HP Business Availability Center

for the Windows and Solaris operating systems

Software Version: 8.00

TransactionVision Deployment

hp ®

i n v e n t

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.

Third-Party Web Sites

HP provides links to external third-party Web sites to help you find supplemental information. Site content and availability may change without notice. HP makes no representations or warranties whatsoever as to site content or availability.

Copyright Notices

© Copyright 2000 - 2009 Mercury Interactive (Israel) Ltd.

Trademark Notices

TransactionVision® is a registered trademark of the Hewlett-Packard Company.

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 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

Table of Contents

Welcome to This Guide

Welcome to the TransactionVision Deployment Guide. This guide introduces you to TransactionVision, provides information on getting started, describes server and component configuration and installation, and details the upgrade process.

This chapter includes:

How This Guide Is Organized

This guide contains the following chapters/parts:

Part I   introduction to TransactionVision
          Introduces TransactionVision and provides an overview of the TransactionVision platform and components.

Part II   Analyzer Installation and Configuration
          Describes how to install and configure the TransactionVision Analyzer.

Part III UI/Job Server Installation and Configuration
          Describes how to install and configure the UI/Job Server.

Part IV   Sensor and Agent Installation and Configuration

Describes how to install and configure the TransactionVision Sensors and Agents.

Part V   Security

Describes how to secure the TransactionVision components.

Part VI   Appendixes

Utilities, configuration files, and other information related to TransactionVision.

Who Should Read This Guide

This guide is for the following users of TransactionVision:

➤ Application developers or configurators

➤ System or instance administrators

➤ Database administrators

Readers of this guide should be moderately knowledgeable about enterprise application development and highly skilled in enterprise system and database administration.

TransactionVision Documentation

TransactionVision documentation provides information on using the TransactionVision application of the Business Availability Center and deploying and administering the TransactionVision-specific components in the Business Availability Center deployment environment.

The TransactionVision documentation includes:

➤ The *TransactionVision Deployment Guide* describes the installation and configuration of the TransactionVision-specific components in the Business Availability Center deployment environment. This guide is available as a PDF in the Business Availability Center Documentation Library.

➤ The *Using TransactionVision Guide* describes how to set up and configure TransactionVision to track transactions and how to view and customize reports and topologies of business transactions. This guide is available as the TransactionVision Portal or as a PDF in the Business Availability Center Online Documentation Library.

➤ The *TransactionVision Planning Guide* contains important information for sizing and planning new installations. This guide is available by download from the HP Software Product Manuals site.

Additional TransactionVision documentation can be found in the following areas of the Business Availability Center:

Readme. Provides a list of version limitations and last-minute updates. From the HP Business Availability Center DVD root directory, double-click readme80.html. You can also access the most updated readme file from the HP Software Support Web site.

What's New. Provides a list of new features and version highlights. In HP Business Availability Center, select Help > What's New.

Online Documentation Library. The Documentation Library is an online help system that describes how to work with HP Business Availability Center and the TransactionVision application. You access the Documentation Library using a Web browser. For a list of viewing considerations, see "Viewing the HP Business Availability Center Site" in chapter 6 of the the *HP Business Availability Center Deployment Guide* PDF.

To access the Documentation Library, in HP Business Availability Center, select Help > Documentation Library. Context-sensitive help is available from specific HP Business Availability Center pages by clicking Help > Help on this page and from specific windows by clicking the Help button. For details on using the Documentation Library, see "Working with the HP Business Availability Center Documentation Library" in Platform Administration.

Additional Online Resources

Troubleshooting & Knowledge Base accesses the Troubleshooting page on the HP Software Support Web site where you can search the Self-solve knowledge base. Choose **Help** > **Troubleshooting & Knowledge Base**. The URL for this Web site is http://h20230.www2.hp.com/troubleshooting.jsp.

HP Software Support accesses the HP Software Support Web site. This site enables you to browse the Self-solve knowledge base. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. Choose **Help** > **HP Software Support**. The URL for this Web site is www.hp.com/go/hpsoftwaresupport.

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 user ID, go to:
http://h20229.www2.hp.com/passport-registration.html

HP Software Web site accesses the HP Software Web site. This site provides you with the most up-to-date information on HP Software products. This includes new software releases, seminars and trade shows, customer support, and more. Choose **Help** > **HP Software Web site**. The URL for this Web site is www.hp.com/go/software.

Documentation Updates

HP Software is continually updating its product documentation with new information.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to the HP Software Product Manuals Web site (http://h20230.www2.hp.com/selfsolve/manuals).

Part I

Introduction to TransactionVision

1

Introduction to TransactionVision

This chapter includes:

About TransactionVision

HP TransactionVision is the transaction tracking solution that graphically shows you the interaction between all the components of your system.

HP TransactionVision non-intrusively records individual electronic events generated by a transaction flowing through a computer network. More importantly, TransactionVision's patented "Transaction Constructor" algorithm assembles those events into a single coherent business transaction.

Key capabilities of HP TransactionVision include:

➤ End-to-end visibility of individual business transactions

➤ Non-intrusively tracks each business transaction across each processing step

➤ Automatically correlates application events into business transactions

➤ Deep visibility to reduce mean time to problem isolation & resolution of business transaction issues.

➤ Collects both technical and business data for each transaction, identifying each transaction's business context (customer identity, financial value, etc).

➤ Comprehensive integration with HP Business Availability Center and key applications including End User Monitoring, Diagnostics and Business Process Insight.

Architecture Diagram

The following diagram shows the key components of TransactionVision.
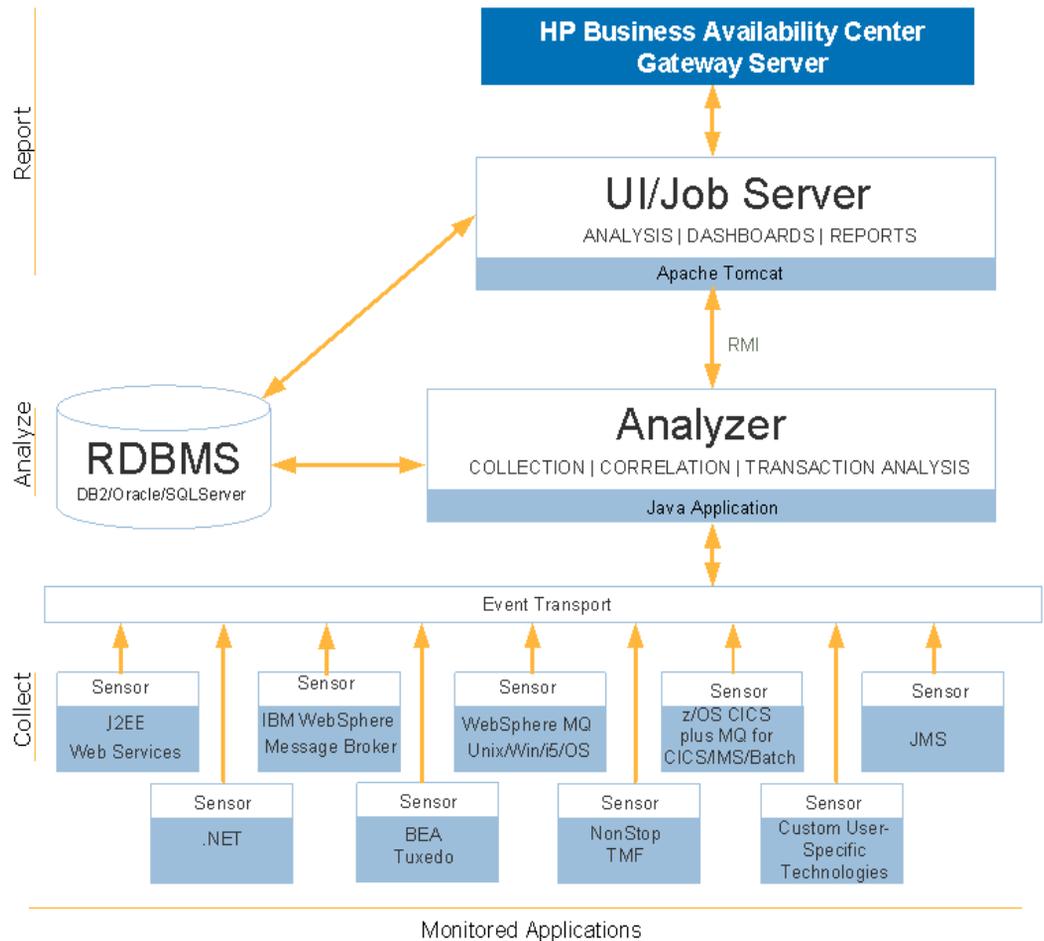
UI/Job Server

The TransactionVision UI/Job Server is a web application running on an instance of the Apache Tomcat Servlet/JSP container. The Server communicates with the Analyzer to provide data collection configuration

information such as communication links and data collection filters. It also connects to project database schemas to display project analysis and report results.

