setup Complete screen.

- 3 The installation wizard installs the new version of TransactionVision and displays the Setup Complete screen.
- 4 Click **Finish** to complete the installation.

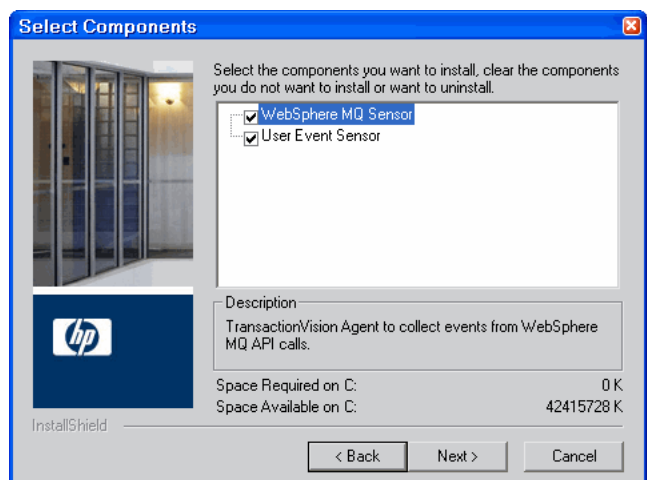
Modifying the Installation

After you install TransactionVision Sensors on a host, you may wish to modify your installation. For example, suppose you initially install the WebSphere MQ Sensor on a host, then later decide to install the User Event Sensor. To modify your installation perform the following steps:

- 1 Double-click **tvision_sensor_750_win.exe** and click **Next>** on the InstallShield Welcome screen to display the TransactionVision Sensor Maintenance screen:



- 2 To install an additional Sensor or remove an installed Sensor, select **Modify** and click **Next>** to display the Select Components screen.



- 3 Select the Sensors you want to install and click **Next>** to run the modified installation.
- 4 Click **Finish** to complete the installation.

Uninstalling Sensors

To uninstall TransactionVision components, perform the following steps:

- 1 From the Start menu, choose **Settings > Control Panel**.
- 2 Double-click **Add/Remove Programs**.
- 3 Select the HP TransactionVision Sensor package and click **Change/Remove**. The maintenance menu screen appears.
- 4 Select **Remove** and click **Next>** to remove TransactionVision components.
- 5 Click **OK** to confirm that you wish to uninstall the specified package. The specified package is uninstalled. The following types of files are not deleted:

- Any files added after the installation.
- Any shared files associated with packages that are still installed.

If shared files do not appear to be associated with any installed packages (for example, if all other TransactionVision packages have been uninstalled), the Shared File Detected screen appears.

- To leave all shared files installed, check Don't display this message again and click **No**.
- To leave the current file, but display this message for any other shared files, click **No**.
- To delete the shared file, click **Yes**.

The Uninstallation Complete screen appears.

- 6 Click **Finish** to complete the uninstallation procedure.

5 Installing Sensors on i5/OS

To install the Sensor on the i5/OS platform, perform the following steps:

- 1 If an earlier version of the TransactionVision Sensor is installed, use the following command to uninstall it:

```
DLTLICPGM LICPGM(3RBB9ES)
```

- 2 On an i5/OS machine, either find an existing library to use or create a new a library to copy the installation file to (for example, TVTMP).
- 3 On a PC, FTP the Sensor installation file `tvision_sensor_wmq_750_i5os_as400.savf` from the CD-ROM and rename it to `sensor750.savf` to the library created in step 1 on your i5/OS machine. Be sure to set binary mode transfer as follows:

```
ftp> bin
ftp> cd /qsys.lib/tvtmp.lib
ftp> put tvision_sensor_wmq_750_i5os_as400.savf sensor750.savf
```

- 4 On the i5/OS machine, run the following command to install the Sensor. Note that you may need to replace TVTMP in the command with the name of the library in which the `sensor750.savf` package resides

```
RSTLICPGM LICPGM(3RBB9ES) DEV(*SAVF) SAVF(TVTMP/SENSOR750)
```

- 5 Verify the installation with the following command:

```
DSPSFWRSC
```

- 6 To use the C Sensor, bind your programs to TVSENSOR/LIBMQM.
- 7 If a new temporary library was created in step 2, it may now be safely deleted.

6 Installing Sensors on z/OS

This chapter provides instructions for installing the following TransactionVision Sensors on the IBM z/OS platform:

- CICS, WebSphere MQ (WMQ) Batch, and WebSphere MQ (WMQ) IMS
- WebSphere MQ CICS and WebSphere MQ IMS Bridge

For additional RACF requirements for authorizations, firewall settings, and MIPS, see [Appendix C, Additional z/OS Settings](#).

Installing the CICS, WebSphere MQ Batch, and WebSphere MQ IMS Sensors on IBM z/OS

To install these Sensors on the z/OS platform, perform the following steps, substituting a valid data set name high-level qualifier for &hlq, for example, TVISION.

Once you complete this installation procedure, see the *TransactionVision Administration Guide* for additional configuration instructions.

The following may be used to perform either an SMP/E or non-SMP/E install. See step 4 for additional details.

- 1 FTP the Sensor installation files from the Sensors/zos/install directory of the TransactionVision CD-ROM to your z/OS system using binary mode and specifying the correct data set attributes on the quote command. For example:

```
ftp> quote site fixrecfm 80 lrecl=80 recfm=fb blksize=3120
vol=&volser u=&unit pri=30 sec=5 tr
```

(where &volser is the target disk volume serial number and &unit is the target disk device type / esoteric)

```
ftp> bin
ftp> put sld750.f1 '&hlq.sld750.f1'
ftp> put sld750.f2 '&hlq.sld750.f2'
ftp> put sld750.f3 '&hlq.sld750.f3'
ftp> put sld750.mcs '&hlq.sld750.mcs'
```



If any of the above ftp commands fail, it may be necessary to delete the target dataset(s) before re-executing.

- 2 Use the TSO RECEIVE command to create the product distribution data sets from the transferred files. The following table shows the RECEIVE commands and the filename to enter for each one in response to the prompt, “INMR906A Enter restore parameters or ‘DELETE’ or ‘END’.”

Command	Filename
RECEIVE INDSNAME('&hlq.SLD750.F1')	DSN('&hlq.ASLD750.F1')
RECEIVE INDSNAME('&hlq.SLD750.F2')	DSN('&hlq.ASLD750.F2')
RECEIVE INDSNAME('&hlq.SLD750.F3')	DSN('&hlq.ASLD750.F3')
RECEIVE INDSNAME('&hlq.SLD750.MCS')	DSN('&hlq.ASLD750.SMPMCS')

- 3 The data sets created make up an SMP/E install package in RELFILE format. Verify the creation of the following SMP/E input data sets:

Data Set	Member(s)	Description
&hlq.ASLD750.F1	ASLD750	SMP/E JCLIN
&hlq.ASLD750.F2	SLDMOD01-39	Sensor modules
	MQCONN	Dummy module for WebSphere 5.1 installations.
	LKSTBN	Sample job to rebind WBI library
&hlq.ASLD750.F3	DUMYCICS	Sample customization job
	DUMYIMS	Sample customization job
	SLDACCPT	Sample SMP/E job
	SLDALLOC	Sample SMP/E job
	SLDAPPLY	Sample SMP/E job
	SLDCICSD	Sample customization job
	SLDCRTQS	Sample customization job
	SLDDDDEF	Sample SMP/E job
	SLDDZON	Sample SMP/E job
	SLDGZON	Sample SMP/E job
	SLDINSTL	Sample non-SMP/E install job
	SLDRECV	Sample SMP/E job
	SLDTZON	Sample SMP/E job
	TVISION	Sample product startup procedure
	TVISIONC	Sample product startup procedure
	TVISIONM	Sample product startup procedure
TVISIONR	Sample recovery procedure	
VERSION	Build number information	
&hlq.ASLD750.SMPMCS	SMP/E MCS	

If any of the above data sets are missing, recheck Steps 1 and 2. If discrepancies remain unresolved, please contact HP Support for assistance.

- 4 The sample jobs listed below perform an SMP/E product installation. Read the comments contained in each member, in addition to the instructions for steps 5-9 carefully. Together, this information should be sufficient to appropriately tailor each jobstream to meet your local site requirements. (The SMP/E FMID for this installation is ASLD750.)

```
&hlq.ASLD750.F3 (SLDALLOC)
&hlq.ASLD750.F3 (SLDGZON)
&hlq.ASLD750.F3 (SLDDZON)
&hlq.ASLD750.F3 (SLDTZON)
&hlq.ASLD750.F3 (SLDDDEF)
&hlq.ASLD750.F3 (SLDRECV)
&hlq.ASLD750.F3 (SLDAPPLY)
```



You may perform a non-SMP/E install by customizing and running the SLDINSTL member and skipping steps 5-9 and step 18. All other steps should be performed.

- 5 Allocate the target and distribution libraries for the product by customizing and running the SLDALLOC member. The following libraries will be created by the job:

Library	Description
target library &THLQUAL.SSLDLOAD	Sensor load modules
target library &THLQUAL.SSLDINST	Sensor installation sample JCL
target library &THLQUAL.SSLDSAMP	Sensor samples
target library &THLQUAL.SSLDPROC	Sensor sample JCL procedures
target library &THLQUAL.SSLDAUTH	Sensor load modules
distribution library &DHLQUAL.ASLDMOD	Sensor distribution modules
distribution library &DHLQUAL.ASLDINST	Sensor installation sample JCL
distribution library &DHLQUAL.ASLDSAMP	Sensor samples
distribution library &DHLQUAL.ASLDPROC	Sensor sample JCL procedures

- 6 If you are installing the Sensor into an existing SMP/E global zone, skip this step. If you are installing the Sensor into a new SMP/E global zone, customize and run members—SLDGZON, SLDDZON, and SLDTZON—to create new global, target, and distribution zones.
- 7 Create the SMP/E DDDEFs in the distribution and target zones by customizing and running the SLDDDEF member. You may install into any SMP/E target zone desired.

The following DDDEFs will be created by the SLDDDEF member:

Library	Description
target zone DDDEF SSLDLOAD	Points to the SSLDLOAD target library allocated in Step 4.
target zone DDDEF SSLDINST	Points to the SSLDINST target library allocated in Step 4.
target zone DDDEF ASLDMOD	Points to the ASLDMOD distribution library allocated in Step 4.

Library	Description
target zone DDDEF SCEELKED	Points to the LE SCEELKED library. This DDDEF might already exist in your chosen target zone.
target zone DDDEF SCSQLOAD	Points to the WebSphere MQ SCSQLOAD library. This DDDEF might already exist in your chosen target zone.
target zone DDDEF SDFHLOAD	Points to the CICS SDFHLOAD library. This DDDEF might already exist in your chosen target zone.
target zone DDDEF SSLDSAMP	Points to the SSLDSAMP target library allocated in Step 4.
target zone DDDEF SSLDPROC	Points to the SSLDPROC target library allocated in Step 4.
target zone DDDEF SSLDAUTH	Points to the SSLDAUTH target library allocated in Step 4.
distribution zone DDDEF ASLDMOD	Points to the ASLDMOD distribution library allocated in Step 4.
distribution zone DDDEF ASLDINST	Points to the ASLDINST distribution library created in Step 4.
distribution zone DDDEF ASLDSAMP	Points to the ASLDSAMP distribution library created in Step 4.
distribution zone DDDEF ASLDPROC	Points to the ASLDPROC distribution library created in Step 4.



Important! If you do NOT have CICS installed on your system, customize and submit the JCL DUMYCICS to generate dummy CICS modules. Then set the CICS high level qualifier (&CICSQUAL) to be the same as the target library high level qualifier (%tvlib). If you do NOT have IMS installed on your system, customize and submit the JCL DUMYIMS to generate dummy IMS modules. Then set the IMS high level qualifier (&IMSQUAL) to be the same as the target library high level qualifier (%tvlib).

- 8 SMP/E RECEIVE the Sensor installation package product by customizing and running the SLDRECV member.

- 9 SMP/E APPLY the Sensor installation package by customizing and running the SLDAPPLY member. After APPLY processing, verify the following:

Library	Contents
&THLQ.SSLDLOAD	SLDPCCX, SLDPCCR, SLDPCCMX, SLDPCCPX, SLDPCCSX, SLDPDSX, SLDPFCX, SLDPICX, SLDPPCX, SLDPPSX, SLDPDTCX, SLDPDTCX, SLDPDTSX, MQCONN, SLDPMDR, SLDPMECB, SLDPMECQ, SLDPMECR, SLDPMECS, SLDPMEM, SLDPSTBB, SLDPSTBQ, SLDPSTBR, SLDPSTBS
&THLQ.SSLDINST	SLDACCT, SLDALOC, SLDAPPLY, SLDCICSD, SLDCRTQS, SLDDDDDEF, SLDDZON, SLDGZON, SLDINSTL, SLDRECV, SLDTZON, VERSION
&THLQ.SSLDPROC	TVISION, TVISIONC, TVISIONM, TVISIONR
&THLQ.SSLMSAMP	No members in this release
&THLQ.SSLDAUTH	SLDPASM, SLDPBQM, SLDPCCMD, SLDPCCSC, SLDPCCSI, SLDPCCSM, SLDPITM, SLDPMON, SLDPSSS, SLDPDTCX, SLDPDTSX, SLDPMSI, SLDPMSM

- 10 You may wish to review Chapter 11 in the *TransactionVision Administration Guide* before performing the remaining installation steps.
- 11 Update your CICS CSD file with the required resource definitions for the Sensor by customizing and running the SLDCICSD member. If your environment consists of multiple CICS regions with separate CSDs or different startup lists, repeat this step for each CICS region to be monitored by the Sensor.
- 12 Place the CICS Sensor program load modules in a library in your DFHRPL concatenation. Either copy the modules into an existing library already in your DFHRPL concatenation or add the SSLDLOAD library to the DFHRPL concatenation. The CICS modules are: SLDPCCX, SLDPCCMX, SLDPCCPX, SLDPCCSX, SLDPDSX, SLDPFCX, SLDPICX, SLDPPCX, SLDPPSX, SLDPDTCX, SLDPDTCX, and SLDPDTSX. If you run multiple CICS regions with separate startup JCL, repeat this step for each CICS region to be monitored by the Sensor.
- 13 Optional. To automatically enable and disable the Sensor's CICS exit programs at CICS startup and shutdown, define SLDPCCSX as a second pass PLTPI and SLDPCCPX as a second pass PLTSD. Refer to *CICS Resource Definition Guide*, DFHPLT section. Add these definitions to your existing DFHPLTxx. If new PLTs are defined, add references to them in the CICS System Initialization Table (SIT). Refer to *CICS System Definition Guide*, "Specifying CICS system initialization parameters." Repeat this step for each CICS region to be monitored by the Sensor.