See Part III for information about the installation and configuration of the UI/Job Server.

Analyzer

The TransactionVision Analyzer is a service on Windows (or a daemon on UNIX) that communicates with TransactionVision Sensors via messaging middleware. It generates and delivers configuration messages to Sensors by placing them on a designated configuration queue. Configuration messages specify Sensor configuration information such as the name of the event queue where the Sensor should place event messages and data collection filter definitions for the project.

By default, TransactionVision uses SonicMQ as the messaging middleware provider. TIBCO EMS and WebSphere MQ are also supported.

The Analyzer also retrieves events placed on an event queue by Sensors and processes them for analysis and display by the web user interface. It performs the unmarshalling, correlation, analysis, and data management functions.

See Part II for information about the installation and configuration of the Analyzer.

Sensors

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.

The .NET Sensor and Java Sensor are referred to as the .NET Agent and the Java Agent respectively. Agents combine the capabilities of the HP Diagnostics Probe and the TransactionVision Sensor into a single component.

See Part IV for information about the installation and configuration of the Sensors and Agents.

RDBMS (Database)

TransactionVision uses a third-party RDBMS to store data. The Analyzer retrieves and processes events collected by Sensors and places them into event related tables. By using schemas to partition event data by project, you can control access to event data collected by each project.

See "Configuring Databases" on page 59 for more information.

TransactionVision in the Deployment Environment

TransactionVision operates in the Business Availability Center Deployment Environment as follows:

➤ The TransactionVision Analyzer is installed on a separate host from that on which the Business Availability Center Gateway Server is installed.

➤ The TransactionVision UI/Job Server is installed on the Business Availability Center Gateway Server host.

➤ The database used by TransactionVision is separate than the database used by Business Availability Center, and it is installed on a host that is optimally accessed from the TransactionVision Analyzer.

The following diagram shows the TransactionVision components in a typical Business Availability Center deployment environment.
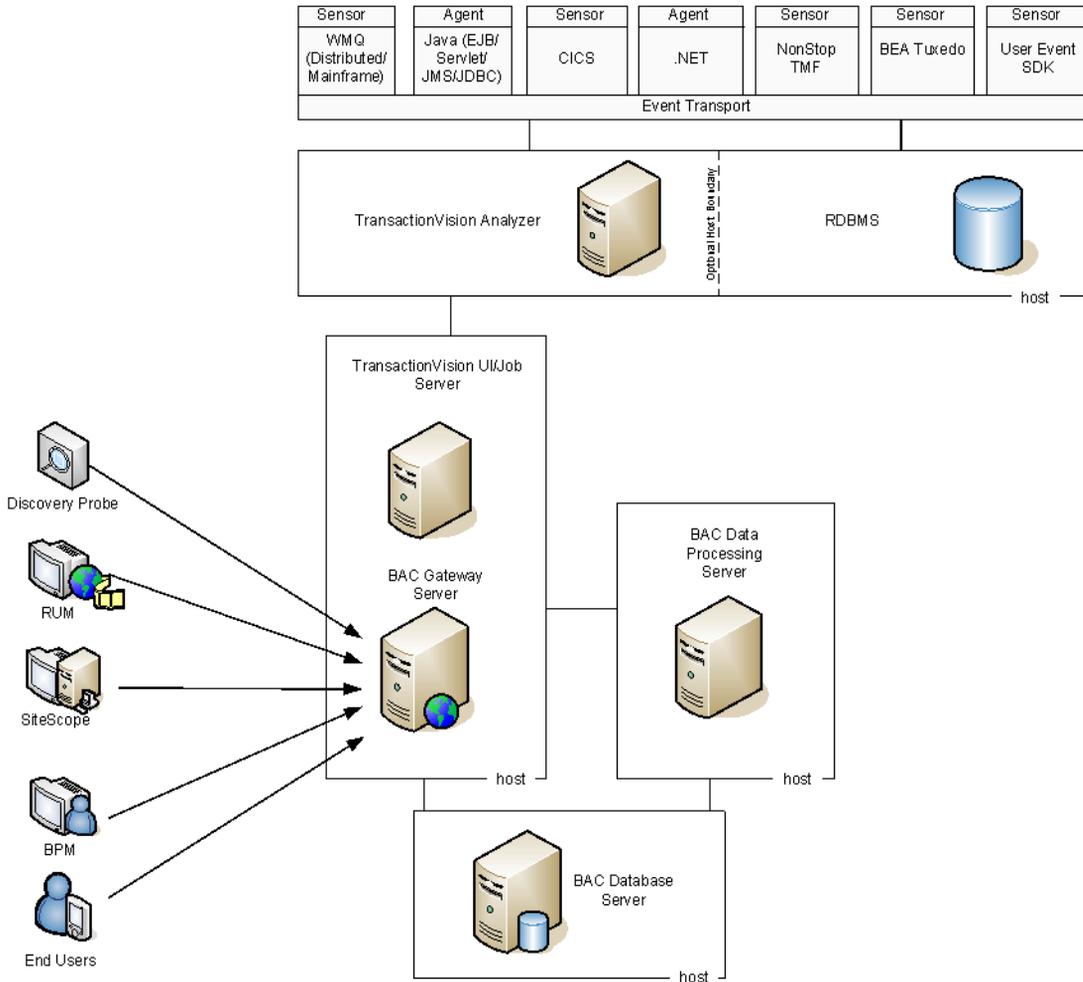
In deployment environments with multiple BAC Gateway Servers, such as in support of a high-availability scenario, only one TransactionVision UI/Job Server can be running at any given time. You can set up a mirrored UI/Job Server on the other BAC Gateway Server hosts, but it cannot be started until the primary TransactionVision UI/Job Server is stopped.

For more information about the Business Availability Center deployment environment, see the *HP Business Availability Center Deployment Guide* PDF.

Installation Packages

The TransactionVision Analyzer, UI/Job Server, and Sensors are installed separately from the Business Availability Center components in the deployment environment.

The TransactionVision components are in packages that are specific to a platform. Each set of installation instructions in this guide indicates which installation package to use and where to locate them.

Before installing a package, make sure the systems you are installing on meet the system requirements for TransactionVision.

Backward Compatibility

The following table summarizes the backward compatibility of TransactionVision components.

	8.00 Analyzer	8.00 UI/ Job Server	8.00 Sensors/ Agents	8.00 .NET Agents
7.50 Analyzer				✓
7.50 Web Component				✓
7.50 Sensor/Agents	✓	✓		
5.0.0 Analyzer				
5.0.0 Web Component				
5.0.0 Sensors/Agents	✓	✓		
5.0.0 SPC Analyzer				
5.0.0 SPC Web Component				
5.0.0 SPC Sensors/Agents	✓	✓		
5.0.0 SPE Analyzer				
5.0.0 SPE Web Component				
5.0.0 SPE Sensors/Agents	✓	✓		
5.0.0 SPH Analyzer				
5.0.0 SPH Web Component				
5.0.0 SPH Sensors/Agents	✓	✓		

TransactionVision 8.00 Analyzer and UI/Job Server components are not compatible with Analyzer and web application components from previous releases.

Except for the .NET Agent, TransactionVision 8.00 Sensors cannot be used with older versions of the TransactionVision Analyzer. The 8.00 .NET Agent is supported with the 7.50 Analyzer.

Other Sensors and Agents from releases 5.0.0, 5.0.0 SPC, 5.0.0 SPE, 5.0.0 SPH and 7.50 can be used with the TransactionVision 8.00 Analyzer and UI/Job Server. However, it is highly recommended that the 8.00 Sensors be installed to take advantage of the latest updates and enhancements.

For information about new features of TransactionVision in this release, see the Business Availability Center *Readme*.

Upgrading from Previous Releases

TransactionVision Web components and Analyzers from release 7.50 can easily be migrated to release 8.0. Sensors do not contain configurations that are migratable.

The general upgrade process is as follows:

1 Run the Analyzer and UI/Job Server installation programs. They each detect any previous installation these components on the host and provide an option to back up old configuration files and migrate them to release 8.00.

2 For hosts that are running a previous version of the TransactionVision Web User Interface, run **TVisionSetupInfo -cleanweb** to remove the old TransactionVision Web User Interface application.

Note: The TransactionVision Web User Interface from previous releases is no longer a product component. The functionality is now provided both by Business Availability Center and the TransactionVision UI/Job Server.

In 8.00, the TransactionVision UI/Job Server component is installed with a self-contained TomCat application server so WebSphere or WebLogic are no longer needed in the deployment environment.

---

3 Migrate project databases with the **MigrateDB** script after TransactionVision is installed and configured. Be sure to backup your databases before running the **MigrateDB** script.

4 After the components are migrated, run the **TVisionSetupInfo** scripts.

5 Run **SupervisorStart** on Windows or **run_topaz** on UNIX platforms.

See the installation chapters in this manual for detailed steps of the upgrade procedures.

See "TransactionVision Known Problems and Issues" in the *Readme* for information regarding upgrading or reinstalling TransactionVision.

2

Reviewing System Requirements

This chapter describes the system requirements required for running the TransactionVision components of the HP Business Availability Center platform.

Note: The HP Business Availability Center readme file, available with the HP Business Availability Center package, contains additional system requirements for the current and previous HP Business Availability Center versions.

This chapter includes:

➤ Supported NonStop TMF Sensor Platforms on page 39

➤ Supported .NET Agent Platforms on page 39

➤ Supported Browser Configurations on page 39

➤ LDAP Support on page 40

➤ Java Support on page 40

➤ Flash Player Support on page 40

➤ Localization and I18N Support on page 40

Note: The system requirements for the Java Agent are described with each Java technology: Servlet, EJB, JMS, and JDBC.

Supported Analyzer Platforms

Operating Environment	WebSphere MQ	TIBCO EMS	SonicMQ
Windows 2003 Server x86 64-bit	6.0, 7.0	4.2.0, 4.4.2	7.5.2
AIX 5L 5.3, 6.1 POWER	6.0, 7.0	4.2.0, 4.4.2	7.5.2
Solaris 9, 10 SPARC	6.0, 7.0	4.2.0, 4.4.2	7.5.2
RedHat Enterprise Linux 5.0 WS/ES/AS x86 64-bit	6.0, 7.0	4.2.0, 4.4.2	7.5.2

Supported TransactionVision UI/Job Server Platforms

Platforms	Operating Environments
Microsoft Windows	Windows 2003 Server x86 64-bit
Sun Solaris	Solaris 9, 10 SPARC

The TransactionVision UI/Job Server component was part of the TransactionVision Web Application in previous releases. As of this release, the TransactionVision User Interface is hosted via Business Availability Center. The TransactionVision UI/Job Server must also be installed in addition to Business Availability Center in order to access the TransactionVision User Interface. See "TransactionVision in the Deployment Environment" on page 23.

Supported Database Management Systems

The following databases and associated platforms are supported by
TransactionVision. These database server configurations may be accessed
remotely via the DB2 client or the Oracle oci or thin client interfaces.
TransactionVision only supports JDBC 2.0.

DBMS Client/Server
DB2 9.1
Oracle 9.2
Oracle 10g
Oracle RAC 10g
Microsoft SQL Server 2005

Supported Messaging Middleware Providers

Agent/ Sensor	WebSphere MQ	Transaction - Vision SonicMQ	SonicMQ	TIBCO EMS	WebLogic JMS	HTTP
Java Agent	✓	✓	✓	✓	✓	
.NET Agent	✓	✓	✓			
WebSphere MQ Agent	✓					
BEA Tuxedo Sensor						✓
NonStop TMF Sensor						✓

Supported WebSphere MQ Sensor Platforms

Platform	Operating Environment	WebSphere MQ	Supports WMQ API Exit Sensor
Microsoft Windows	Windows 2003 Server x86 32 and 64-bit	6.0, 7.0 (32-bit)[1, 6]	Yes
Sun Solaris [4]	Solaris 9, 10 SPARC	6.0, 7.0 [1, 6] (32-bit) 6.0, 7.0 [1, 6] (64-bit)	Yes
Hewlett-Packard HP-UX [4]	HP-UX 11i v3 PA-RISC & Itanium	6.0, 7.0 [1, 6] (32-bit) 6.0, 7.0 [1](64-bit for library Agent only)	Yes
IBM AIX [4]	AIX 5L 5.3, 6.1 POWER	6.0, 7.0 [1, 6] (32-bit) 6.0, 7.0 [1, 6] (64-bit)	Yes
RedHat Linux [4]	Enterprise Linux 5.0 WS/ES/AS x86 32 and 64-bit	6.0, 7.0 [1, 6] (32-bit), 6.0, 7.0 [1, 6] (64-bit)	Yes

Platform	Operating Environment	WebSphere MQ	Supports WMQ API Exit Sensor
IBM i5/OS [2]	i5/OS V5R4 iSeries	6.0, 7.0 [1, 6]	Yes
IBM z/OS [3, 5]	z/OS 1.7, 1.8, 1.9 zSeries z/OS 1.7, 1.8, 1.9 Batch z/OS 1.7, 1.8, 1.9 CICS TS 2.x, 3.x z/OS 1.7, 1.8, 1.9 RRS z/OS 1.7, 1.8, 1.9 IMS 7.x, 8.x, 9.x	6.0, 7.0 [1]	N/A

[1] TransactionVision 8.00 supports WMQ applications running against WMQ 7.0 only if they are not using any new 7.0 specific API features.

[2] The C Library Agent is not supported for monitoring Java applications on i5/OS systems. Please use the API Exit Agent instead.

[3] BTTRACE and BTMQEXIT are not supported on z/OS.

[4] 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 Agent, 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*.

[5] If you are using the WebSphere MQ CICS Agent for z/OS, please contact Hewlett-Packard TransactionVision Technical Support to ensure you have the latest and most efficient version of this z/OS component.

[6] For WebSphere MQ 7.0, Fix Pack 7.0.0.1 is required in order to use the WebSphere MQ API Exit Sensor.

Supported WebSphere Message Broker Configurations

TransactionVision has the ability to monitor WebSphere MQ 6 API calls into and out of WebSphere Message Broker. The following grid shows the WebSphere Message Broker configurations supported by TransactionVision.

Operating Environment	WebSphere Message Broker Version
AIX 5L 5.3, 6.1 POWER	6.1
Windows 2003 Server x86 32 and 64-bit	6.1

Supported Servlet Sensor Platform

Platform	Operating Environment	Application Servers
Microsoft Windows	Windows 2003 Server x86 32 and 64-bit	IBM WebSphere Application Server V5.1, 6.0, 6.1 FP9+ [1,2] BEA WebLogic Application Server 8.1.6, 9.2.3, 10
Sun Solaris	Solaris 9, 10 SPARC	IBM WebSphere Application Server V5.1, 6.0, 6.1 FP9+ [1,2] BEA WebLogic Application Server 8.1.6, 9.2.3, 10
IBM AIX	AIX 5L 5.3, 6.1 POWER	IBM WebSphere Application Server V5.1, 6.0, 6.1 FP9+ [1,2] BEA WebLogic Application Server 8.1.6, 9.2.3, 10
RedHat Linux	RedHat Enterprise Linux 5.0 WS/ES/AS x86 32 and 64-bit	IBM WebSphere Application Server V5.1, 6.0, 6.1 FP9+ [1,2] BEA WebLogic Application Server 8.1.6, 9.2.3, 10

[1] TransactionVision 8.00 does not support IBM WebSphere Application Server Community Edition.

[2] TransactionVision 8.00 requires FixPack 9 or higher when using WebSphere Application Server 6.1 in order to eliminate performance problems seen with prior versions.

Supported EJB Sensor Platforms

Platform	Operating Environment	Application Servers
Microsoft Windows	Windows 2003 Server x86 32 and 64-bit	IBM WebSphere Application Server V5.1, 6.0, 6.1 FP9+ [1,2] BEA WebLogic Application Server 8.1.6, 9.2.3, 10
Sun Solaris	Solaris 9, 10 SPARC	IBM WebSphere Application Server V5.1, 6.0, 6.1 FP9+ [1,2] BEA WebLogic Application Server 8.1.6, 9.2.3, 10
IBM AIX	AIX 5L 5.3, 6.1 POWER	IBM WebSphere Application Server V5.1, 6.0, 6.1 FP9+ [1,2] BEA WebLogic Application Server 8.1.6, 9.2.3, 10
RedHat Linux	RedHat Enterprise Linux 5.0 WS/ES/AS x86 32 and 64-bit	IBM WebSphere Application Server V5.1, 6.0, 6.1 FP9+ [1,2] BEA WebLogic Application Server 8.1.6, 9.2.3, 10

[1] TransactionVision 8.00 does not support IBM WebSphere Application Server Community Edition.