Sample SIT entries:

```
...
PLTPI=BI,
PLTSD=BS,
...
```

Sample definitions of PLTs with the suffixes BT and BS:

```
//DFHPLTPI      EXEC DFHAUPL
//ASSEM.SYSUT1 DD *
DFHPLT TYPE=INITIAL, SUFFIX=BI
DFHPLT TYPE=ENTRY, PROGRAM=DFHDELIM
```



```

        DFHPLT TYPE=ENTRY, PROGRAM=SLDPCSX
        DFHPLT TYPE=FINAL
        END
// *
//DFHPLTSD      EXEC DFHAUPLD
//ASSEM.SYSUT1 DD *
        DFHPLT TYPE=INITIAL, SUFFIX=BS
        DFHPLT TYPE=ENTRY, PROGRAM=SLDPCPX
        FHPLT TYPE=ENTRY, PROGRAM=DFHDELIM
        DFHPLT TYPE=FINAL
        END
//

```

➤ If the optional step above is not performed, CICS transaction SLDS can be executed to enable Sensors, and CICS transaction SLDP can be executed to disable them.

- 14 APF-authorize the TransactionVision library, &hlq.SSLDAUTH.

➤ &hlq.SSLMLOAD must NOT be authorized. Refer to *MVS Initialization and Tuning Reference*, PROGxx or IEAAPFxx system parameters for additional details.

- 15 The TVISION address space must be non-swappable. TVISION assures this by issuing a SYSEVENT TRANSWAP macro, so it is not necessary to include the program in the PPT. If you do include it, specify the entry as follows:

```

        PPT PGMNAME(SLDPTVM) CANCEL KEY(8) NOSWAP NOPRIV
        NODSI PASS SYST AFF(NONE) NOPREF

```

Refer to *MVS Initialization and Tuning Reference*, SCHEDxx system parameters.

- 16 Customize the sample Sensor startup procedures, TVISION and TVISIONC, in the &hlq.SSLDPROC data set and copy them to an appropriate procedure library for your site. See Chapter 11 in the *TransactionVision Administration Guide* for guidelines for customizing the startup procedures.
- 17 Create appropriate Sensor configuration and event queues on the WebSphere MQ queue manager to be used for communication between the CICS Sensor Driver component and the TransactionVision Analyzer. Customize and run the SLDCRTQS member. See Chapter 11 of the *TransactionVision Administration Guide* for more information.
- 18 When you are satisfied with the results of your installation, perform an SMP/E ACCEPT by customizing and running the SLDACCP member.
- 19 These sensors use EZASMI to retrieve the IP address. As a result, the startup procedures need an OMVS environment that is RACF authorized, for example TVISIONC or TVISIONM.

Installing the WebSphere MQ CICS and WebSphere MQ IMS Bridge Sensors on IBM z/OS

There is a single Sensor installation package for WebSphere MQ CICS and WebSphere MQ IMS bridge Sensors on z/OS. Some configuration steps only apply to the WebSphere MQ Sensor for CICS; they are noted as “CICS only.” If you do not intend to use the WebSphere MQ Sensor for CICS, omit those steps. The WebSphere MQ IMS Bridge Sensor also requires additional steps, which are detailed in [Additional Setup for the WebSphere MQ IMS Bridge Sensor](#) on page 83. If you do not intend to use the WebSphere MQ IMS Bridge Sensor, omit those steps.

Before You Install the WebSphere MQ Sensor for CICS

The WebSphere MQ Sensor for z/OS CICS consists of two component programs. The program that monitors an application’s WebSphere MQ API calls is an WebSphere MQ API crossing exit program named CSQCAPX. This program must be installed in your CICS region in order for the Sensor to collect any events. A second program, SLMC (with an associated CICS transaction of the same name), is optional. It can be run to automatically enable and disable the crossing exit Sensor program in your CICS region as needed by the Sensor. You do not have to run SLMC for the Sensor to function correctly, but doing so improves your region’s WebSphere MQ application performance when no Analyzer is monitoring the region. If you choose not to run SLMC, ensure that the crossing exit is left enabled in your CICS region.

WebSphere MQ API crossing exit programs for CICS are required to have the fixed program name CSQCAPX. Thus, only a single WebSphere MQ API crossing exit can be installed in a given CICS region at one time. If you already have a program of this name installed in the target CICS region, remove the existing CSQCAPX load module from your DFHRPL concatenation and delete or disable the existing CICS program definitions for CSQCAPX before installing the Sensor.

The WebSphere MQ Sensor for z/OS CICS requires z/OS Language Environment runtime support. Before installing the Sensor, be certain that your target CICS region has LE support enabled. The Sensor requires the CICS region to support LE programs compiled from the C language. The procedure for enabling LE support is described in the *CICS System Definition Guide* (the “Installing Application Programs” chapter in the CICS TS documentation). Consult the documentation for your version of CICS for full details.

SMPE Installation Procedure

To install these Sensors on z/OS, perform the following steps:


- 1 FTP the Sensor installation files from Sensors/zos/install directory of the TransactionVision CD-ROM to your z/OS machine. Be sure to set binary mode transfer and the data set characteristics as follows:

```
ftp> quote site fixrecfm 80 lrecl=80 recfm=fb blksize=3120 vol=&volser
u=&unit pri=30 sec=5 tr
```

(where &volser is the target disk volume serial number and &unit is the target disk device type / esoteric)

```
ftp> bin
ftp> put slm750.f1 '&hlq.slm750.f1'
ftp> put slm750.f2 '&hlq.slm750.f2'
```

```
ftp> put slm750.f3 '&hlq.slm750.f3'
ftp> put slm750.mcs '&hlq.slm750.mcs'
```

 If any of the above ftp commands fail, it may be necessary to delete the target dataset(s) before re-executing.

- 2 Use the TSO RECEIVE command to create the SMP/E input data sets from the transferred files. The following table shows the RECEIVE commands and the filename to enter for each one in response to the prompt, "INMR906A Enter restore parameters or 'DELETE' or 'END':"

Command	Filename
RECEIVE INDSNAME('&hlq.SLM750.F1')	DSN('&hlq.ASLM750.F1')
RECEIVE INDSNAME('&hlq.SLM750.F2')	DSN('&hlq.ASLM750.F2')
RECEIVE INDSNAME('&hlq.SLM750.F3')	DSN('&hlq.ASLM750.F3')
RECEIVE INDSNAME('&hlq.SLM750.MCS')	DSN('&hlq.ASLM750.SMPMCS')

- 3 The data sets created are an SMP/E install package in RELFILE format. Verify the creation of the following SMP/E input data sets:

Data Set	Member	Description
&hlq.ASLM750.SMPMCS		SMP/E MCS file for Sensor
&hlq.ASLM750.F1	ASLM750	JCLIN for SMP/E

Data Set	Member	Description
&hlq.ASLM750.F2	DFHEAII	Dummy module for CICS
	MQCONNX	Dummy module for MQ51
	SLMBCNFG	Configuration queue table
	SLMMOD01	Sensor crossing exit program for z/ OS CICS
	SLMMOD02	SLMC crossing exit management program for z/OS CICS
	SLMMOD10	WebSphere MQ IMS Bridge support module
	SLMMOD11	WebSphere MQ IMS Bridge support module
	SLMMOD12	WebSphere MQ IMS Bridge support module
	SLMMOD13	WebSphere MQ IMS Bridge support module
	SLMMOD14	WebSphere MQ IMS Bridge support module
	SLMMOD15	WebSphere MQ IMS Bridge support module
	SLMMOD16	WebSphere MQ IMS Bridge support module
	SLMMOD17	WebSphere MQ IMS Bridge support module
	SLMMOD18	WebSphere MQ IMS Bridge support module
SLMMOD19	WebSphere MQ IMS Bridge support module	
SLMYIOE0	WebSphere MQ IMS Bridge support module	

Data Set	Member	Description
&hlq.ASLM750.F3	DUMYCICS	Sample customization job
	DUMYIMS	Sample customization job
	SLMACCPT	Sample JCL for SMP/E Accept step
	SLMALLOC	Sample JCL for TransactionVision SMP/E target and distribution library allocation
	SLMAPPLY	Sample JCL for SMP/E Apply step
	SLMBCFGQ	Sample job to change the default TransactionVision configuration queue name for the batch and IMS Sensors or the WebSphere MQ IMS Bridge Sensor
	SLMCICSD	Sample JCL for updating CICS CSD file with Sensor program definition
	SLMCRTQS	Sample JCL to create TransactionVision communication queues
	SLMDDDEF	Sample JCL for creating SMP/E DDDEFs for TransactionVision
	SLMDZON	Sample JCL to create the SMP/E distribution zone
	SLMGZON	Sample JCL to create the SMP/E global zone
	SLMINSTL	Sample non-SMP/E install job
	SLMRECV	Sample JCL for SMP/E Receive step
	SLMTZON	Sample JCL to create the SMP/E target zone
	TVISIONB	Sample procedure to start the WebSphere MQ IMS Bridge Sensor control address space
TVISIOND	Sample procedure to start the WebSphere MQ IMS Bridge Sensor event dispatcher address space	
VERSION	Text file containing TransactionVision build number information	

If any of the above data sets are missing, recheck Steps 1 and 2. If in doubt, contact HP support for assistance.

- 4 The sample jobs listed below perform an SMP/E product installation. Please read the comments contained in each member, in addition to the instructions for steps 5-9 carefully. Together, this information should be sufficient to appropriately tailor each jobstream to meet your local site requirements. (The SMP/E FMID for this installation is ASLM750.)

```
&hlq.ASLM750.F3(SLMALLOC)
&hlq.ASLM750.F3(SLMGZON)
&hlq.ASLMD750.F3(SLMDZON)
&hlq.ASLM750.F3(SLMTZON)
&hlq.ASLM750.F3(SLMDDEF)
&hlq.ASLM750.F3(SLMRECV)
&hlq.ASLM750.F3(SLMAPPLY).
```

- 5 Allocate the target and distribution libraries for the product. Sample JCL for this purpose is provided in the SLMALLOC member. You must tailor this job for the requirements of your installation before executing it. The following libraries will be created by the job:

Library	Description
target library &THLQUAL.SDFHLOAD	Dummy CICS data set for a user who does not have a CICS installation
target library &THLQUAL.SDFSRESL	Dummy IMS data set for a user who does not have an IMS installation
arget library &THLQUAL.SSLMAUTH	Sensor load modules
target library &THLQUAL.SSLMINCL	Sensor header files
target library &THLQUAL.SSLMINST	Sensor installation sample JCL
target library &THLQUAL.SSLMLOAD	Sensor load modules
target library &THLQUAL.SSLMPROC	Sensor sample JCL
target library &THLQUAL.SSLMSAMP	Sensor samples
target library &THLQUAL.SSLMSRC0	Sensor sample source file
distribution library &DHLQUAL.ASLMINCL	Sensor header files
distribution library &DHLQUAL.ASLMINST	Sensor installation sample JCL
distribution library &DHLQUAL.ASLMMOD	Sensor distribution modules
distribution library &DHLQUAL.ASLMPROC	Sensor sample JCL
distribution library &DHLQUAL.ASLMSAMP	Sensor samples
distribution library &DHLQUAL.ASLMSRC0	Sensor sample source file

- 6 If you are installing the Sensor into an existing SMP/E global zone, continue to step 7. If you are installing the Sensor into a new SMP/E global zone, use the sample JCL in the SLMGZON, SLMDZON, and SLMTZON members to create new global, target, and distribution zones. You must tailor these jobs for the requirements of your installation before executing them.

- 7 Create the SMP/E DDDEFs in the distribution and target zones. Sample JCL for this purpose is provided in the SLMDDDEF member. You may install into any SMP/E target zone desired. You must tailor this job for the requirements of your installation before executing it.

The following DDDEFs will be created by the job:

Library	Description
target zone DDDEF SSLMLOAD	Points to the SSLMLOAD target library allocated in Step 4.
target zone DDDEF SSLMINST	Points to the SSLMINST target library allocated in Step 4.
target zone DDDEF ASLMMOD	Points to the ASLMMOD distribution library allocated in Step 4.
target zone DDDEF SCEELKED	Points to the LE SCEELKED library. This DDDEF might already exist in your chosen target zone.
target zone DDDEF SCSQLOAD	Points to the WebSphere MQ SCSQLOAD library. This DDDEF might already exist in your chosen target zone.
target zone DDDEF SDFHLOAD	Points to the CICS SDFHLOAD library. This DDDEF might already exist in your chosen target zone.
target zone DDDEF SSLMINCL	Points to the SSLMINCL target library allocated in Step 4.
target zone DDDEF SSLMSRC0	Points to the SSLMSRC0 target library allocated in Step 4.
target zone DDDEF SSLMSAMP	Points to the SSLMSAMP target library allocated in Step 4.
target zone DDDEF SSLMPROC	Points to the SSLMPROC target library allocated in Step 4.
target zone DDDEF SSLMAUTH	Points to the SSLMQUTH target library allocated in Step 4.
distribution zone DDDEF ASLMMOD	Points to the ASLMMOD distribution library allocated in Step 4.
distribution zone DDDEF ASLMINST	Points to the ASLMINST distribution library created in Step 4.
distribution zone DDEF ASLMINCL	Points to the ASLMINCL distribution library created in Step 4.

Library	Description
distribution zone DDEF ASLMSRC0	Points to the ASLMSRC0 distribution library created in Step 4.
distribution zone DDEF ASLMSAMP	Points to the ASLMSAMP distribution library created in Step 4.
distribution zone DDEF ASLMPROC	Points to the ASLMPROC distribution library created in Step 4.



Important! If you do NOT have CICS installed on your system, customize and submit the JCL DUMYCICS to generate dummy CICS modules. Then set the CICS high level qualifier (&CICSQUAL) to be the same as the target library high level qualifier (%tvlib). If you do NOT have IMS installed on your system, customize and submit the JCL DUMYIMS to generate dummy IMS modules. Then set the IMS high level qualifier (&IMSQUAL) to be the same as the target library high level qualifier (%tvlib).

- 8 SMP/E RECEIVE the Sensor installation package. Sample JCL for this purpose can be found in the SLMRECV member. You must tailor this job for the requirements of your installation before executing it, changing the &DSNPREFIX and &GZONECSI parameters with your editor. Set the &DSNPREFIX parameter to account for the high level qualifier under which you created the ASLM750.* data sets. For example, if the data sets were installed in USERNAME.ASLM750.* and your global CSI data set was MY.GLOBAL.CSI, your customized JCL for the SMP/E RECEIVE might look as follows:

```
//RECEIVE EXEC PGM=GIMSMP,REGION=OM
//SMPCSI DD DSN=MY.GLOBAL.CSI,
//DISP=SHR
//SMPPTFIN DD DSN=USERNAME.ASLM750.SMPMCS,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SMPCNTL DD *
SET BDY(GLOBAL).
RECEIVE SELECT(ASLM750) RFPREFIX(USERNAME) SYSMODS LIST.
/*
```

- 9 SMP/E APPLY the Sensor installation package. Sample JCL for this purpose can be found in the SLMAPPLY member. You must tailor this job for the requirements of your installation before executing it. After the job is done verify the following:

Library	Contents
&THLQUAL.SSLMLOAD	CSQCAPX, MQCONN, SLMC, SLMCSTUB, SLMXSRV, BTMQEXIT, SLMBCNFG
&THLQUAL.SSLMINST	SLMACCPT, SLMALLOC, SLMAPPLY, SLMCICSD, SLMCRTQS, SLMDDDEF, SLMMDZON, SLMGZON, SLMRECV, SLMTZON, SLMCPYDC, SLMCPYDI, VERSION, SLMINSTL
&THLQUAL.SSLMINCL	BTMQEXIT, BTTRACE
&THLQUAL.SSLMPROC	SLMEJCL, SMLLKSTB, SLMTJCL, SLMTRJCL, TVISIONB, SLMBCFGQ, TVISIOND

Library	Contents
&THLQUAL.SSLMSAMP	SLMEXITS, SLMTSAMP
&THLQUAL.SSLMSRC0	SLMEXITD
&THLQUAL.SSLMAUTH	DFSYIOE0, SLMXCMD, SLMXCTL, SLMXINT, SLMXMON, SLMXRDQ, SLMXTKC, SLMXWRQ, SLMXXMC, SLMYIOE0

- 10 (CICS only) Update your CICS CSD file with the CSQCAPX and SLMC resource definitions for the Sensor. Sample JCL for this purpose can be found in the SLMCICSD member. You must tailor this job for the requirements of your installation before executing it.

This job step will define a program resource definition for CSQCAPX in your CICS region. Note that if your region already has a program resource defined for CSQCAPX and a CSQCAPX program is already installed then this previous program load module must be removed from your DFHRPL concatenation before the Sensor CSQCAPX module can be installed and used. This requirement results from the fixed naming mechanism used by WebSphere MQ for the API crossing exit facility for CICS/WebSphere MQ (only one crossing exit can be installed and it must be named CSQCAPX).

This job step will also define program and transaction definitions for SLMC.

- 11 (CICS only) Place the CSQCAPX and SLMC load modules in a library within your DFHRPL concatenation. You can either copy the modules into an existing library already present in your DFHRPL concatenation or you can add the SSLMLOAD library to the DFHRPL concatenation.
- 12 (CICS only) Enable the WebSphere MQ API crossing exit within your CICS region using the CKQC transaction. You can do this from the Connection > Modify > Enable API Exit menu item in the CKQC panel, or by specifying arguments to CKQC when invoking it. For example, you could enter CKQC MODIFY N E to enable the crossing exit (E for enable, D for disable).
- 13 Create appropriate Sensor configuration and event queues on the queue manager. Sample JCL for this purpose can be found in the SLMCRTQS member. You must tailor this job to the requirements of your installation before running it. See [Chapter 7, Configuring WebSphere MQ Sensors](#), for more information about Sensor configuration and event queues.
- 14 When you are satisfied with the results of your installation, run the SMP/E ACCEPT step. Sample JCL for this purpose can be found in the SLMACCPT member. You must tailor this job to the requirements of your installation before running it.
- 15 These sensors use EZASMI to retrieve the IP address. As a result, these startup procedures need an OMVS section that is RACF authorized, like TVISIOND.

Configuring SLMC for CICS

SLMC is an optional Sensor component that can improve the WebSphere MQ application performance of programs run in your CICS region when the Analyzer is not monitoring the region. SLMC monitors the TransactionVision configuration queue for messages from the Analyzer, examines any messages present there, and automatically enables or disables the WebSphere MQ crossing exit mechanism as needed by the Analyzer. Disabling the crossing

exit when the Analyzer is not actively monitoring the region (when no configuration messages are present or when configuration messages do not apply to the CICS region or host) improves application performance.

To use SLMC, you must configure your CICS region to run the SLMC transaction, which runs the SLMC program. Because SLMC uses the WebSphere MQ CSQCRST program, which must run with a terminal attached, to enable and disable the crossing exit, SLMC must also be run with a terminal attached. For ease of operations, it is recommended that you run SLMC using a sequential terminal, allowing it to be automatically started in the background when the CICS region is brought up. However, you can invoke it directly from an ordinary terminal if required.

A sequential terminal definition defines input and output data sets used by CICS to supply terminal input and output. The input data set contains the CICS transaction names you want to run (SLMC in this case) and the output data set receives the terminal output from the transaction(s). Creating a sequential terminal definition requires you to assemble CICS terminal resource macro definitions using DFHAUPLE, set the TCT parameter in your SIT to enable reading of the TCT in which your assembled macros are placed, and add DD names to your CICS region's startup JCL to define the input and output data sets for the terminal.

To create a sequential terminal to run SLMC, use the sample sequential terminal definitions supplied by IBM in the CICS SDFHSAMP members DFHTCT5\$ and DFH\$TCTS. For complete details on this sample and how to set up a sequential terminal, consult the CICS System Definition Guide and CICS Resource Definition Guide for your version of CICS.

To run the SLMC transaction with your sequential terminal, the sequential terminal input data set (for example, the CARDIN DDNAME in the IBM sequential terminal sample) should contain the following two lines:

```
SLMC\  
CESF LOGOFF\  

```



The \ character is the standard CICS end-of-data character. If you have redefined the end-of-data character on your EODI system initialization parameter, specify your end-of-data character instead.

SLMC issues messages describing its operation to the system log. These messages are described in the following table. In each of the message descriptions, *nnn* represents the CICS task number of the running SLMC transaction.

Message	Description
SLMS300I CICS SLMC <i>nnn</i> : TransactionVision sensor: Management Initiated for WebSphere MQ API Crossing Exit	Issued when SLMC begins execution
SLMS202I CICS SLMC <i>nnn</i> : TransactionVision sensor: Enabling WebSphere MQ API Crossing Exit	Issued when SLMC enables the crossing exit
SLMS201I CICS SLMC <i>nnn</i> : TransactionVision sensor: Disabling WebSphere MQ API Crossing Exit	Issued when disabling the crossing exit
SLMS203I CICS SLMC <i>nnn</i> : TransactionVision sensor: SLMC Exiting	Issued after a diagnostic error if the error causes SLMC to exit
SLMS117E CICS SLMC <i>nnn</i> : TransactionVision sensor: <i>diagnostic message</i>	Issued when an error is encountered. The diagnostic message describes the error

Additional Setup for the WebSphere MQ IMS Bridge Sensor

The WebSphere MQ IMS Bridge Sensor is implemented as three components:

- 1 An OTMA Input/Output Edit exit routine, DFSYIOE0, which runs in the IMS control region.
- 2 A control function, referred to as TVISIONB, which runs as a started task in a separate address space.
- 3 An event dispatcher function, referred to as TVISIOND, which runs as a started task in another separate address space.

These three components communicate via cross memory program calls, which requires some system setup considerations:

- 1 TVISIONB requires one system linkage index (LX), which it reserves the first time it is started after an IPL and thereafter reuses. Refer to the IBM z/OS or z/OS MVS Initialization and Tuning Reference, IEASYSxx (System Parameter List) NSYSLX=nnn parameter.
- 2 The TransactionVision library, thlqual.SSLMAUTH, must be APF-authorized. Note that thlqual.SSLMLOAD must not be authorized. Refer to MVS Initialization and Tuning Reference, PROGxx or IEAAPFxx system parameters.
- 3 The TVISIONB address space must be non-swappable. TVISIONB assures this by issuing a SYSEVENT TRANSWAP macro, so it is not necessary to include the program in the PPT. If you do include it, specify the entry as follows:

```
PPT PGMNAME(SLMXCTL) CANCEL KEY(8) NOSWAP NOPRIV
NODSI PASS SYST AFF(NONE) NOPREF
```

Refer to *MVS Initialization and Tuning Reference*, SCHEDxx system parameters.

For instructions on completing Sensor setup and operating the WebSphere MQ IMS Bridge Sensor, see [Chapter 7, Configuring WebSphere MQ Sensors](#).

7 Configuring WebSphere MQ Sensors

TransactionVision provides two types of WebSphere MQ Sensors:

- WebSphere MQ Sensor library
- WebSphere MQ API Exit

Configuring the WebSphere MQ Sensor Library

The WebSphere MQ Sensor library is dynamically loaded at runtime by including its library search path before the standard WebSphere MQ library search path. The standard WebSphere MQ library is dynamically loaded by the Sensor library. Before you can use TransactionVision to record event data for an application, you must configure the application environment to load the Sensor library instead of the standard WebSphere MQ library.

Important! When using the Sensor Library with 64-bit applications, including 64-bit Java, there may be library conflicts between the WebSphere MQ 32-bit libraries and the 64-bit libraries. See the “Implications of a 64-bit queue manager” section in the *WebSphere MQ Quick Beginnings* book to resolve any problems seen when trying to use the Sensor Library for 64-bit applications.

Distributed Platforms

Distributed platforms include all platforms other than z/OS and i5/OS. On distributed platforms, you must add the directory location of the Sensor library to an environment variable setting on the computer where the monitored application runs before starting the application.

The following table shows the appropriate environment variable and directory location to set for each platform for if you are using 32-bit applications. If a path including the standard WebSphere MQ library exists as part of the environment value, the Sensor entry must appear before it in order to use TransactionVision.

Platform	Environment Variable	Default Directory
Windows	PATH	C:\Program Files\Hewlett-Packard\TransactionVision\lib
Sun Solaris	LD_LIBRARY_PATH	/opt/HP/TransactionVision/lib
HP-UX	SHLIB_PATH	/opt//HP/TransactionVision/lib
IBM AIX	LIBPATH	/usr/lpp/HP/TransactionVision/lib
RedHat Linux	LD_LIBRARY_PATH	/opt/HP/TransactionVision/lib

All of the directory locations in this table are the default Sensor installation locations. If the Sensor was installed in a location other than the default, specify the directory location for the Sensor executable.

When using 64-bit applications, set the library paths as shown in the following table:

Platform	Environment Variable	Default Directory
Sun Solaris	LD_LIBRARY_PATH	/opt/HP/TransactionVision/lib64:/opt/mqm/lib64
HP-UX	SHLIB_PATH	/opt/HP/TransactionVision/lib64:/opt/mqm/lib64
IBM AIX	LIBPATH	/usr/lpp/HP/TransactionVision/lib64:/usr/lpp/mqm/lib64
RedHat Linux x86-64	LD_LIBRARY_PATH	/opt/HP/TransactionVision/lib64:/opt/mqm/lib64

On the AIX platform, you must run `/usr/sbin/slibclean` to clear the original shared library from memory before you can pick up a new library that has the same name as an existing library.

On the HP-UX platform, you may have to use the `chatr` command to enable your application to pick up the Sensor library if the use of the `SHLIB_PATH` environment variable is not enabled or is not the first in the dynamic library search path for your application. You may use the `chatr` command to check the current settings of your application. In any case, make sure that `SHLIB_PATH` is enabled and is before the standard WebSphere MQ library path. See Chapter 10 in the *TransactionVision Administration Guide* for details.

Important! When using multithreaded applications with WebSphere MQ on UNIX systems, ensure that the applications have sufficient stack size for the threads. IBM recommends using a stack size of at least 256KB when multithreaded applications are making MQI calls. When using the TransactionVision WebSphere MQ Sensor, even more stack size may be required; the recommended stack size is at least 512KB. For more information, see Chapter 7, “Connecting to and disconnecting from a queue manager,” in the *WebSphere MQ Application Programming Guide*.

To run applications to be monitored by Sensors without setting the library environment globally, run the `wmqsensor` script provided with TransactionVision as follows:

On UNIX platforms:

```
installation_directory/wmqsensor application_command_line
```

On Windows platforms:

```
installation_directory\wmqsensor application_command_line
```

For example, if you run your WMQ application as `amqsput...`, you will use

```
installation_directory\wmqsensor amqsput...
```

z/OS Batch, IMS, and WebSphere MQ-IMS Bridge

On the z/OS batch and IMS platforms, the `SLMLKSTB` member of the sample procedure library `thlqual.SSLMPROC` contains a sample job to bind the Sensor to an WebSphere MQ application program. The WebSphere MQ batch stub section—`CSQBSTUB`, `CSQBRRSI`, or `CSQBRSTB` for Batch or `CSQQSTUB` for IMS—is replaced in the application load module

with the corresponding Sensor batch or IMS stub—SLMBSTUB, SLMBRRSI, SLMBRSTB, or SLMQSTUB. After this bind, the application will invoke the Sensor instead of WebSphere MQ directly.

Note that thlqual is the high-level qualifier chosen by your System Administrator when installing TransactionVision. For example, if the high-level qualifier is TVISION, the sample procedure library is TVISION.SSLMPROC.



Note that if your current applications use the z/OS Resource Recovery Services (RRS) in batch, the calls to SRRCMIT and SRRBACK are not recorded by the Sensor but are simply passed through to WebSphere MQ.