[2] TransactionVision 8.00 requires FixPack 9 or higher when using WebSphere Application Server 6.1 in order to eliminate performance problems seen with prior versions.

Supported JMS Sensor Platforms

Platform	Operating Environment	JMS Service Provider
Microsoft Windows	Windows 2003 Server x86 32 and 64-bit	WebSphere MQ 6.0 [1] TIBCO EMS 4.2.0, 4.4.2 SonicMQ 7.5.2 WebLogic JMS 8.1.6, 9.2.3, 10
Sun Solaris	Solaris 9, 10 SPARC	WebSphere MQ 6.0 (32-bit and 64-bit) [1] TIBCO EMS 4.2.0, 4.4.2 SonicMQ 7.5.2 WebLogic JMS 8.1.6, 9.2.3, 10
IBM AIX	AIX 5L 5.3, 6.1 POWER	WebSphere MQ 6.0 (32-bit and 64-bit) [1] TIBCO EMS 4.2.0, 4.4.2 SonicMQ 7.5.2 WebLogic JMS 8.1.6, 9.2.3, 10
RedHat Linux	RedHat Enterprise Linux 5.0 WS/ES/AS x86 32 and 64-bit	WebSphere MQ 6.0(32-bit and 64-bit) [1] TIBCO EMS 4.2.0, 4.4.2 SonicMQ 7.5.2 WebLogic JMS 8.1.6, 9.2.3, 10

[1] WebSphere JMS, which is the JMS embedded in WebSphere Application Server, is not supported.

Supported JDBC Sensor Platforms

Platform	Database Version	Operating Environment
Oracle	9.2, 10g, RAC 10g	Windows 2003 Server 32 and 64-bit Solaris 9, 10 AIX 5L 5.3, 6.1 RedHat Enterprise Linux x86 5.0 32 and 64-bit
DB2	9.1	Windows 2003 Server 32 and 64-bit Solaris 9, 10 AIX 5L 5.3, 6.1 RedHat Enterprise Linux 5.0 x86 32 and 64-bit

Supported CICS Sensor Platforms

Platform	Operating Environment	WebSphere MQ
z/OS	z/OS 1.7, 1.8, 1.9 CICS TS 2.x, 3.x zSeries [1]	6.0, 7.0

[1] To run TransactionVision with CICS TS 3.2 on z/OS, will need to have PTF UK37779 applied (this PTF includes APAR PK66562 which actually addresses the problem). PTF UK37616 is also required, which is a prerequisite for UK37779.

Supported BEA Tuxedo Sensor Platforms

Platform	Operating Environment	BEA Tuxedo Version
Sun Solaris	Solaris 9, 10 SPARC	8.1 (32 & 64 bit)
IBM AIX	AIX 5L 5.3 POWER	8.1 (32 & 64 bit)
HP HP-UX	HP-UX 11i v2, v3 PA-RISC	8.1, 9.1 (32 & 64 bit)

Supported NonStop TMF Sensor Platforms

Platform	Operating Environment	TMF
NonStop	Guardian G06.29.02	T8652G08^10JUN2006^TMFCOM^AGL
NonStop	Guardian G06.30	T8608G08^08JAN2007^TMP^AGS

Supported .NET Agent Platforms

Platform	Operating Environment	.NET
Microsoft Windows	Windows 2003 Server x86 32 and 64-bit	1.1, 2.0, 3.0 [1]

[1] TransactionVision 8.00 supports .NET applications running against .NET 3.0 only if they are not using any new 3.0 specific API features.

Supported Browser Configurations

The browser configurations supported by the TransactionVision application are the same as those supported by Business Availability Center. See the *HP Business Availability Center Hardening Guide* PDF.

LDAP Support

TransactionVision LDAP support is managed by Business Availability Center. See the *HP Business Availability Center Hardening Guide* PDF.

Java Support

TransactionVision includes the Java 1.5 Runtime Environment for running the Analyzer and UI/Job Server components.

Supported JVMs for the JMS, JDBC, Servlet and EJB Agents match those versions distributed with WebSphere Application Server 5.1/6.0/6.1 and WebLogic Application Server 8.1.6/9.2.2/10.

The applets that display the Component Topology Analysis, Aggregated Topology and Instance Topology views use JRE 1.6.0_x (latest version is recommended).

Flash Player Support

Some TransactionVision reports and topologies require Adobe Flash Player. See the *HP Business Availability Center Hardening Guide* PDF for version information.

Localization and I18N Support

In TransactionVision 8.00, the User Interface has been localized for English, French, Japanese, Korean, and Simplified Chinese. TransactionVision 8.00 is I18N compliant and supports display of event fields, user data, and reports in non-English locales, as well as non-English project names and query names.

The following TransactionVision content is not localized:

➤ Product names, application names

➤ User-definable objects (such as jobs, filters, and communication links)

➤ TVisionSetupInfo and configuration files.

➤ Most date format.

Part II

Analyzer Installation and Configuration

3

Preparing to Install the TransactionVision Analyzer

This chapter includes:

➤ About the TransactionVision Analyzer on page 45

➤ The TransactionVision Analyzer in the Deployment Environment on page 46

➤ Overview of the Analyzer Installation and Configuration on page 46

About the TransactionVision Analyzer

The TransactionVision Analyzer communicates with TransactionVision Sensors and processes event data collected by the Sensors into meaningful analysis. The Analyzer runs as a windows service on Windows and as a daemon on Unix.

TransactionVision uses RMI (Remote Method Invocation) to communicate with the Analyzer on a specified port (default is 21100). The TransactionVision UI/Job Server as well as the command line utilities then use RMI to communicate to the Analyzer in order to control its behavior, initiating service shutdown, allowing collection to be started, stopped, and getting status information.

The host on which the TransactionVision Analyzer is installed must have a static IP address. When the Analyzer is registered to a TransactionVision project, its hostname is resolved to its underlying IP address which is then used as a unique identifier for the Analyzer. For more information, see *Using TransactionVision*.

The TransactionVision Analyzer in the Deployment Environment

In most deployment environment environments, the TransactionVision Analyzer is installed on a separate host from the Business Availability Center Gateway Server and TransactionVision UI/Job Server.

For information about the system requirements for the host of Analyzer, see "Supported Analyzer Platforms" on page 31.

Overview of the Analyzer Installation and Configuration

The general process for installing and configuring the Analyzer is as follows:

1 Locate the host on which the Business Availability Center Gateway Server is installed. This will be the host on which you also install the Analyzer.

2 Review the system requirements for the Analyzer. See "Supported Analyzer Platforms" on page 31.

3 Install the Analyzer, optionally migrating configuration settings from a previous release.

For installing on a host running a Windows operating system, see "Installing the Analyzer on Windows" on page 49.

For installing on host running a UNIX operating system, see "Installing the Analyzer on UNIX Platforms" on page 53.

4 (Optional) Install a supported Messaging Middleware product if you don't want to use SonicMQ, which is included with TransactionVision.

5 Set up the database.

See the "Configuring Databases" chapter.

6 Run the **TVisionSetupInfo** utility.

See "Configuring the Analyzer" on page 69.

7 Perform any optional configuration as desired.

See "Additional Analyzer Configuration" on page 82.

8 Start the Analyzer.

See "Managing the Analyzer" on page 82.

4

Installing the Analyzer on Windows

This chapter includes:

➤ Starting the Analyzer Installation Program on Windows on page 49

➤ Initial Installation on page 50

➤ Upgrade Installation on page 51

➤ Uninstalling the Analyzer on page 51

Starting the Analyzer Installation Program on Windows

To install the Analyzer, perform the following steps:

1 Ensure that you are logged into the target system either as Administrator or as a user with Administrator privileges.

2 If you are upgrading the Analyzer from a previous release, be sure that the JAVA_HOME environment variable is set or ensure that Java is included in your path.

3 Close all Windows programs currently running on your computer, including automatic backup programs. Antivirus, antispyware, and threat protection programs do not need to be shut down.

4 From Windows Explorer, double-click **tvalzr_800_win.exe**. The InstallShield Welcome screen appears.

5 Click **Next**> and wait until the TransactionVision Setup Welcome screen appears.

6 If the InstallShield Save Files screen appears, click **Next**> to use the default folder for extracting installation files (for example, C:\TEMP\Hewlett-Packard\TransactionVision), or click **Change** to select the desired folder and click **Next**> to continue.

If this is the first time installing the TransactionVision Analyzer on this host, continue with "Initial Installation" on page 50. If an earlier version of the Analyzer is installed on this computer, continue with "Upgrade Installation" on page 51.

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 use the default installation folder (C:\Program Files\HP\TransactionVision), click **Next**>. To choose a different installation folder, click **Browse...**, select the desired installation folder, then click **Next**>.

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

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

Upgrade Installation

For an upgrade installation, the installation wizard displays the setup maintenance menu.

1 If you want to install TransactionVision 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 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 Click **Finish** to complete the installation.

4 After performing the upgrade installation it is required to migrate the existing database schemas with the **MigrateDB.bat** script. See Appendix A, "Utilities Reference" for information about this utility.

If you need to migrate large amounts of data see Appendix C, "Database Migration" for more detailed information about the migration process.

Uninstalling the Analyzer

To uninstall the Analyzer, perform the following steps:

1 From the Start menu, choose **Control Panel**.

2 Double-click **Add/Remove Programs**.

3 Select the HP TransactionVision package you want to uninstall 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 want 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**.

If you uninstall the servlet Sensor, it will be first turned off in the WebSphere Application Server the Sensor is monitoring. The instrumented classes under $WAS_HOME/classes will be removed along with its parent directories if they are empty.

6 The Uninstallation Complete screen appears. Click **Finish** to complete the uninstallation procedure.

Note: After uninstalling the TransactionVision web user interface, you must clean up its temporary cache and distribution directory. WebSphere does not do this automatically, and any old files could cause a new installation to work incorrectly.

5

Installing the Analyzer on UNIX Platforms

This chapter includes:

➤ Installation Files on page 53

➤ Starting the Analyzer Installation Program on UNIX on page 54

➤ Initial Installation on page 55

➤ Upgrade Installation on page 55

➤ Uninstalling the Analyzer on page 56

Installation Files

The following table shows the installation file names for the TransactionVision Analyzer package for each distributed platform.

Platform	File Name
AIX	tvalzr_800_aix_power.tar.gz
Linux	tvalzr_800_linux_x86.tar.gz
Solaris	tvalzr_800_sol_sparc.tar.gz

Starting the Analyzer Installation Program on UNIX

The Analyzer is installed to the following directory on Solaris and Linux: **/opt/HP/TransactionVisio**n. On AIX, the Analyzer is installed to: **/usr/lpp/ HP/TransactionVision**.

To install the TransactionVision Analyzer on UNIX platforms, perform the following steps.

1 Change to the directory location of the TransactionVision installation files (either a DVD device or download directory). NOTE: On Solaris and HP-UX, you must instead copy the installation files from the DVD device to a temporary directory on your host's hard drive.

2 Unzip and untar the packages for your platform; see "Installation Files" on page 53. For example:

```
gunzip tvalzr_800_sol_sparc.tar.gz
tar xvf tvalzr_800_sol_sparc.tar
```

3 Log in as superuser:

```
su
```

4 Enter the following command to begin the installation procedure:

./tvinstall_800_unix.sh

After the package is unzipped, a menu representing the available components is displayed. For example:

```
The following TransactionVision packages are available for installation:

1. TransactionVision Analyzer
2. TransactionVision UI/Job Server
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:
```

If this is the first time installing TransactionVision components on this computer, continue with Initial Installation. If an earlier version of TransactionVision is installed on this computer, continue with "Upgrade Installation" on page 55.

Initial Installation

1 Type 1 and press Return.

The installation script installs the package, and displays the following message:

Installation of <TVANLZR> was successful.

The TransactionVision component menu is displayed.

2 Type **q** and press **Return** to quit the installation process.

To install additional components on this host, see the installation instructions for those components.

Upgrade Installation

1 Type 1 and press Return.

If the installation script determines that a previous version of the Analyzer is installed, it displays the following message:

There is an earlier version of TransactionVision installed on the system.
The earlier version has to be uninstalled before installing the current package(s).
Continue with the uninstallation? (y/n) [n]:

2 Type **Y** and press **Return** to uninstall the previous version. The following prompt is displayed:

Installation has detected a previous installation of TransactionVision.
Do you want to migrate existing TransactionVision configuration files? (y/n) [y] :

3 Type **Y** and press **Return** to migrate configuration to the new version. The installation script automatically creates a backup copy of existing Analyzer configuration files for the migration. It then uninstalls the previous version of the Analyzer and displays the TransactionVision component menu. Continue to step 4.

If you type **N** and press **Return**, the installation script provides the option of making a backup copy of configuration files for future reference:

> Although migration will not be performed at this time, you may optionally back up configuration files from your previous installation for reference purposes. Note that answering N will overwrite these files, causing any existing setup information to be lost. Do you wish to back up configuration files from the previous installation? (y/n) [y] :

Type **Y** and press **Return** to create a backup copy of your configuration files. The installation script prompts you to specify a backup location:

> Enter the directory to which existing TransactionVision configuration files should be backed up [/opt/TVision/migrate_tv800_yyyymmdd_HHMMSS]:

Press **Return** to use the default location, or enter the desired backup directory location. The installation utility performs the following tasks:

➤ Copies the current configuration files to the specified directory.

➤ Uninstalls the previous version of TransactionVision.

➤ Displays the TransactionVision component menu again.

4 Type **q** and press **Return** to quit the installation procedure.

5 Migrate the database schemas with the **MigrateDB.sh** script. See Appendix A, "Utilities Reference" for information about this utility.

To migrate large amounts of data, also see Appendix C, "Database Migration", for more detailed information about the migration process.

Uninstalling the Analyzer

To uninstall TransactionVision components, perform the following steps:

1 Log in as superuser:

```
su
```

2 Enter the following command:

./tvinstall_800_unix.sh -u

The following menu is displayed (note that actual options depend on the TransactionVision components that are installed.

```
The following TransactionVision packages are available for installation:

1. TransactionVision Analyzer
2. TransactionVision UI/Job Server
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:
```

Note: Note that actual options and numbers depend on the installation files available on your computer.

3 Enter **1** and press **Return**.

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

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

4 Type **q** and press **Return** to quit the installation

6

Configuring Databases

This chapter includes:

About Configuring Databases

Before configuring the TransactionVision Analyzer, it is important to ensure that your DBMS environment is configured properly to work with TransactionVision. This configuration differs for DB2, Oracle, and SQL Server databases.

The **TVisionSetupInfo** script does not create a database and relies on a connection to an existing database. Database variables or tuning parameters must then be adjusted to ensure that the database will operate at an acceptable level of performance for your application. Several tools are provided with TransactionVision to aid in the identification and resolution of database performance bottlenecks.

Supported Databases

The following databases and associated platforms are supported by
TransactionVision.

➤ DB2 9.1

➤ Oracle 9.2

➤ Oracle 10g

➤ Oracle RAC 10g

➤ Microsoft SQL Server 2005

Configuring Database Access

TransactionVision is connecting to the database via JDBC. When you run
the TVisionSetupInfo utility to configure the TransactionVision Analyzer
and UI/Job Server, you are prompted for the database type (Oracle, DB2, SQL
Server), host name, database name, database port, user name, and password.
This input will modify the settings in <TVISION_HOME>/config/datamgr/
Database.properties which are used for establishing the JDBC connection to
the database. The user password will be stored in this file in encrypted form.

TransactionVision uses third-party JDBC drivers from DataDirect for
accessing the database. If there is a need to use the original vendor JDBC
drivers instead, it is possible to manually configure the JDBC driver used by
TransactionVision.

To manually configure the JDBC driver used by TransactionVision:

1 Edit the following properties in <TVISION_HOME>/config/datamgr/
Database.properties:

➤ set the property 'jdbc_driver' to the JDBC driver class. The property file
contains several example settings for the most common vendor drivers

➤ set the property 'jdbc_url' to the full JDBC connection URL. The property
file contains several URL examples for the most common vendor drivers

For example, to configure TransactionVision to use the DB2 universal JDBC driver, set the following properties:

```
jdbc_driver=com.ibm.db2.jcc.DB2Driver
jdbc_url=jdbc:db2://host:port/
dbname:retrieveMessagesFromServerOnGetMessage=true;
```

2 Add the JDBC driver jars to the TransactionVision CLASSPATH. This can be done by adding the jar files to the custom CLASSPATH when prompted so from TVisionSetupInfo:

The Analyzer can optionally be customized by plugin beans in JAR files.

The location of these JAR files needs to be added to the Analyzer CLASSPATH.

Please specify a semicolon delimited list of any additional JAR files you wish to be added to the CLASSPATH. (for example, 'C:\TVision\myext.jar;C:\TVision\myutil.jar')]: C:\oracle\10g\jdbc\lib\ojdbc14.jar

This will add the JDBC driver jars to the CLASSPATH setting in SetupEnv.bat/.sh.