When running the application, make sure that the library containing the Sensor is specified in the LNKLST, STEPLIB, or JOBLIB concatenation. In UNIX System Services, define environment variable STEPLIB to specify the library containing the Sensor.

z/OS CICS

On the z/OS CICS platform, Sensors use the API-crossing exit mechanism provided by the CICS adapter of WebSphere MQ for z/OS.

i5/OS

On the i5/OS platform, the main Sensor service program has the same name as the WebSphere MQ service program: LIBMQM for non-threaded programs and LIBMQM_R for threaded programs. The two TransactionVision service programs have the same signature and exported symbols (in the same order) as their WebSphere MQ counterparts.

TransactionVision also provides two utility service programs:

- MQMUTL5 binds to QMQM/LIBMQM
- MQMUTL5_R binds to QMQM/LIBMQM_R

The main Sensor service program binds to one of these three service programs so a program's MQI call will be passed into the WebSphere MQ service program.

Use one of the following methods to use the Sensor service program:

- User program is created by CRTPGM with the parameter "BNDSRVPGM(QMQM/LIBMQM)" or "BNDSRVPGM(QMQM/LIBMQM_R)" or "BNDSRVPGM(QMQM/AMQZSTUB)".

Specify the "ALWUPD(*YES)" and "ALWLIBUPD(*YES)" parameters to CRTPGM. The default values are *YES for ALWUPD and *NO for ALWLIBUPD. If the user program is created without these parameters, rebind it by either method.

After the program binding, you can use UPDPGM to switch between the service programs provided by WebSphere MQ and the Sensor. The parameter for this command is SRVPGMLIB, which you can set to either QMQM or TVSENSOR.

- User program is created by CRTPGM with the parameter "BNDSRVPGM(*LIBL/LIBMQM)" or "BNDSRVPGM(*LIBL/LIBMQM_R)".

After the binding of the program, use ADDLIB or CHGLIB to switch between the WebSphere MQ library QMQM and the Sensor library TVSENSOR. The Sensor library must precede the WebSphere MQ library QMQM in the library list in order to use TransactionVision.

► Note that an RPG program created by the ILE RPG compiler uses the same WebSphere MQ service program as the C program created by the ILE C compiler. TransactionVision has been tested in the following scenarios using MQI in an ILE RPG program.

- Using MQI through a call to MQM
- Using prototyped calls to the MQI

Configuring Sensor Logging

On some operating systems, there is no additional work to obtain error and trace logging from the WebSphere MQ Sensors. However, on UNIX platforms, syslogd may need to be configured to log the logging facility used by the WebSphere MQ Sensors. Refer to [Chapter 9, Configuring Sensor Logging](#), for details on WebSphere MQ Sensor logging on UNIX platforms.

Setting the Configuration Queue Name

By default, Sensors look for a configuration queue named TVISION.CONFIGURATION.QUEUE on the queue manager specified in the WebSphere MQ API call. However, you may specify a different configuration queue name when you create a communication link. If you are using a communication link that specifies a non-default configuration queue name, you must configure Sensors to look for configuration messages on that queue instead of TVISION.CONFIGURATION.QUEUE.

UNIX, Windows, and i5/OS

On UNIX, Windows, and i5/OS platforms, set the TVISION_CONFIGURATION_QUEUE environment variable to the Sensor configuration queue specified in the communication link for all processes that use the Sensor.

IBM z/OS Batch, IMS and WebSphere MQ-IMS Bridge

On IBM z/OS Batch, IMS, and WebSphere-IMS bridge, a user-installable load module named SLMBCNFG can be generated to control which queue the Sensor reads to retrieve configuration messages from the Analyzer.

- 1 Edit thlqual.SSLMPROC(SLMBCFGQ) and change as follows:
 - a Change the value of the SET CONFIGQ statement to specify the desired configuration queue name.
 - b Change the SET THLQUAL statement to specify the high-level qualifier for your TransactionVision Sensor libraries.
 - c If your system does not have the IBM-supplied HLASMCL procedure available, change the JCL as necessary to assemble and bind the included source code. Please do not change any of the source code.
- 2 Submit thlqual.SSLMPROC(SLMBCFGQ) to effect the change.

For more information about the WebSphere MQ-IMS bridge Sensor, see [Using the WebSphere MQ-IMS Bridge Sensor](#) on page 99.

On IBM z/OS CICS, a user-installable program named SLMCNFQ can be written to control which queue the Sensors read from to retrieve configuration messages from the Analyzer.

If the program SLMCNFQ can be executed by the Sensor via an EXEC CICS LINK, the Sensor does so, passing SLMCNFQ a CICS comm area large enough to hold a configuration queue name (48 bytes). The comm area initially contains the default configuration queue name (TVISION.CONFIGURATION.QUEUE). When executed by the Sensor, SLMCNFQ should write the desired configuration queue name to the comm area passed to it, and then return. Subsequently, the Sensor will read the comm area to retrieve the correct configuration queue name.

► Note that the installation procedure for the z/OS CICS Sensor does NOT create an SLMCNFQ program. For instructions on writing this program, see [Writing SLMCNFQ](#), which follows.

If the attempt to execute the SLMCNFQ fails, the z/OS CICS WebSphere MQ Sensor tries to load the program SLMBCNFG and gets the configuration queue name. For instructions on generating this program, see [Generating SLMBCNFG](#) on page 89.

If the attempt to load SLMBCNFG fails, the Sensor reads from TVISION.CONFIGURATION.QUEUE.

► To use a different configuration queue name for each CICS region, see [Using Separate Configuration Queues for Each CICS Region](#) on page 90.

Writing SLMCNFQ

You can write an SLMCNFQ program in any language supported by your CICS region. The following is an example written in C that sets the configuration queue name to MY.CONFIGURATION.QUEUE:

```
#include <cmqc.h>
#include <string.h>
#include <stdlib.h>
int main(int argc, char * argv[])
{
    void *pCommArea = NULL;
    EXEC CICS ADDRESS COMMAREA(pCommArea) EIB(dfheiptr);
    if (pCommArea && (dfheiptr->eibcalen >= sizeof(MQCHAR48)))
    {
        memset(pCommArea, 0, sizeof(MQCHAR48));
        strcpy(pCommArea, "MY.CONFIGURATION.QUEUE");
    }
    EXEC CICS RETURN;
}
```

Generating SLMBCNFG

SLMBCNFG is generated the same way as for z/OS batch and IMS; that is:

- 1 Edit thlqual.SSLMPROC(SLMBCFGQ) and change as follows:
 - a Change the value of the SET CONFIGQ statement to specify the desired configuration queue name.
 - b Change the SET THLQUAL statement to specify the high-level qualifier for your TransactionVision Sensor libraries.

- c If your system does not have the IBM-supplied HLASMCL procedure available, change the JCL as necessary to assemble and bind the included source code. Please do not change any of the source code.
- 2 Submit thlqual.SSLMPROC(SLMBCFGQ) to effect the change.

Using Separate Configuration Queues for Each CICS Region

If you have multiple CICS regions, you may want to use a different configuration queue name for each CICS region while sharing the Sensor library among those CICS regions. To achieve this, perform the following steps:

- 1 If you have not added the table SLMBCNFG to your CSD definition, please do so. The definition of SLMBCNFG is shown below:

```
DEFINE PROGRAM(SLMBCNFG) GROUP(BTITV)
  DESCRIPTION(TVISION SENSOR CONFIGQ TABLE)
  LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
  USELPACOPY(NO) STATUS(ENABLED) DATALOCATION(ANY)
```

- 2 Create a new member called CONFIGQ in &TVISION.SSLMSAMP. The contents of this member, which is extracted from the supplied sample TVISION.SSLMPROC(SLMBCFGQ), are as follows:

```
.*-----
.* SLMCONFG - TVision configuration macro - PLEASE DO NOT CHANGE
.*-----
      MACRO
      SLMCONFG &CONFIGQ=TVISION.CONFIGURATION.QUEUE
SLMBCNFG      AMODE 31
SLMBCNFG      RMODE ANY
SLMBCNFG      SECT
CONFIGQ       DC    CL48 '&CONFIGQ'
      MEND
*
      SLMCONFG CONFIGQ=&SYSPARM
      END
```

- 3 In your CICS startup JCL, make the following updates:

In the PROC statement of DFHSTART, add one parameter, CONFIGQ. For example:

```
//DFHSTART, PROC, START=AUTO,
// INDEX1='CICSTS22',
...
// SIP=0,
// CONFIGQ='TVISION.CONFIGURATION.QUEUE',
...
```

Add an additional step before the actual CICS execution step:

```
//*-----
// EXEC HLASMCL,
//   PARM.C='NORLD,NOXREF,NORXREF,SYSPARM(&CONFIGQ)',
//   PARM.L='MAP,REFR'
//C.SYSIN DD DISP=SHR,DSN=TVISION.SSLMSAMP(CONFIGQ)
//L.SYSLMOD DD
DSN=&&CONFIGQ(SLMBCNFG),DISP=(,PASS),SPACE=(TRK,(1,1))
//*-----
```

Add a DD statement referencing the temporary PDS created above to DFHRPL concatenation preceding the TVISION library. For example:

```
...  
//      DD  DISP=SHR, DSN=&&CONFIGQ  
//      DD  DISP=SHR, DSN=TVISION.SSLMLOAD  
...
```

- 4 After you have made these modifications, you can start each CICS region with different configuration queue name by simply passing a unique CONFIGQ parameter. For example:

```
S CICS.CICS1, START=COLD, . . . , CONFIGQ= 'TV.CONFIG.Q1 '  
S CICS.CICS2, START=COLD, . . . , CONFIGQ= 'TV.CONFIG.Q2 '
```

Setting the Configuration Queue Check Interval

By default, Sensors check the configuration queue for new configuration messages every five seconds. On UNIX, Windows, and i5/OS platforms, however, you may specify a different configuration queue check interval. To specify a non-default configuration queue check interval for a Sensor, set the TVISION_CONFIG_CHECK_INTERVAL environment variable to the desired interval, in milliseconds.

Configuring the WebSphere MQ Messaging System Provider

TransactionVision uses queues for the communication between the Analyzers and the Sensors. You need at least three queues: the configuration queue, the event queue, and the exception queue. The default name of these queues are TVISION.CONFIGURATION.QUEUE, TVISION.EVENT.QUEUE, and TVISION.EXCEPTION.QUEUE, respectively. You must set up these queues on your message system provider in a vendor-specific way. See Chapter 5 “Managing Communication Links” in the *TransactionVision Administration Guide* for details.

You may create the queues on your queue managers using the IBM WebSphere MQ runmqsc utility program or the MQ Explorer graphical user interface that is available on Windows and Linux. For using the client connection type, you also need to define a server connection channel and a listener on your queue manager. See the vendor's documentation for details.

Configuring the WebSphere MQ API Exit Sensor

The WebSphere MQ API Exit Sensor is an exit program which examines all MQI calls made with respect to the associated queue manager. The exit program registers functions to be invoked before and after an MQI call. It is implemented in the following shared objects/DLLs:

- tvisionapiexit
- tvisionapiexit_r

Though the Sensor is registered with respect to queue managers, it is actually loaded and executed within the address space of the application making the MQI calls. For example, the Sensor is running in the amqsput program address space, not the queue manager space.

You can use the WebSphere MQ API Exit Sensor to monitor any WebSphere MQ server applications. You can monitor client applications indirectly by collecting the corresponding MQI calls in the server connection channel agents (listeners).

The WebSphere MQ API Exit Sensor differs from the WebSphere MQ Sensor Library in the following ways:

- There is no need to disable FASTPATH_BINDING (see [WebSphere MQ Sensors and FASTPATH_BINDING](#) on page 96 chapter for more information).
- The completion and reason codes for MQDISC calls are not collected by the API Exit Sensor and are set to fixed values of MQCC_OK and MQRC_NONE, respectively. The event time for MQDISC events is set to the before-MQDISC function invocation time.
- The API Exit Sensor collects some TransactionVision internal events generated from WebSphere MQ (WMQ) calls made by the Analyzer, and also internal WMQ events from Java Sensors using a client connection to the listener.



If the API Exit Sensor and WebSphere MQ Sensor Library are active at the same time, the API Exit Sensor will log a warning and not register the MQI exits, staying inactive. The WebSphere MQ Sensor Library will then continue to process events.

Configuring the API Exit Sensor on Distributed and i5/OS Platforms

To use the WebSphere MQ API Exit Sensor on distributed platforms or i5/OS, you must perform the following steps:

- 1 Link the appropriate WebSphere MQ API Exit Sensor shared object/DLL for your environment (distributed platforms only).
- 2 Add the required stanzas to the mqs.ini and qm.ini files.

Linking the WebSphere MQ API Exit Sensor

Because TransactionVision supports both 32-bit and 64-bit versions of WebSphere MQ, you must link to the correct shared object/DLL.

For WebSphere MQ V5.3 and V6.0 (32-bit), use the following commands to link the Sensor:

```
ln -s <TransactionVision Install Directory>/lib/tvisionapiexit /var/mqm/exits/tvisionapiexit  
  
ln -s <TransactionVision Install Directory>/lib/tvisionapiexit_r /var/mqm/exits/tvisionapiexit_r (not required for Solaris)
```

For WebSphere MQ V6.0 (64-bit), use the following commands to link the Sensor:

```
ln -s <TransactionVision Install Directory>/lib64/tvisionapiexit /var/mqm/exits64/tvisionapiexit  
  
ln -s <TransactionVision Install Directory>/lib64/tvisionapiexit_r /var/mqm/exits64/tvisionapiexit_r (not required for Solaris)
```

Ensure that the following stanza is in the qm.ini file:

```
ExitPath:  
  ExitsDefaultPath=/var/mqm/exits/  
  ExitsDefaultPath64=/var/mqm/exits64
```

Note that if ExitsDefaultPath parameter value and/or ExitsDefaultPath64 values of the ExitPath: stanza in qm.ini file are changed, you must change the directory name /var/mqm/exits and/or /var/mqm/exits64 described in the link commands appropriately.