Setting DB2 Variables

Before using TransactionVision, set the values for the following DB2 variables to the values shown in the following table:

Variable	Value	Description
APP CTL HEAP SZ	1024	Maximum application control heap size. This value indicates the number of 4KB blocks.
MAXAPPLS	150	Maximum number of active applications
APPLHEAPSZ	1024	Default application heap size. this value indicates the number of 4KB blocks.
LOCKLIST	Estimate based on system	Amount of storage (in 4KB units) allocated to the lock list.

In addition, you should increase the size of the bufferpool.

APP CTL HEAP SZ, MAXAPPLS, and APPLHEAPSZ

Use the following commands to set these values. Note that the last three commands will drop all active database connections and then stop and start the DB2 server. Be sure to run these steps at an appropriate time when other database users will not be affected.

```
db2 connect to tvision
db2 get db cfg for tvision
db2 update db cfg for tvision using APP_CTL_HEAP_SZ 1024
db2 update db cfg for tvision using MAXAPPLS 150
db2 update db cfg for tvision using APPLHEAPSZ 1024
db2 force application all
db2stop
db2start
```

LOCKLIST

To determine the number of pages required for your lock list, perform the following steps:

1 Calculate a lower bound for the size of your lock list:

(512 * 36 * mapappls) / 4096

where 512 is an estimate of the average number of locks per application and 36 is the number of bytes required for each lock against an object that has an existing lock.

2 Calculate an upper bound for the size of your lock list:

(512 * 72 * maxappls) / 4096

where 72 is the number of bytes required for the first lock against an object.

3 Estimate the amount of concurrency you will have against your data. Based on you expectations, choose an initial value for LOCKLIST that falls between the upper and lower bounds that you have calculated. Use the following command to set the value:

db2 update db cfg for *database_name* using LOCKLIST *n*

where n is the value for LOCKLIST.

4 Using the database system monitor, tune the value of this parameter. For more information, see Chapter 32, "Configuring DB2 Capacity Management," in the *DB2 V7 Administration Guide*.

Bufferpool

The size of the default DB2 bufferpool is only 250 pages. Increase it to 1024 (or more if sufficient memory for DB2 is available) to improve performance. The following command will change the size for the default bufferpool from 250 pages to 1024 pages:

```
db2 ALTER BUFFERPOOL IBMDEFAULTBP SIZE 1024
```

Setting Oracle Variables

If it is expected that TransactionVision will be used in a relatively simple environment where minimal database connections are required, no special configuration is required for the Oracle DBMS. However, environments where a large number of users will be simultaneously accessing the web user interface, several Analyzers will be in use, or where the Analyzer will incorporate higher than the default number of threads, it will be necessary to increase the Open Cursors database parameter. Related error messages may be expected to show up in the TransactionVision UI or Analyzer logs if this limit is exceeded. To increase the number of Open Cursors, execute the following command:

```
alter system set open_cursors = 600
```

This change may be made dynamically while the Oracle server is running. It is not necessary to restart the RDBMS.

DBMS Performance Tuning

Because TransactionVision uses the DBMS extensively for its data collecting and analyzing process, the performance of the DBMS is vital to the overall performance of TransactionVision. Inserting records and updating records represent the majority of the database operations associated with TransactionVision; therefore the speed of the physical disks/I/O interface has a significant impact on the performance.

Optimizing I/O Throughput

The key to DBMS performance is to overcome the operation bottleneck – I/O throughput limit. Usually this limit is imposed by the physical disk and the I/O interface.

Prior to deployment, it is imperative to make sure the actual DBMS system has a good I/O and disk subsystem attached and that the subsystem has been tuned for writing. This includes checking that the disk is RAID configured for performance, write-cache is enabled for the disks, and the I/O interface is fast (preferably fiber-optic interface).

To achieve high throughput of I/O, some forms of parallel processing should be used:

➤ Use separate DBMS instances for separate projects - if the data can be partitioned then separate DBMS instances on different hosts can be employed to achieve parallelism. This setup requires setting up multiple instances of TransactionVision.

➤ Use RAID disks for table space containers. RAID disks provide parallel I/O at hardware level.

➤ Separate table space containers and log file directories. DB2 log files, Oracle Rollback Segments, and SQL Server Transaction logs hold uncommitted database operations and usually are highly utilized during database insert/update. For this reason they should have their own containers on physically separated disks, and preferably on RAID disks.

➤ Stripe table spaces. If parallel I/O is not available at the hardware level, DBMS can be set up to span a tablespace across multiple disks, which introduces parallelism at the DBMS space management level.

There are many other database parameters that may impact the performance of TransactionVision. For DB2 in particular, those parameters previously mentioned in "Setting DB2 Variables" on page 61 must be examined one-by-one to ensure they are optimized.

There is also some benefit when the tablespaces used by TransactionVision are managed by the database directly (DMS or DMS RAW tablespaces).

Testing DBMS and Diagnosing Performance Bottlenecks

HP provides independent tools, DB2Test, OracleTest, and SQLServerTest that can be used to test the performance of DBMS relevant to TransactionVision (especially the record insert rate). The tool is written in Java and should be run where the TransactionVision Analyzer will be installed. For more information about these tools, see Appendix A, "Utilities Reference".

The tools simulate the database update operation generated by TransactionVision. Run the test multiple times to get a complete picture of the DBMS performance. Note that the result of the test does not directly correlate with TransactionVision processing rate; rather, it is an indicator of how well does the DBMS performance for the given configuration.

Make sure each test lasts at least a few minutes to minimize the overhead of any initialization process. The test should be run against the same database and table sets with which the TransactionVision Analyzer will be run.

➤ Run the insert test with one thread and with record size of 1KB - this will gauge the raw event insert performance.

➤ Run the insert test with multiple threads and with a record size of 1KB. This will test whether the insert can benefit from multiple threads. Usually the thread count is set to two times the number of CPUs.

➤ Run the insert test with one thread and with record size of (7 KB + average message size) - this will gauge the analyzed event insert performance.

➤ Run the insert test with multiple threads and with record size of (7 KB + average message size). This will test if the insert can benefit from multiple threads. Usually the thread count is set to two to four times the number of CPUs.

➤ If the Analyzer host and the DBMS host are different, the above tests should be run on the Analyzer host. However at least one test should be run on the DBMS host to see if there is any communication/DB client configuration related issues.

The rate of insert should be on par with the result achieved from similar systems tested by HP. During the test the following parameters of the DBMS system should be monitored:

➤ Disk I/O usage for all involved physical disks (tablespaces and log files), especially I/O busy percentage.

➤ CPU usage, including wait time percentage.

If a disk hits 80-90% utilization or the I/O wait time is extraordinary long, that's an indication of a disk I/O bottleneck. If no obvious bottleneck is seen, then a DBMS configuration or O/S configuration issue may exist. Check database, DBMS and kernel parameters with HP for any configuration issues.

Another useful tool for analyzing DBMS performance is the DB2 performance snapshot monitor.

Updating database statistics

Each database product provides tools for updating statistics about the physical characteristics of tables and the associated indexes. These characteristics include number of records, number of pages, and average record length. The database query optimizer uses these statistics when determining access paths to the data.

The database statistics should always get updated when tables have had many updates, such as when data is continuously collected into the database by the TransactionVision Analyzer.

It could result in large performance gains in queries made by TransactionVision views and reports, as well as queries made internally by the TransactionVision Analyzer to correlate events. It is recommended to use the functionality offered by the database product (for example, Automatic Maintenance in DB2 or the built-in job scheduler in Oracle) to regenerate the statistics on a regular basis. As an alternative, you can create SQL scripts for statistics generation with the TransactionVision CreateSqlScript utility and set up manual schedules as tasks with the task scheduler in Windows or as cron jobs in UNIX. See "CreateSqlScript" on page 328.

DBMS Disk Space Requirements

The other factor that needs to be finalized is the amount of disk storage space required for TransactionVision events. This is determined by the average size of the messages, the rate of events collected by TransactionVision, and the duration of which TransactionVision event data needs to be kept in the database.

The formula for calculating disk storage usage is:

(Average message size + 7K Byte) x Event Rate x Event Retention Time

For example, if the average message size is 2K Bytes, the transaction rate is 5 transactions/second (for 8 hours/day and all weekdays, this translates into 720 thousand transactions/week). If TransactionVision data is required to be stored for the duration of four weeks before the data is either archived or deleted, then the total required storage is about 52 GB ((2 + 7)K Bytes x 720,000 x 2 x 4 = 52G Bytes).

Configuring Databases for Unicode Data

TransactionVision can display Unicode data in views and reports. However, you must create the TransactionVision database with the required code/ character set, and set the appropriate database property within TransactionVision.

Database Code/Character Set

When you create the TransactionVision database, you must specify the properties shown in the following table:

Database Provider	Required Settings
DB2	The TransactionVision database must be created with Code Set UTF-8.
Oracle	➤ The TransactionVision database must be created with the character set AL32UTF8 or UTF8. ➤ The NLS_LENGTH_SEMANTICS initialization parameter must be set to BYTE.
SQL Server	No special setting is required at database creation time.

TransactionVision Database Properties

In addition to setting the correct database character set at database creation time, the property 'unicode_db' property in Database.properties has to be set. All character-based XDM columns with the attribute unicode=true set will be generated with double the byte size to allow the specified number of characters to be stored in the database. For character sets requiring more than two bytes per character, set unicode_bytes_per_character to the required number of bytes per character in the Database.properties file. For more information on modifying the Database.properties files, see Appendix B, "Configuration Files".

7

Configuring the Analyzer

This chapter includes:

➤ About Configuring the Analyzer on page 69

➤ Files Modified by TVisionSetupInfo on page 70

➤ Information Required by TVisionSetupInfo on page 70

➤ Running TVisionSetupInfo on page 76

➤ Managing the Analyzer on page 82

➤ Additional Analyzer Configuration on page 82

➤ Reducing Event Database Size on page 93

About Configuring the Analyzer

TransactionVision provides the **TVisionSetupInfo** utility to guide you
through the Analyzer configuration process. This utility prompts you to
enter information it needs for the setup process and sets required
environment variables.

After running the utility, you can perform some optional configuration as
described in this chapter.

After all configuration is done, you must start the Analyzer as described in
this chapter.

Files Modified by TVisionSetupInfo

When configuring the Analyzer, the **TVisionSetupInfo** utility modifies the following files:

➤ TVISION_HOME/config/datamgr/Database.properties

➤ TVISION_HOME/config/setup/Setup.properties

In addition, **TVisionSetupInfo** does the following:

➤ Saves the installation path for software tools in **TVISION_HOME/config/ setup/DefaultInstallPath.xml**.

➤ Generates **TVISION_HOME/bin/SetupEnv.**[**sh**|**bat**], which is run by TVisionSetup to set the JAVA_HOME, CLASSPATH, and system library path environment variables required by TransactionVision.

Information Required by TVisionSetupInfo

The **TVisionSetupInfo** utility prompts you to enter the following information necessary to complete the configuration. You can review these information requirements before running **TVisionSetupInfo** so that you will be prepared to enter appropriate responses.

Licensing Information

You are prompted to enter your TransactionVision license.

If you want to apply the license after running **TVisionSetupInfo**, modify <TVISION_HOME>/config/license/License.properties.

Log File Location

You must supply the pathname of the directory where TransactionVision is to store the Analyzer log files. The default value is TVISION_HOME/logs. See Chapter 8, "Configuring Analyzer Logging".

Database Information

You must specify the type of database you plan to use with
TransactionVision (IBM DB2, Oracle, or SQL Server).

Note: The **TVisionSetupInfo** utility automatically sets up the DB2 Universal
JDBC driver (type 4) or the Oracle thin client driver. For instructions on
using other drivers, such as the DB2 Universal Driver (type 2) or the Oracle
oci driver, see "Running the Analyzer with a DB2 Type 2 or Oracle oci JDBC
Driver" on page 84.

The following table shows the database parameters that are collected for
each supported database type. More information about each parameter
follows the table.

	Database		
Parameter	**DB2**	**Oracle**	**SQL Server**
Log file location			
Database instance name	✓		✓
Database connection name	✓	✓	
Database name	✓	✓	✓
Database host	✓	✓	✓
Database user name	✓	✓	✓
Database user password	✓	✓	✓
Database listener port	✓	✓	✓
Database installation path	✓	✓	
JDBC driver installation path	✓	✓	✓

DB2 Database-Specific Information

For a DB2 database, you must supply the following information:

➤ The value of the DB2INSTANCE environment variable. The default value is DB2. Before starting your WebSphere 5 server, your DB2INSTANCE environment variable must be set. Failure to set this variable will cause a failure during TransactionVision startup and cause the WebSphere server to fail to start and initialize.

➤ The name of the database connection to be used by TransactionVision. See Chapter 6, "Configuring Databases" for more information on identifying the connection name and database name.

➤ The name of the database that TransactionVision will connect to. This name may be different from the "database_connection_name" if a client database connection is used. See Chapter 6, "Configuring Databases" for more information on identifying the connection name and database name.

➤ The name of the host that the database runs on. The default value is local_host.

➤ The user name for connecting to the database. If this field is empty, the currently logged in user is used to make the database connection. Make sure that the user specified or currently logged in has privileges for database access.

➤ The password for the database user name. If this field is empty, the currently logged in user's password is used to make the database connection. If TVisionSetupInfo is able to detect an installed JCE provider that supports the DES encryption algorithm, it sets the jce_provider value in the Database.properties file to the class name of the JCE provider and encrypts the password. Otherwise, it disables the password encryption feature and stores the password as unencrypted text.

➤ The database port number. This is the TCP/IP port the database server is listening on. A typical value is 50000. To identify the correct value, select the following menu items from the DB2 Control Center while the DB2 administration server is running:

All Systems > *system_name* > Instances > *instance_name*

Right-click and select **Setup Communications** from the context menu.
Click **Properties** to get the port number your database instance is using.

Oracle Database-Specific Information

For an Oracle database, you must supply the following information:

➤ The name of the database connection to be used by TransactionVision.
   The default value is TVISION. See Chapter 6, "Configuring Databases" for
   more information on identifying the connection name and database
   name.

➤ The name of the database that TransactionVision will connect to. The
   default value is TVISION. This name may be different from the
   "database_connection_name" if a client database connection is used. See
   Chapter 6, "Configuring Databases" for more information on identifying
   the connection name and database name.

➤ The name of the host that the database runs on. The default value is
   local_host.

➤ The user name for connecting to the database. If this field is empty, the
   currently logged in user is used to make the database connection. Make
   sure that the user specified or currently logged in has privileges for
   database access.

➤ The password for the database user name. If this field is empty, the
   currently logged in user's password is used to make the database
   connection. If TVisionSetupInfo is able to detect an installed JCE
   provider that supports the DES encryption algorithm, it sets the
   jce_provider value in the Database.properties file to the class name of the
   JCE provider and encrypts the password. Otherwise, it disables the
   password encryption feature and stores the password as unencrypted
   text.

➤ The database port number. This is the TCP/IP port the database server is
   listening on. A typical value is 1521. To identify the correct value, run the
   lsnrctl command on the server side. Use the status command to find out
   the port number.

SQL Server Database-Specific Information

For an SQL Server database, you must supply the following information:

➤ The name of the host that the database runs on. The default value is local_host.

➤ The name of the SQL Server instance. This should typically be empty unless you are running multiple server instances.

➤ The port number to connect to on the SQL Server

➤ The name of the SQL Server database that TransactionVision will connect to. The default value is TVISION.

➤ The user name for connecting to the database. If this field is empty, the currently logged in user is used to make the database connection. Make sure that the user specified or currently logged in has privileges for database access.

➤ The password for the database user name. If this field is empty, the currently logged in user's password is used to make the database connection. If TVisionSetupInfo is able to detect an installed JCE provider that supports the DES encryption algorithm, it sets the jce_provider value in the Database.properties file to the class name of the JCE provider and encrypts the password. Otherwise, it disables the password encryption feature and stores the password as unencrypted text.

LW-SSO (Lightweight Single Sign-on) Settings

Specify LW-SSO settings; required for authorizing access from Business Availability Center.

Message Middleware Provider Information

TransactionVision provides SonicMQ for communication between Sensors and the Analyzer and the Progress SonicMQ software is installed when the Analyzer is installed. If you are using this included version of SonicMQ, you do not need to enter any message system provider information.

Note: The built-in SonicMQ must be the messaging system provider for BEA Tuxedo or Non-Stop TMF Sensors, or for integration with RUM.

TIBCO EMS, WebSphere MQ, and SonicMQ as a separate installation are also supported and one of these can be specified as the messaging middleware provider. If you are using one of these message system providers, you are prompted for the installation path and additional information as described below.

WebSphere MQ Setup Verification

If you are using WebSphere MQ as the messaging middleware provider, you are prompted to verify that the Analyzer is able to communicate with the specified queue manager. You may test a server or client connection.