New Stanzas

You must define the API Exit Sensor in new stanzas in the `mqs.ini` file, which contains definitions applied to the whole WebSphere MQ environment, and the `qm.ini` file, which applies to individual queue managers. The `mqs.ini` file is typically located in the directory `/var/mqm`. The `qm.ini` file is typically in the directory `/var/mqm/qmgrs/<qmgr_name>`. A stanza consists of a section header followed by a colon, which is then followed by lines containing attribute/value pairs separated by the "=" character. Note that the same attributes may be used in either `mqs.ini` or `qm.ini`.

Add the following stanzas to `mqs.ini`:

- `ApiExitCommon`

The attributes in this stanza are read when any queue manager starts, then overwritten by the API exits defined in `qm.ini`.

- `ApiExitTemplate`

When any queue manager is created, the attributes in this stanza are copied into the newly created `qm.ini` file under the `ApiExitLocal` stanza.

Add the following stanza to `qm.ini`:

- `ApiExitLocal`

When the queue manager starts, API exits defined here override the defaults defined in `mqs.ini`.

Stanza Attributes and Values

All of these required stanzas have the following attributes and values:

- `Name=TransactionVisionWMQSensor`

The descriptive name of the API exit passed to it in the `ExitInfoName` field of the `MQAXP` structure. This attribute should be set to the string "TransactionVisionWMQSensor".

- `Function=TVisionEntryPoint`

The name of the function entry point into the module containing the API exit code. This entry point is the `MQ_INIT_EXIT` function. This attribute should be set to the string "TVisionEntryPoint".

- `Module=tvisionapiexit`

The module containing the API exit code. Set this attribute to the `TransactionVision` WebSphere MQ API Exit Sensor binary. For platforms that support separate threaded libraries (AIX, HP-UX, and Linux), this is set to the path for the non-threaded version of the Sensor module. The threaded version of the WebSphere MQ application stub implicitly appends `_r` to the given module name before it is loaded.



Important! Do not specify a path; the module path is determined by the `link` command you used to link to the correct module (see [Linking the WebSphere MQ API Exit Sensor](#) on page 92). The location depends on whether you use 32-bit or 64-bit WebSphere MQ libraries. The 32-bit module is located in `<TransactionVision installation directory>/lib`, while the 64-bit module is located in `<TransactionVision installation directory>/lib64`.

- `Data=TVQ=queue_name`

To set the queue object names for which the Sensor should ignore WMQ events on, set the TVQ attribute to the object name or part of the object name with wildcards. If no Data section is specified, events on objects matching TVISION* will be ignored by the Sensor. To completely turn off this feature specify an empty string for TVQ (Data=TVQ=).

The data field can have a maximum of 24 characters; therefore, the TVQ object name value may be up to 20 characters and may include the * wildcard character at the beginning and/or end of the string. Only one queue string may be specified for the TVQ attribute. For more information, see [Discarding WebSphere MQ Events on TransactionVision Queues](#) on page 95.

- Sequence=sequence_number

The sequence in which the TransactionVision WebSphere MQ API Exit Sensor module is called relative to other API exits. An exit with a low sequence number is called before an exit with a higher sequence number. There is no need for the sequence number of exits to be contiguous; a sequence of 1, 2, 3 has the same result as a sequence of 7, 42, 1096. If two exits have the same sequence number, the queue manager decides which one to call first. This attribute is an unsigned numeric value.

The following is an example illustrating the Sensor configuration per queue manager (qm.ini).

```
ApiExitLocal:
  Name=TransactionVisionWMQSensor
  Sequence=100
  Function=TVisionEntryPoint
  Module=tvisionapiexit
```

Configuring the API Exit Sensor on Windows Platforms

Configure the WebSphere MQ API Exit Sensor on Windows platforms using the WebSphere MQ Services snap-in or the **amqmdain** command to update the Windows Registry.

A new property page for the WebSphere MQ Services node, API Exits, describes the two types of API exit managed from this node: ApiExitCommon and ApiExitTemplate. In the Exits property page for individual queue managers, you can update the ApiExitLocal. The Configure... buttons launch a dialog to manage the entries within each stanza. The dialog consists of a multi-column list of any API exits already defined in the appropriate stanza, with buttons to add, view, change the properties of, and remove exits. See [Configuring the API Exit Sensor on Distributed and i5/OS Platforms](#) on page 92 for a description of required stanzas and attribute values.

When you finish defining or changing an exit, press OK to update the Registry, or press Cancel to discard changes.

Identifying Programs to Monitor

The WebSphere MQ API Exit Sensor uses two files to identify which programs to monitor:

- exit_sensor.allow defines which programs will be monitored. All other programs are NOT monitored. Note that if this file is empty, no programs will be monitored. On i5/OS, this file name is ALLOW.MBR.
- exit_sensor.deny defines which programs will not be monitored. All other programs will be monitored. On i5/OS, this file name is DENY.MBR. This file is shipped with the WebSphere MQ Sensor and contains some default programs from which events are not collected by the API Exit Sensor, such as the WebSphere MQ command server.

These files are located at the top level TransactionVision installation directory. For example, on Solaris if TransactionVision is installed at /opt/HP/TransactionVision, these two files exist in the /opt/HP/TransactionVision directory. On i5/OS, these files are in /qsys.lib/tvsensor.lib/EXITSENSOR.FILE.

In these files, comment lines begin with a # character. You may use the * wildcard character at the beginning and/or end of program names.

If both exit_sensor.allow and exit_sensor.deny exist, the Sensor ignores exit_sensor.deny.

Most WebSphere MQ commands (programs) are denied, and the API exit is not registered for them. Additional programs can be denied by the user by specifying the names in exit_sensor.deny.

The following is an example exit_sensor.allow file, which will only collect from WebSphere MQ listeners:

```
# File: exit_sensor.allow
# Description: Only collect from WebSphere MQ Listeners
amqcrsta
amqrmppa
runmqtsr
```

The following is an example exit_sensor.deny file to collect any program except for those that start with amq:

```
# File: exit_sensor.deny
# Description: Collect any program except those that
# start with "amq"
amq*
```

Discarding WebSphere MQ Events on TransactionVision Queues

By default, the Sensor discards any WebSphere MQ traffic related to any queue object with the name prefix "TVISION." To specify a different queue object name, set TVQ to the object name string in the Data attribute. Use the * wildcard character to indicate where in the object name the specified characters occur, as in the following examples:

- HP_TV*
"HP_TV" is the prefix for all TransactionVision queue objects.
- *HP_TV
"HP_TV" is the suffix for all TransactionVision queue objects.
- *HP_TV*
All TransactionVision queue objects contain the string "HP_TV."



Note that wildcards may be used before and/or after the TVQ value, but not within it. For example, a value of T*VISION is invalid.

If you require finer control over which queue objects to track, use a data collection filter instead. For instructions on using data collection filters, see the "Managing Data Collection Filters" chapter of the *TransactionVision Administration Guide*.

WebSphere MQ Sensors and FASTPATH_BINDING

For the standard WebSphere MQ Library Sensor on distributed platforms, if FASTPATH_BINDING is set for the monitored application, it binds the application to the same address space as the queue manager and tries to load a secondary DLL that is linked against the standard WebSphere MQ library. Since this environment is configured to load the Sensor library instead of WebSphere MQ, the secondary DLL tries to call internal symbols that are not available.

To work around this potential problem, Sensors disable all FASTPATH_BINDING by setting the MQ_CONNECT_TYPE environment variable to STANDARD whenever the monitored application calls MQCONN.

Using Sensors with WebSphere MQ Samples

If you want to use Sensors to monitor WebSphere MQ sample applications, note the following limitations:

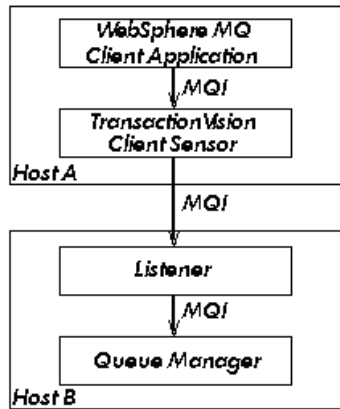
- On Windows, there are two locations for WebSphere MQ samples. If you run the samples under WebSphere MQ\bin, you must copy the sample executables (amqsput, amqsget, and so on) to a different directory to enable them to load the Sensor library instead of the standard WebSphere MQ library. This is because the WMQ libraries reside in this same folder and take precedence over the Sensor libraries even if PATH is set properly. The samples under WebSphere MQ\TOOLS\c\samples\bin do not show this problem.
- On the HP-UX and Linux platforms, the sample executables have a hard-coded WebSphere MQ library path and therefore will not load the Sensor library.
- When using the WebSphere MQ sample amqsgbr, do not use the Sensor event queue as the first parameter.

WebSphere MQ Client Application Monitoring

For applications using WebSphere MQ client bindings, TransactionVision is capable of monitoring and tracing these applications' messaging activities in either a distributed or centralized mode.

Distributed Monitoring

The following diagram illustrates how TransactionVision works in a distributed monitoring environment:



In general, applications that make use of WebSphere MQ client binding will communicate with a “listener” process (also known as the channel responder) that runs on the same host as the targeted queue manager. All WebSphere MQ activities (that is, MQI calls) are forwarded to and processed by the listener program, which in turn issues the appropriate MQI calls to the corresponding queue managers on behalf of the client applications.

In the distributed monitoring mode, an instance of the TransactionVision client Sensor will be installed on the same host where the client application runs. The Sensor will intercept and monitor the MQI calls made by the client application, generate trace information accordingly, and invoke the corresponding MQI entry points in the regular WebSphere MQ client binding.

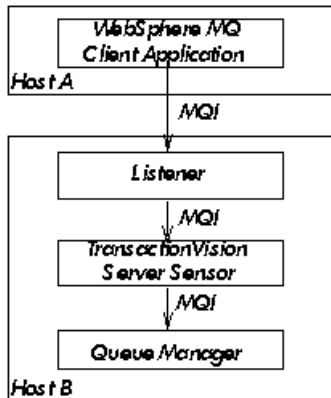
The trace information generated will be based on the client application context. This means information such as program name, program instance, and host name, will be related to the client application directly.

This monitoring scheme requires a client Sensor installed on each machine where WebSphere MQ client applications run. Moreover, the client Sensor is capable of monitoring any applications making use of the C language WebSphere MQ client runtime binding. In other words, the client Sensor supports applications developed in C and C++. On the other hand, WebSphere MQ Java Client class does not make use of the C runtime binding. Thus this approach is not applicable to WebSphere MQ Java client applications or applets. (Note that WebSphere MQ Java Server applications are indeed supported through the C language TransactionVision Server Sensor).

This approach is supported for client applications running on Windows, Solaris, HP-UX, AIX, and Linux operating systems.

Centralized Monitoring

Centralized monitoring of the WebSphere MQ listener program is only supported using the API Exit Sensor and is not supported with the library sensor. The following diagram illustrates how the Sensor works in a centralized monitoring environment:



In this case, the Sensor is deployed on the host where the listener process and queue manager reside. Instead of direct monitoring of the client application, the Sensor monitors the listener program instead. Note that the TransactionVision Server Sensor is deployed. The Sensor intercepts and reports any MQI calls issued by the listener program. In other words, the listener program will execute the same MQI calls that the client application invokes (with a few exceptions, as we will discuss later). Therefore, by examining the listener program MQI call records, TransactionVision users can have a good understanding of the messaging activities originated from the client applications.

One advantage of this approach is that Sensor deployment can be centralized around the host machines where the queue manager runs. Unlike the distributed approach, no client Sensors are needed on the client machines. This can greatly reduce the installation and administration efforts in environment where client applications may run on thousands of machines distributed in different physical facilities.

Another note is that this approach can support WebSphere MQ Java client application/applet monitoring. Such monitoring is not supported in the distributed mode.

If you decide to deploy this model of monitoring, take note of the following:

- Since the Sensor monitors the listener program instead of client applications directly, certain context information reported such as program name, program instance identifiers, host name, and so on, correspond to the listener program instead. However, since each client connection is handled in a particular thread or process instance of the listener program, MQI calls from the same client application and same connection will be listed under the same listener program instance.
- The listener program will not invoke the MQCONN or MQCONNX calls on client connection requests. Thus there will be no corresponding TransactionVision trace information reported for such connection events.
- The listener program may make additional MQI calls on its behalf. For example, when processing a new connection, it will make a MQOPEN-MQINQ-MQCLOSE call sequence for querying queue manager information. Also, it will make a MQCMIT-MQBACK call sequence when processing a disconnection request from the client.
- There is a one-to-one correspondence between the MQPUT/MQPUT1/MQGET calls from the client applications and listener program. So the listener messaging activities should accurately reflect those of the clients it serves.
- As discussed before, context information reported is associated with the listener program. However, client application origin context information can be retrieved indirectly through the message header (MQMD) structure embedded in the MQPUT/MQPUT1/MQGET feedback data through the call parameters.

- If you use the Servlet, JMS, or EJB Sensors with a client connection to a queue manager through a listener, which is being monitored with the TransactionVision WebSphere MQ Sensor, internal TransactionVision events will be captured. It is recommended to either use a separate unmonitored listener for the Servlet, JMS, or EJB Sensors or use server binding connections from these Sensors. If this cannot be done, change the data collection filter to exclude the configuration and event queues.

The table below summarizes the characteristics of the two approaches:

Distributed Monitoring	Centralized Monitoring
Direct client MQI tracing	Indirect tracing through listener MQI calls
Direct client context information access	Client context information through call parameters
Client Sensor on each client host	Server Sensor per queue manager host
Monitors applications using WebSphere MQ C binding	Server Sensor per queue manager host
Supports client applications running on Windows, Solaris, HP-UX, AIX, and Linux	Support clients connecting to queue managers running on Windows, Solaris, HP-UX, and AIX

Installation and Configuration Considerations

For distributed monitoring, install client Sensors on each host where WebSphere MQ client applications run. Follow the standard Sensor installation instructions in [Installing and Configuring the Java Agent on Windows](#) on page 16.

For centralized monitoring, install server Sensors on each host where one or more listener programs are to be monitored.



The centralized monitoring mechanism is exclusive with the distributed monitoring mechanism. In general, the server Sensor monitoring the listener program will report activities originated from all clients it services, including those clients that may be monitored by client Sensors residing on the client host. In order to avoid redundant reports, if you choose the centralized monitoring approach, make sure that on every host where clients are connecting to the queue manager whose listener is being monitored, the client Sensor is disabled.

Using the WebSphere MQ-IMS Bridge Sensor

The WebSphere MQ-IMS bridge is a WebSphere MQ component that enables WebSphere MQ applications to invoke IMS transactions and receive their reply messages. The application performs an MQPUT to an WebSphere MQ-IMS bridge input queue with a message consisting of an IMS transaction code followed by transaction data and receives the IMS output message by performing an MQGET to the reply-to queue specified in the message descriptor on the MQPUT. The IMS transaction does not need to change to accommodate this interface.

The TransactionVision WebSphere MQ-IMS bridge Sensor monitors WebSphere MQ-IMS bridge messages rather than the WebSphere MQ API calls made by the calling applications.

Sensor Setup

Before using the WebSphere MQ-IMS bridge Sensor, perform the following setup tasks:

- 1 Customize the sample TVISIONB startup procedure in thlqual.SSLMPROC and copy it to an appropriate PROCLIB. TVISIONB requires four startup parameters, which may be specified in the procedure or on the START command.
 - The QMGR parameter specifies the name of the WebSphere MQ queue manager to which TVISIONB must connect to access its configuration and event queues. Note that this queue manager is the one to which the Analyzer connects when establishing a communication link to the Sensor and not necessarily the queue manager(s) to which the WebSphere MQ-IMS bridge is connected. It must be the same queue manager used when defining the configuration and event queues during installation (see the sample job in thlqual.SSLMINST(SLMCRTQS)).
 - The MAXQ parameter specifies the maximum amount of storage, in megabytes, that TVISIONB will allocate for its buffer queue. Please refer to [The TVISIONB Buffer Queue](#) on page 101.
 - The EDPROC parameter specifies the name of the procedure to start the TVISIOND address space.
 - The IMSJOB parameter specifies the jobname of the IMS control region for the IMS system to be monitored.
- 2 Include the thlqual.SSLMAUTH in the STEPLIB concatenation for each IMS control region for which TransactionVision WebSphere MQ-IMS bridge monitoring is required or copy the DFSYIOE0 module to an existing qualifying library.

WebSphere MQ-IMS Bridge Sensor Operation

To operate the WebSphere MQ-IMS bridge Sensor, perform the following steps:

- 1 Assure that IMS control region is started with the TransactionVision DFSYIOE0 exit routine accessible in its STEPLIB.
- 2 Start the TVISIONB address space from the system operator's console, specifying any parameters to be overridden in the startup procedure. For example:

```
S TVISIONB[.jobname],IMSJOB=IMS71CR1,QMGR=CSQ1, MAXQ=10
```

If you will be running multiple instances of the Sensor, you should specify a unique jobname for each instance. Otherwise, the jobname will default to the procedure name and all MVS modify and stop commands will apply to all instances. Alternatively, create separate, uniquely named startup procedures for each IMS system to be monitored.

If IMSJOB is omitted (for example, specified as nul), the started Sensor instance will monitor each IMS system in which the DFSYIOE0 exit routine is driven and which is not explicitly monitored by another instance of the Sensor. If an IMS system-specific Sensor is started while a monitor-all Sensor is running, monitoring of the targeted IMS system will be switched to the specific Sensor instance. Conversely, when a specific Sensor is stopped, monitoring of the targeted IMS system will be switched to the monitor-all Sensor, if running. To avoid confusion, it is recommended that you run only specific Sensors or run a monitor-all Sensor and no specific Sensors. Only one monitor-all Sensor will be allowed and only one Sensor monitoring each specific IMS system will be allowed.

```
TVISIONB will automatically start TVISIOND.
```

- 3 Request bridge monitoring from the TransactionVision web application on a connected workstation. Please refer to the *TransactionVision User's Guide* for more information.
- 4 Ordinarily, the activity of the bridge Sensor is controlled from the TransactionVision web application. However, you may disable the Sensor from the system console with the MODIFY command: F TVISIONB,DISABLE MQIMSB DG. When disabled the TransactionVision exit routine, DFSYIOE0, continues to run in the IMS control region but sends no events to the TVISIONB server component. Re-enable the Sensor as follows: F TVISIONB,ENABLE MQIMSB DG.
- 5 Stop the TVISIONB address space as follows: P TVISIONB. This will implicitly disable the Sensor; the exit routine continues running but does not attempt to send events to the TVISIONB. TVISIOND will automatically be stopped.

Any events in the buffer queue will be sent to the event dispatcher component before shutdown completes. To avoid this quiesce function, you may request an immediate shutdown, in which case all events in the buffer queue are discarded: P TVISIONB IMMED.

The TVISIONB Buffer Queue

The Sensor server component maintains an in-storage queue to buffer events flowing from the exit routine through TVISIONB to TVISIOND. It is likely that the rate of events from the exit routine will be several times faster than the rate of event dispatching by TVISIOND. The queue will expand and contract in response to these respective flows. The maximum size of the queue may be controlled irrespective of the REGION specification.

On the TVISIONB start command or in the startup procedure, specify MAXQ=nn, where nn is the maximum size of the queue in megabytes. The minimum size is 3. The maximum allowed value is 2046—to allow TVISIONB to use the entire 2GB address space.

TVISIONB allocates and frees its queue storage in 1MB blocks. If TVISIONB cannot allocate an additional block when required, either because of the MAXQ limitation or REGION size constraints, it issues a warning message and, when the current block is full, it discards any new events until it is able to allocate a new block. Events already queued will continue to be collected.

To define the optimum MAXQ specification for your environment will require some experimentation. However, a generous specification that turns out to be unnecessary is not costly since the queue will contract to as low as 2MB when the excess is not needed regardless of the MAXQ setting.

Event Data

The WebSphere MQ-IMS bridge Sensor collects the following event data for each WebSphere MQ-IMS bridge event:

- Input/output flag
- Segment sequence indicator
- Transaction code
- IMS message (or message segment)
- Userid
- Cross Systems Coupling Facility (XCF) member name of queue manager
- The message descriptor (MQMD) specified on the MQPUT in the originating application.

To cause the Sensor to add the queue manager and queue object to the WebSphere MQ-IMS bridge entry event data, the Analyzer requires an event modifier bean. The bean provided with TransactionVision provides a simple approach. It defines the WebSphere MQ queue manager and queue objects in separate XML configuration files, and defines a special event modifier to pick up the definition and insert that into WebSphere MQ-IMS bridge entry events. The following two files, located in `<TVISION_HOME>/config/services`, are used to set up an WebSphere MQ-IMS bridge entry event modifier:

- Beans.xml
- IMSBridgeObject.xml

Beans.xml

This file sets up the event analysis framework by defining a chain of processing beans. By default, `com.Bristol.tvision.services.analysis.event-modifier.IMSBridgeEntryModiferBean` is already defined under `EventModifierCtx`, which reads an object definition from `IMSBridgeObject.xml` and plugs the definition (in the format of an XML document fragment) into the event XML document if that event is an WebSphere MQ-IMS bridge entry event.

```
<Module type="Context" name="EventModifierCtx">
  <!--
    This context contains beans that modify XML event,
    which are unmarshalled from the raw event stream.
    User can easily plug in their modifier in this
    section. The sample bean checks the user data to
    see if its a valid XML document. If so, the
    document will be parsed and plugged under the user
    data node
  -->
  <Module type="Bean"
class="com.bristol.tvision.services.analysis.eventmodifier
.DefaultModifierBean"/>
  -->
  <!--
    This bean read MQObject definition for IMS bridge
    entry event from $TVISION_HOME/config/service/
    IMSBridgeObject.xml
  -->
  <Module type="Bean"
class="com.bristol.tvision.services.analysis.eventmodifier
.IMSBridgeEntryModifierBean"/>
</Module>
```

IMSBridgeObject.xml

This file defines the WebSphere MQ queue objects that generate the WebSphere MQ-IMS bridge events, as in the following sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<IMSBridgeMQObject>
  <MQObject objectName="IMS.BRIDGE.QUEUE" queueManager="MQS1"
objectType="Q_LOCAL"/>
</IMSBridgeMQObject>
```

Attribute	Description
objectName	Defines the WebSphere MQ queue name
queueManager	Defines the queue manager name
objectType	Defines the type of queue. Valid values are Q_LOCAL, Q_ALIAS, Q_REMOTE, Q_CLUSTER, Q_LOCAL_CLUSTER, Q_ALIAS_CLUSTER, Q_REMOTE_CLUSTER, and DISTRIBUTION_LIST

Note that only one MQOBJECT element is defined under the root element IMSBridgeMQObject. If multiple MQObject elements are defined, the event modify bean just picks up the first one.

Depending on the object type, the XML document may extend the structure to provide more detailed information. For example, the following defines a remote queue object:

```
<?xml version="1.0" encoding="UTF-8"?>
<IMSBridgeMQObject>
  <MQObject objectName="REMOTE.BRIDGE.QUEUE" queueManager="MQS1 "
objectType="Q_REMOTE">
  <MQObject objectName="IMS.BRIDGE.QUEUE" queueManager="MQS2 "
objectType="Q_LOCAL">
  </MQObject>
</IMSBridgeMQObject>
```

The XML schema is located in *<TVISION_HOME>/config/xmlschema/IMSBridgeObj.xsd*.

Data Collection Filters and Queries

Filtering (either in a data collection filter or query) is not provided on some event attributes such as user name, IMS PSB name, IMS region type, IMS identifier, program, entry event queue, and queue manager or return code.

To filter on the WebSphere MQ-IMS bridge entry or exit events, select the appropriate API, either on the WebSphere MQ API criteria page (queries) or the MQ IMS Bridge API criteria page (data collection filters):

API	Description
MQIMS_BRIDGE_ENTRY	WebSphere MQ-IMS bridge entry event
MQIMS_BRIDGE_EXIT	WebSphere MQ-IMS bridge exit event

Using the WebSphere Business Integration Sensor

TransactionVision provides a WebSphere Business Integration (WBI) Sensor that enables TransactionVision to distinguish the various message flows and identify individual logical transaction paths within WBI. The WBI Sensor is a WBI plug-in that supports trace nodes (TransactionVisionTrace), inserted into normal execution paths, and failure nodes (TransactionVisionFailure), inserted into failure paths.

Any number of processing nodes may be inserted into an existing message flow at the desired points. Each processing node is a checkpoint that collects the state of the current message flow and reports it to the Analyzer. The reported event provides information such as broker name, message flow name, message data, etc. A unique label may be assigned to each node; the label is reported in the TransactionVision event associated with the node instance.

To install and configure the WBI Sensor, you must do the following:

- Integrate the TransactionVision plugin with the Message Brokers Toolkit for WebSphere Studio.
- Install the TransactionVision WBI Sensor on the WBIMB platform.

Message Brokers Toolkit for WebSphere Studio Integration

The following steps are based on the standard Message Brokers Toolkit and Eclipse Technology plug-in installation procedures:

- 1 Ensure that the Message Brokers Toolkit is not running.
- 2 Unzip `sensor_install_directory\mqsi\TransactionVisionWBIPlugin.zip` to `WBIMB_install_directory\eclipse\plugins`.

When the Message Brokers Toolkit is started, the TransactionVision trace nodes will be visible with the other built-in nodes when editing a message flow.

TransactionVision User-Defined Node Installation for WBIMB

Perform the following steps to install the TransactionVision WBI Sensor on the WBIMB platform:

- 1 Stop the WBI message broker(s).
- 2 Copy the WBI Sensor user-defined node library to the corresponding WBIMB install subdirectory.

Windows:

Copy the library `<sensor_install_directory>\mqsi\tvisiontrace.lil` to the directory `<WBIMB_install_directory>\bin`.

UNIX:

Copy the library to the directory `<WBIMB_install_directory>/lil`.

- 3 Restart the WBI message broker(s).

Node Insertion

You may now insert any number of TransactionVision Sensor trace and failure nodes into any message flows through the Message Brokers Toolkit. Remember that any changes to the configuration repository must be deployed to the appropriate brokers.

See the *TransactionVision User's Guide* for information about using the WBI Sensor in TransactionVision data collection and analysis.

8 Configuring the Proxy Sensor

The TransactionVision Proxy Sensor enables TransactionVision to provide a basic level of correlation of business transactions into process that are not monitored using TransactionVision Sensors. Some examples of the appropriate applications of the Proxy Sensor include:

- Transactions where a monitored application places a request message on a queue, after which an application running on a platform not supported by the TransactionVision Sensor (such as Tandem) retrieves the message, processes it, and places a reply on a queue for retrieval by the monitored application.
- Transactions where a monitored application places a request message on queue, after which an application at a business partner location (where TransactionVision is not installed) retrieves the message, processes it, and places a reply on a queue for retrieval by the monitored application.

In these scenarios, where some unsensored applications are participating in the business transaction, the Proxy Sensor enables TransactionVision to provide limited information about the entire business transaction.

Unlike the TransactionVision Sensor, the Proxy Sensor is a Java bean that runs within the Analyzer. It recognizes transactions that are going to unsensored applications and creates special proxy objects to represent the unsensored applications involved in the transaction.

Application Requirements

For the Proxy Sensor to correlate business transactions involving unsensored applications, the applications must meet the following requirements:

- The application monitored by the Sensor must maintain the message ID and correlation IDs in the MQMD.
- The application monitored by the Sensor must specify a Reply-To queue in the request.
- The unsensored application must provide a meaningful program name in the MQMD for reply events.

Enabling the Proxy Sensor

The Proxy Sensor is enabled by the TransactionVision license code.

Configuring the Proxy Definition File

The Analyzer generates proxy objects when WebSphere MQ events are from certain queues and belong to a request-reply MQPUT-MQGET pair with matching message and correlation IDs. The proxy definition file is an XML file that defines the attributes of proxy objects. It is located in `<TVISION_HOME>/config/services/ProxySensorDef.xml`.

You must define a proxy element for each unsensored application you wish to include in your TransactionVision analysis.



Whenever you modify this file, you must restart the Analyzer for the changes to take effect.

The following example defines a proxy element for the program P2:

```
<ProxySensor>
  <Proxy matchMsgIdToCorrelId="true">
    <Request queue="Q1" queueManager="QM1" />
    <Reply queue="Q2" queueManager="QM2" />
    <Retrieve queue="INPUT_LQ" queueManager="DWMQI1" />
    <Program name="P2" path="/usr/local/bin/P2path" />
    <Host name="P2_host_name" os="SOLARIS" />
  </Proxy>
</ProxySensor>
```

Subelements

Specify the following subelements for each proxy element:

Element	Required?	Attributes	Description
Request	Yes	queue queueManager	The WebSphere MQ queue and queue manager from which the proxy program gets the event.
Reply	Yes	queue queueManager	The WebSphere MQ queue and queue manager where the proxy program puts the reply event of the same message ID and correlation ID as the request event.
Program	Yes	name path	The name and path of the proxy program.
Host	Yes	name os	The name and operating system of the host where the proxy program runs.
Retrieve	No	queue	Causes the Proxy Sensor to check the given queue object against its definition rather than the object defined in <Reply>. This element allows the Sensor to work with looser coupling.

Element	Required?	Attributes	Description
z/OSBatch	No	jobID jobName stepName tcbAddr	If the proxy program is an z/OS Batch job, specify the job ID, job name, step name, and TCB address.
z/OSCICS	No	regionName transactionID taskNum	If the proxy program is an z/OS CICS task, specify the region name, transaction ID, and task number.
z/OSIMS	No	psbName transactionName regionID jobName imsID imsType	If the proxy program is an z/OS IMS job, specify the PSB name, transaction name, region ID, job name, IMS ID and IMS type.

Optional Attributes for the Proxy Element

In addition to the subelements above, you may specify the following optional attributes for any proxy element:

Attribute	Description
matchMsgIdToCorrelId	Causes the Proxy Sensor to match the message Id of the MQPUT with the correlation Id of the MQGET
matchCorrelIdToMsgId	Causes the Proxy Sensor to match the correlation Id of the MQPUT with the message Id of the MQGET
swapMsgCorrelId	Set to true to cause TransactionVision to swap the message ID and correlation ID for MQPUT/MQPUT1 events when generating the lookup key. This attribute cannot be used with either <i>matchMsgIdToCorrelId</i> or <i>matchCorrelIdToMsgId</i>

Configuring the User Interface

By default, the Component Topology Analysis view does not show proxy related links in dynamic mode. To enable the proxy node in this view, set the `hasProxySensor` attribute in the `UI.properties` file to true. For more information about changing this configuration file, see [Appendix B, Configuration Files](#).

9 Configuring Sensor Logging

Log Files

Java Sensors

By default, the TransactionVision Servlet, EJB, JDBC and JMS Sensors log error and warning messages to `sensor.log` in the location specified when you run the Java Agent Setup Module. This location is stored in the `Setup.properties` file.

To enable the Servlet and JMS Sensors to print banners when activated, set the `com.bristol.tvision.sensor.banner` Java property to `true`. The banner is printed to standard output.

WebSphere MQ Sensors

By default, the TransactionVision WebSphere MQ Sensors log error, warning, and trace messages to the `local0` facility in the UNIX system log (`syslogd`), the Windows event log, the z/OS operator console log, or the i5/OS user job log.

On UNIX platforms, you can specify the log facility by setting the `TVISION_SYSLOG` environment variable to one of the following values: `user`, `local0`, `local1`, `local2`, `local3`, `local4`, `local5`, `local6`, or `local7`. If `TVISION_SYSLOG` is not set or is set to a value other than those listed, TransactionVision uses `local0`. The target log file must already exist for `syslogd` to log to it. Contact your system administrator to set up the system log facility, if required.

On UNIX and Windows platforms, set the `TVISION_BANNER` environment variable, then start the application. A banner indicating that the application is loading the Sensor should appear. To disable this behavior, unset `TVISION_BANNER`. This environment variable can be set to any value. On Windows, it must be set to a value other than an empty string. On i5/OS, `TVISION_BANNER` does not display the library path as it does on UNIX.

Circular Logging

By default, the Servlet, EJB, and JMS Sensors employ a form of circular logging. When the log file reaches the configured maximum size, it is renamed as a backup file and a new, empty log file is created. By default, the maximum log size is 10 MB and there is one backup log file.

Using the defaults, when a log file (for example, the Sensor log file `sensor.log`, reaches 10 MB in size, it is renamed `sensor.log.1` and a new `sensor.log` file is created. If you change the configuration so that there are two backup files, the following events take place when `sensor.log` reaches 10 MB:

- `sensor.log.2` is removed if it exists.

- sensor.log.1 is renamed sensor.log.2.
- sensor.log is renamed sensor.log.1.
- A new sensor.log is created.

If you do *not* wish to use circular logging, you may change the configuration to use linear logging, in which a single log file is generated.

The `<TVISION_HOME>/config/logging/*.Logging.xml` files specify the type of logging used, the maximum log file size, and the number of backup log files for each component. For example, `Sensor.Logging.xml` specifies the configuration for the servlet and JMS Sensors. This file contains entries similar to the following:

```
<appender class="tvision.org.apache.log4j.RollingFileAppender"
name="SENSOR_LOGFILE">
  <param name="File" value="c:/Program Files/HP/TransactionVision/logs/
sensor.log" />
  <param name="Append" value="true" />
  <param name="MaxBackupIndex" value="2" />
  <param name="MaxFileSize" value="10MB" />
  <layout class="tvision.org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d [%t] %-5p %c %x - %m%n" />
  </layout>
</appender>
```

Maximum Log File Size

To change the maximum size of the log file, change the value of the `MaxFileSize` parameter to the desired size. Values provided should end in “MB” or “KB” to distinguish between megabytes and kilobytes.

Maximum Number of Backup Log Files

To change the number of backup files, change the value of the `MaxBackupIndex` parameter to the desired number of backup files.

Changing from Circular to Linear Logging

To use linear logging rather than circular logging, do the following:

- 1 In the appender class value, change `RollingFileAppender` to `FileAppender`. For example, in the previous example, change the first line to the following:

```
<appender class="tvision.org.apache.log4j.FileAppender"
name="SENSOR_LOGFILE">
```

- 2 Remove the entries for the `MaxBackupIndex` and `MaxFileSize` parameters.

Trace Logging

Trace logging provides verbose information of what a TransactionVision Sensor is doing internally. It is used mainly to troubleshoot problems and should **not** be turned on in production environments.

You can enable trace logging in TransactionVision Sensors to debug Sensor configuration issues. Trace information for the WebSphere MQ Sensor is logged to the UNIX system log, the Windows event log, the z/OS operator console log, or the i5/OS user's job log. For the Servlet, EJB, and JMS Sensors, trace messages are sent to the Sensor's log4j TraceLog.

To enable or disable Sensor trace logging in the communication link, see the *TransactionVision Administration Guide* for instructions.

Configuring Separate Log Files for Multiple Sensor Instances

If multiple applications servers or JVM processes on the same machine have the Sensor enabled, the Sensor must be set up to log either to the Windows or UNIX system log, or to a different log file for each application server or JVM. Not doing so will result in corrupt or overwritten log file entries.

To set up each Sensor instance to log to a different log file, perform the following steps:

- 1 Copy the `<TVISION_HOME>/config/sensor/Sensor.properties` file to another name in the `<TVISION_HOME>/config/sensor` directory. For example, if you are setting up the Sensor for two servers, serverA and serverB, copy `Sensor.properties` to `Sensor_serverA.properties` and `Sensor_serverB.properties`.
- 2 Copy the `<TVISION_HOME>/config/logging/Sensor.Logging.xml` file to another name in the `<TVISION_HOME>/config/logging` directory. For example, for each server (serverA and serverB), copy this file to `Sensor.Logging.serverA.xml` and `Sensor.Logging.serverB.xml`.
- 3 Modify the `logging_xml` property in each `Sensor.properties` file. For example, in `Sensor_serverA.properties`, change the property line to `logging_xml=Sensor.Logging.serverA.xml`. Similarly, in `Sensor_serverB.properties`, change the property line to `logging_xml=Sensor.Logging.serverB.xml`.
- 4 Set the JVM property `com.bristol.tvision.sensor.properties` to the appropriate `Sensor.properties` file. For example, for serverA set the JFM property as follows:

```
com.bristol.tvision.sensor.properties=sensor/Sensor_serverA.properties
```

For serverB, set the JVM property as follows:

```
com.bristol.tvision.sensor.properties=sensor/Sensor_serverB.properties
```

For stand-alone programs, this JVM property is set on the command line when the JVM is invoked, as follows:

```
java -Dcom.bristol.tvision.sensor.properties=sensor/Sensor_serverA.properties
```

For WebSphere, set this JVM property using the Administration console for the given application server under **Servers > Application Servers > Process Definition > Java Virtual Machine > Custom Properties**.

For WebLogic, set this JVM property using the WebLogic startup script. Open any text editor to edit the script. For example, startWebLogic.cmd. Set the JAVA_OPTIONS environment variable to include com.bristol.tvision.sensor.properties as follows:

```
SET JAVA_OPTIONS=%JAVA_OPTIONS% -Dcom.bristol.tvision.  
sensor.properties=<TVISION_HOME>/config/sensor/<custom properties file>
```

Using Windows and UNIX System Logs

On UNIX and Windows platforms, you can configure TransactionVision to log output to the system event logging facilities—the event log for Windows or syslog for UNIX. Examples of the logging configuration files needed to do this can be found in TVISION_HOME/config/logging/system/*/Sensor.Logging.xml.

For both Windows and UNIX, you must define a specialized event appender.

Windows Event Appender

The following example shows how to configure the Windows event appender to use the event log:

```
<appender name="NT_EVENT_LOG" class="tvision.org.apache.log4j.nt  
.NTEventLogAppender">  
  <layout class="tvision.org.apache.log4j.PatternLayout">  
    <param name="ConversionPattern" value="%d [%t] - %m%n"/>  
  </layout>  
</appender>
```

NT_EVENT_LOG can then be referenced in a category definition of your choice. For example:

```
<category additivity="false" class="com.bristol.tvision.util.  
log.XCategory" name="sensorLog">  
  <priority class="com.bristol.tvision.util.log.XPriority" value="info"/>  
  appender-ref ref="NT_EVENT_LOG"/>  
</category>
```

On Windows, you must also add a special DLL to your path. This DLL, NTEventLogAppender.dll, can be found in the config\logging\system\bin directory. For example:

```
set path=%TVISION_HOME%\config\logging\system\bin;%PATH%
```


UNIX Event Appender

The following example shows a UNIX event appender to use syslog:

```
<appender name="SYSLOG" class="tvision.org.apache.log4j.net.
SyslogAppender">
  <param name="SyslogHost" value="localhost"/>
  <param name="Facility" value="local0"/>
  <layout class="tvision.org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="[%t] %-5p %c %x
- %m%n"/>
  </layout>
</appender>
```

Specify the SyslogHost and Facility parameters as appropriate for your environment.

A Utilities Reference

SetupModule

Description

This script starts the Java Agent Setup Module.

Options

Option	Description
<code>-recordFile name.rec</code>	Records
<code>-installFile name.rec</code>	Plays back recording or automates
<code>-console</code>	Launches in console mode

MigrateConfig

Location

TVISION_HOME/bin/MigrateConfig.[sh|bat]

Description

This is an internal script called during TransactionVision installation to migrate configuration files for the Analyzer and UI from an older version of TransactionVision to the current version.

MigrateConfig will not migrate any Sensor configuration files from previous versions of the product. Sensors will need to be completely reinstalled and reconfigured.



Do NOT call this script directly; it may only be run during installation.

rebind_sensor

Location

TVISION_HOME/bin/rebind_sensor.sh

Description

This script rebinds the TransactionVision WebSphere MQ Sensor on the AIX platform.

In WebSphere MQ support pacs, internal symbols exported from the TransactionVision WebSphere MQ Sensor on the AIX platform may change. When an internal symbol that has been exported from the Sensor library is no longer available in the WebSphere MQ library, the application cannot start and fails with various symbol resolution errors.

Hence, the rebind_sensor script needs to be run whenever a WebSphere MQ support or fix pac is installed.

It modifies the TransactionVision Sensor libraries in TVISION_HOME/lib.

On WebSphere MQ 6.0 and above, this utility needs to be run twice, to instrument the 32-bit libraries and the 64-bit libraries, as follows:

```
$TVISION_HOME/bin/rebind_sensor.sh (rebinds the 32-bit library)
$TVISION_HOME/bin/rebind_sensor.sh -64 (rebinds the 64-bit library)
```

Syntax

```
rebind_sensor.sh [-v|-s|-h] [-64]
```

Options

Option	Description
-v	Writes errors to the console. The default behavior is to write errors to TVISION_HOME/logs/mqsensorbind.log
-s	Uses silent mode, which does not prompt before executing
-h	Displays usage message
-64	Rebinds the 64-bit Sensor library. The default, when this option is not present, is to rebind the 32-bit library

B Configuration Files

The TransactionVision setup utilities save Sensor configuration information necessary for TransactionVision to operate in the following configuration files. You may also modify these files directly if you need to make any changes to your configuration.



If you modify property files, you must restart the associated application for you changes to take effect.

License.properties

The `<TVISION_HOME>/config/license/License.properties` file specifies the TransactionVision license code supplied by HP.

Performance.properties

The `<TVISION_HOME>/config/services/Performance.properties` file contains the following settings for performance logging:

- `performance`
Specifies whether to performance logging is on or off. The default is off.
- `count_interval`
Specifies the number of events after which performance data is logged. The default is 1000.

Sensor.properties

The `<TVISION_HOME>/config/sensor/Sensor.properties` file specifies information about the servlet and JMS Sensors. It has the following entries:

- `logging_xml`
Specifies the name of the logging configuration file.
- `configuration_file`
Specifies the path name of the Sensor configuration XML file.

SensorConfiguration.xml

The SensorConfiguration.xml file defines the interaction between the Sensor and the configuration queue. It defines the following attributes, which are specified when you create or edit a communication link.

Attribute	Description
ConfigurationQM	The name of the configuration queue manager
ConfigurationQ	The name of the configuration queue
ConfigurationQMHost	The host name of the configuration queue manager
ConfigurationQMPort	The listener port number of the configuration queue manager
ConfigurationQMChannel	The channel name of the configuration queue manager
ConnectionRetryDelay	The delay in milliseconds between connection attempts when the connection to the configuration queue manager is lost. The default is 1000.
ConnectionRetryTimeout	The amount of time in milliseconds to try to reconnect to the configuration queue manager when the connection is lost. A value of -1 (the default value) is to retry forever.
ConfigurationRetrieveInterval	The time interval in milliseconds to check the configuration queue for new configuration messages. The default value is 10000.
SensorClientTimeSkewInterval	The time interval in milliseconds to check the time skew from the host running the Sensor and the host running the configuration queue manager. The default value is 300000 (5 minutes).
EventPackageFlushTimeout	The amount of time in milliseconds that an event package remains idle (no events added) before it is forced to be written to the event queue. The minimum value is 10000; the default is 300000 (5 minutes).
RepeatLogInterval	The amount of time in milliseconds to repeat a repetitive error that would normally be suppressed. The default value is 600000.

Setup.properties

The <TVISION_HOME>/config/setup/Setup.properties file specifies the following properties:

- default_tool_install_path
The directory location of the DefaultInstallPath.xml file, which lists the locations of software components required by TransactionVision
- logs_dir

- The name of the directory in which to store log files
- logging_xml
 - The name of the logging configuration file
- minimum_java_version
 - The minimum Java version required by TransactionVision
- minimum_java_version_sun
 - The minimum Java version required by TransactionVision on the Solaris platform
- maximum_java_version
 - The highest Java version supported by TransactionVision

tvision-wl-sensorconfig.properties

The `<TVISION_HOME>/config/weblogic/tvision-wl-sensorconfig.properties` file specifies WebLogic application server information for the servlet and JMS Sensors. It specifies the following properties:

- adminserver_name
 - Specifies the administration server name.
- adminserver_host
 - specifies the administration server host name.
- adminserver_port
 - Specifies the administration server port number.
- admin_user_name
 - Specifies the administration user name.
- admin_user_password
 - Specifies the administration user password.
- domain_dir
 - Specifies the domain directory.
- server_name
 - Specifies the name or names of the servers that to be monitored by TransactionVision. To specify multiple server names, separate them with commas.

tvision-ws-sensorconfig.properties

The `<TVISION_HOME>/config/websphere/tvision-ws-sensorconfig.properties` file specifies WebSphere application server information for the servlet and JMS Sensors. It specifies the following properties:

- appserver_names_v5

For WebSphere Application Server 5.x, specifies the cell name, node name, and server name. Separate multiple entries with a comma.

- appserver_names_v4

For WebSphere Application Server 4.x, specifies the node name and server name. Separate multiple entries with a comma.

- adminserver_host

For WebSphere Application Server 4.x, specifies the administrative server name.

- adminserver_port

For WebSphere Application Server 4.x, specifies the administrative server port number.

C Additional z/OS Settings

This appendix lists RACF authorizations, firewall settings, and MIPS requirements for TransactionVision on the z/OS platform.

RACF Authorizations

RACF authorizations are highly customized to the user's environment and hence there are no specific requirements for the TransactionVision Sensors. Use the programs and transactions listed in the following tables to help determine the required RACF authorizations.

For the TransactionVision CICS Sensor

Transaction	Needs to Run	Needs to Access WebSphere MQ	Needs to Run as a CICS Global User Exit
SLDS	Yes	No	No
SLDM	Yes	No	No
SLDP	Yes	No	No
SLDC	Yes	No	No
SLDD	Yes	No	No
SLDI	Yes	No	No

Transaction Program	Needs to Run	Needs to Access WebSphere MQ	Needs to Run as a CICS Global User Exit
SLDPCSX	Yes	No	No
SLDPCMXX	Yes	No	No
SLDPCCX	Yes	No	No
SLDPDSX	Yes	No	No

Transaction Program	Needs to Run	Needs to Access WebSphere MQ	Needs to Run as a CICS Global User Exit
SLDPUXI	Yes	No	No
TVISION	Yes	No	No
TVISIONC	Yes	Yes	No

Exit Program	Needs to Run	Needs to Access WebSphere MQ	Needs to Run as a CICS Global User Exit
SLDPTCX	Yes	No	Yes
SLDPPSX	Yes	No	Yes
SLDPICX	Yes	No	Yes
SLDPTDX	Yes	No	Yes
SLDPPCX	Yes	No	Yes
SLDPFCX	Yes	No	Yes
SLDPTSX	Yes	Yes	Yes

For the TransactionVision CICS WebSphere MQ (WMQ) Sensor

Transaction	Needs to Run	Needs to Access WebSphere MQ	Needs to Enable/Disable CICS Crossing Exit
SLMC	Yes	No	Yes

Transaction Program	Needs to Run	Needs to Access WebSphere MQ	Needs to Enable/Disable CICS Crossing Exit	Needs to be Accessed from CICS Crossing Exit
CSQCAPX	Yes	Yes	No	Yes
SLMC	Yes	Yes	Yes	No
SLMBCNFG	Yes	Yes	No	No

For the TransactionVision IMS WebSphere MQ (WMQ) Sensor

The TransactionVision IMS WMQ Sensor is run from within the IMS WMQ application that is being monitored. This is done by link editing a TransactionVision stub against the IMS WMQ application. Therefore, there may be some RACF authorizations to be performed.

The following libraries are used and may also need to be RACF authorized based on the local RACF authorization scheme.

Libraries	APF Authorized	In DFHRPL
SSLDLOAD	No	Yes
SSLDAUTH	Yes	No

Firewall Settings

Since the TransactionVision Sensors use WebSphere MQ to communicate with the TransactionVision Analyzer, no firewall settings are required. All communication is configured through WebSphere MQ.

MIPS Required

There are no specific MIPS requirements for the TransactionVision Sensor. Sensors are architected to be active only when required; even then, they use very little resources to achieve the required functionality.

Index

A

- application server
 - configuring See application server configuration
- application server configuration
 - WebLogic 9.x and 10, 50
- application servers
 - adding interceptors for Sensors for WebSphere, 46
 - configuring WebLogic 8.1 for JRockit JVM, 48
 - configuring WebLogic 8.1 for Sun JVM, 46
 - configuring WebLogic 9.2, 50
 - configuring WebLogic overview, 46
 - configuring WebLogic remote-started managed servers, 51
 - configuring WebSphere 5.x and 6.0, 41
 - configuring WebSphere 6.1, 44
 - running JRE Instrumenter for WebSphere IDE, 45
 - using JMS Sensor with WebSphere, 46

B

- Beans.xml, 102

C

- CICS Sensor
 - configuring SLMC, 81
 - installing on z/OS, 68
- CICS Sensors, 12
- circular logging, 109
- client application monitoring, 96
- configuration files, 119
- configuration queue check interval, 91
- configuration queue name, 88
- configuring
 - WebLogic 8.1 for JRockit JVM, 48
 - WebLogic 8.1 for Sun JVM, 46
 - WebLogic 9.2, 50

E

- environment variables
 - LD_LIBRARY_PATH, 86
 - LIBPATH, 86
 - SHLIB_PATH, 86
- event log, 111
- exit_sensor.deny, 92

F

- FASTPATH_BINDING, 96
- files
 - Java Agent installation, 16
 - UNIX installation for WebSphere MQ and User Event Sensors, 55

I

- IMSBridgeObject.xml, 102
- installation
 - packages, 12
 - upgrading, 12
- installations
 - silent for Java Agent, 32

J

- Java Agent
 - about, 15
 - configuring remote-started WebLogic managed servers, 51
 - configuring WebLogic 8.1, 46
 - configuring WebLogic 8.1 for JRockit JVM, 48
 - configuring WebLogic 9.2, 50
 - installation files, 16
 - Java Sensors, 11
 - launching the installer on Windows, 16
 - running the installation on Windows, 17
 - running the JRE Instrumenter, 33
 - setupModule utility, 115
 - silent installation, 32
- Java Sensors
 - logging, 109

JRE Instrumenter
 processing, 34
 running for WebSphere IDE, 45
 running on UNIX, 37
 running on Windows, 34
JRockit JVM, configuring WebLogic 8.1, 48

L

LD_LIBRARY_PATH environment variable, 85, 86
LIBPATH environment variable, 85, 86
License.properties file, 119
license code, 119
logging, 120
 circular, 109
 Java Sensors, 109
 multiple log files, 111
 separate log files for multiple Sensors, 111
 trace, 111
 UNIX event appender, 113
 WebSphere MQ Sensors, 109
 Windows event appender, 112

M

messaging system provider
 configuring WebSphere MQ, 91
messaging system providers
 configuring SonicMQ, 53
 configuring TIBCO EMS, 53
 configuring WebLogic JMS, 53
 configuring WebSphere MQ, 52
MigrateConfig utility, 116
MQ_CONNECT_TYPE, 96

O

operator console log, 111

P

PATH environment variable, 85
Performance.properties, 119
Proxy Sensor
 application requirements, 105
 configuring, 105
 configuring the definition file, 106
 configuring the user interface, 107
 enabling, 105
 option attributes, 107
 subelements, 106
ProxySensorDef.xml, 106

R

rebind_sensor utility, 117

S

Sensor.properties, 119

Sensors

 CICS, 12
 client application monitoring, 96
 configuring the Proxy Sensor, 105
 configuring WebSphere MQ library, 85
 installation on IBM i5/OS, 65
 installing WebSphere MQ and User Event on
 UNIX, 55
 installing WebSphere MQ and User Event on
 Windows, 59
 installing WebSphere MQ CICS, WebSphere MQ
 IMS Bridge Sensors on IBM z/OS, 74
 Java, 11
 loading WebSphere MQ, 88
 logging, 109
 modifying WebSphere MQ and User Event on
 Windows, 62
 multiple log files, 111
 rebinding on AIX, 117
 rebinding WebSphere MQ on AIX, 56
 trace logging, 111
 types, 9
 uninstalling WebSphere MQ and User Event on
 UNIX, 57
 uninstalling WebSphere MQ and User Event on
 Windows, 63
 upgrading from previous releases, 12
 upgrading WebSphere MQ and User Event on
 Windows, 60
 WebSphere MQ, 10
 WebSphere MQ API Exit, 91
 WMQ-IMS bridge, 99

Setup.properties file, 120

setupModule utility, 115

SHLIB_PATH environment variable, 85, 86

silent installation, 32

SLMC

 configuring for CICS, 81

SMPE installation procedure, 74

SonicMQ

 configuring as a messaging system provider, 53

subelements of Proxy Sensor, 106

SYSEVENT TRANSWAP macro, 73

system log, 111

T

- TIBCO EMS
 - configuring as a messaging system provider, 53
- trace logging, 111
- TVISION_CONFIG_CHECK_INTERVAL
 - environment variable, 91
- TVISION_CONFIGURATION_QUEUE
 - environment variable, 88
- TVISION_SYSLOG environment variable, 111
- tvisionapiexit, 91
- tvision-wl-sensorconfig.properties file, 121
- tvision-ws-sensorconfig.properties, 121

U

- uninstalling
 - WebSphere MQ and User Event Sensors on UNIX, 57
 - WebSphere MQ and User Event Sensors on Windows, 63
- UNIX
 - configuring the event appender to use the event log, 113
 - installing WebSphere MQ and User Event Sensors, 55
 - running the JRE Instrumenter, 37
 - uninstalling WebSphere MQ and User Event Sensors installation, 57
 - WebSphere MQ and User Event Sensor installation files, 55
- upgrading
 - from previous Sensor releases, 12
 - WebSphere MQ and User Event Sensors on Windows, 60
- upgrading from previous releases, 12
- User Event Sensor
 - installing on UNIX, 55
 - installing on Windows, 59
 - modifying on Windows, 62
 - uninstalling on UNIX, 57
 - uninstalling on Windows, 63
 - upgrading on Windows, 60
- utilities
 - MigrateConfig, 116
 - rebind_sensor, 117
 - setupModule, 115

W

- WBI
 - broker user-defined node installation, 104
 - integration, 104
- WebLogic
 - configuration overview, 46
 - configuring 8.1 for JRockit JVM, 48
 - configuring 8.1 for Sun JVM, 46
 - configuring 9.2, 50
 - configuring remote-started managed servers, 51
- WebLogic 9.x and 10, application server configuration, 50
- WebLogic JMS
 - configuring as a messaging system provider, 53
- WebSphere
 - adding interceptors for Sensors, 46
 - configuring 5.x and 6.0, 41
 - configuring 6.1, 44
 - running JRE Instrumenter for IDE, 45
 - using the JMS Sensor, 46
- WebSphere MQ
 - configuring as a messaging system provider, 52
 - configuring the messaging system provider, 91
 - Sensors, 10
- WebSphere MQ API Exit Sensor, 91
 - configuring on distributed platforms, 92
 - configuring on i5/OS, 92
 - configuring on Windows, 94
 - discarding WMQ events on TransactionVision queues, 95
- WebSphere MQ Batch Sensor
 - installing on z/OS, 68
- WebSphere MQ IMS Sensor
 - installing on z/OS, 68
- WebSphere MQ Sensor
 - installing on UNIX, 55
 - installing on Windows, 59
 - logging, 109
 - modifying on Windows, 62
 - rebinding on AIX, 56
 - uninstalling on UNIX, 57
 - uninstalling on Windows, 63
 - upgrading on Windows, 60
- WebSphere MQ Sensor library
 - configuring, 85

Windows

- configuring the event appender to use the event log, 112
- installing WebSphere MQ and User Event Sensors, 59
- launching the Java Agent installer, 16
- modifying WebSphere MQ and User Event Sensors installation, 62
- running the Java Agent installation, 17
- running the JRE Instrumenter, 34
- uninstalling WebSphere MQ and User Event Sensors installation, 63
- upgrading WebSphere MQ and User Event Sensors installation, 60

WMQ-IMS Bridge Sensor

- additional setup, 83
- using, 99

wmqsensor, 86

Z

z/OS

- installing CICS, WebSphere MQ Batch, WebSphere MQ IMS Sensors, 68
- installing WebSphere MQ CICS, WebSphere MQ IMS Bridge Sensors, 74
- setting up WebSphere MQ IMS Bridge, 83
- SMPE installation procedure, 74

z/OS CICS

- configuring SLMC, 81