You will need the following information

➤ Host name for WebSphere MQ queue manager

➤ Channel name for WebSphere MQ queue manager

➤ Port number for WebSphere MQ queue manager

➤ CCSID for WebSphere MQ queue manager (leave blank if unknown)

TIBCO Setup Verification

If you are using TIBCO as the messaging middleware provider, you are prompted to verify that the Analyzer is able to communicate with the specified host. You will need the following information:

➤ TIBCO EMS Server Host Name

➤ TIBCO EMS Server Port Number

➤ User Name

➤ Password

Agent and Sensor Types

User Event, .NET, NonStop TMF and Tuxedo events are always collected, but event collection from other technologies can be enabled/disabled.

Specify the types of Sensors you are using in your environment and for which events is enabled. Options are:

➤ CICS

➤ EJB

➤ Servlet

➤ TIBCO EMS, SonicMQ JMS or WebLogic JMS

➤ WebSphere MQ

➤ WebSphere MQ IMS Bridge

➤ WebSphere MQ JMS

➤ JDBC

➤ All of the above

Plugin Beans to Customize the Analyzer

The location of any JAR files that need to be added to the Analyzer CLASSPATH. These JAR files contain plugin beans that customize the Analyzer.

Registration with Business Availability Center

You must specify the location of the Business Availability Center Gateway Server used by the TransactionVision components.

Running TVisionSetupInfo

1 Ensure that you are logged into the target system either as root, Administrator, or as a user with Administrator privileges.

2 Enter TVisionSetupInfo.[sh|bat] to run the TVisionSetupInfo script.

Operating System	Script Command
AIX, Linux, Solaris	<TVISION_HOME>/bin/TVisionSetupInfo.sh
Windows	<TVISION_HOME>\bin\TVisionSetupInfo.bat

Note: The installation sets the TVISION_HOME environment variable to the absolute path of the installation directory. For example, on Solaris, TVISION_HOME would be /opt/HP/TransactionVision; on AIX it would be /usr/lpp/HP/TransactionVision.

TVisionSetupInfo prompts you to enter information required to configure the following:

➤ Log files

➤ Database properties

➤ 3rd-party messaging middleware provider information

➤ Agents and Sensors in the target environment

➤ Registration with Business Availability Center

When responding to prompts from TVisionSetupInfo, press **Enter** to accept the default value shown in brackets; otherwise, type the correct value and press **Enter**. To specify an empty value, press the spacebar, and then press **Enter**. In the following sections, sample input is shown in italics.

As **TVisionSetupInfo** completes each configuration section, it displays messages indicating which files have been updated.

Example

The following shows an example session of **TVisionSetupInfo** on Windows.

```
This program collects configuration information in order to set up the TransactionVision
environment. This includes:
- Location of software that TransactionVision depends upon such as the
 messaging middleware and the relational database system
- Parameters required to connect to the messaging middleware
- Parameters to connect to the database system
- Setup parameters for installed TransactionVision components

You will be prompted to input required configuration parameters.
If a default value is provided in [], pressing <Enter> will set the parameter to this default
value. Pressing <Space><Enter> will set the parameter to an empty value.

Please specify name of the directory where you want to store your log files
 [C:/PROGRA~1/HP/TRANSA~1\logs]:
TransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\config\setup\Setup.properties" has been successfully
updated
Modifying *.Logging.xml files to use log file directory C:/PROGRA~1/HP/
TRANSA~1\logsTransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\config\logging\bpitveventimporter_loggingconfig.prope
rties" has been successfully updated

Please provide your TransactionVision license key: PreSalesTraining@hp.com-020EF
5-815082021D0482C
TransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\config\license\License.properties" has been
successfully updated

-----------------------------------------------------------
 Database Settings
-----------------------------------------------------------

Retrieving database configuration parameters...

  Type of Database? (DB2/Oracle/SQLServer) [Oracle]:

  Name of the host the database is running on [your_database_host]: ros89891duh.
rose.hp.com

  Name of database TransactionVision connects to.
```

This is the Oracle database SID [your_database_name]: orcl

Enter the port number that the database listener is on [1521]:

Database user name []: system

User password []: rpmtest
TransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\config\datamgr\Database.properties" has been
successfully updated

LW-SSO (Lightweight Single Sign-On) Settings

Enter the LW-SSO init string, used for the initialization of the enc algorithm. This value
needs to be the same in all applications integ with LW-SSO. Unless you have modified
the default value in other app such as BAC, you can just accept the default. [Xy6stqZ]:

Enter the LW-SSO application domain, used for LW-SSO cookie creation.This is the full
or partial domain part of the fully-qualified hostn of the machine running the
TransactionVision UI. (for example, 'mydomain.com') [hpqcorp.net]:

Enter a comma delimited list of LW-SSO protected domains. Applicable if other
applications integrated with LW-SSO are running in different domains. It is not
necessary to include the TransactionVision UI doma (for example, if BAC is in domain
bac.com and Diagnostics is in domadiag.com, enter
'bac.com,diag.com'):TransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\cproperties" has been successfully updated

Generating script file for environment setup...
TransactionVision Info(NewFileCreated): File "C:\PROGRA~1\HP\TRANSA~nv.bat"
has been successfully created

Initialize and verify database setup for TransactionVision

Do you wish to initialize the TransactionVision Database and add communication links
for SonicMQ on this Analyzer ? [y]:

Default TVISION commlink has been added

Default HTTP commlink has been added

```
Verifying database initialization for TransactionVision...
TransactionVision Info(DBInitialized): Database has been properly initialized

------------------------------------------------------------
 Register TransactionVision with Business Availability Center
------------------------------------------------------------

Do you wish to register TransactionVision with Business Availability
Center at this time? [n]: y

Note: If your BAC server has Lightweight Single Sign On enabled, the hostnames of the
TransactionVision UI and BAC servers must be a fully qualified domain name.

Please enter the hostname of the BAC server [] : ovresx1-vm11.rose.hp.com

Please enter the port of the BAC server [80] :

Do you wish to configure TransactionVision to access BAC using HTTPS? [n] :

TransactionVision has been configured to use the BAC server at: http://
ovresx1-vm11.rose.hp.com:80

Do you wish to register the TransactionVision UI with BAC? [y] :

Note: If your BAC server has Lightweight Single Sign On enabled, the hostnames of the
TransactionVision UI and BAC servers must be a fully qualified domain name.

Enter TransactionVision UI host [ROS89891DUH.americas.hpqcorp.net] :

Do you wish to configure BAC to access TransactionVision using HTTPS? [n] :

Enter TransactionVision UI port [21000] :

Do you wish to configure BAC to access TransactionVision using HTTPS? [n] :

Enter TransactionVision UI port [21000] :

Do you wish to register a TransactionVision Analyzer for RUM Data Publishing with
BAC? [y] :

Enter the TransactionVision Analyzer host for RUM Data Publishing
[ROS89891DUH.americas.hpqcorp.net] :

Enter RUM data publishing port [21113] :
```

User name for Basic Authentication to TransactionVision Analyzer
used by Real User Monitor :

Password for Basic Authentication to TransactionVision Analyzer used by Real User
Monitor :

The BAC server at http://ovresx1-vm11.rose.hp.com:80 will be configured with:

TransactionVision Web Server: http://
ROS89891DUH.americas.hpqcorp.net:21000
TransactionVision URL for RUM: http://
ROS89891DUH.americas.hpqcorp.net:21113/tv_rum
TransactionVision User for RUM:
TransactionVision Password for RUM:

Please review the above settings a verify that they are correct.

Proceed with registration to BAC? [y] :

Registration with BAC was successful.

TransactionVision Info(TVisionSetupInfoSuccess): TVisionSetupInfo has completed
successfully.

All program output and user input has been logged to
"C:/PROGRA~1/HP/TRANSA~1/logs/setup.log".

Please start HP Business Availability Center by running
%TVISION_HOME%\bin\SupervisorStart.bat

Completion Information

Upon completion, TVisionSetupInfo displays the following messages:

TransactionVision Info(TVisionSetupInfoSuccess): TVisionSetupInfo has completed
successfully.

Restarting Business Availability Center

Business Availability Center may need to be restarted after the **TVisionSetupInfo** script is run. If this is necessary, the **TVisionSetupInfo** script displays the appropriate command to use as follows:

On Windows:

Please start HP Business Availability Center by running
%TVISION_HOME%\bin\SupevisorStart.bat

On UNIX:

Please start HP Business Availability Center by running $TVISION_HOME/bin/
run_topaz start

Managing the Analyzer

The Analyzer service is managed by the "HP Business Availability Center service. See the the *HP Business Availability Center Deployment Guide* PDF.

To exit the Analyzer without sending a stop collection message to the Sensor (so that the Sensor continues to collect events), use the following command:

ServicesManager -exit -keepcollect

Note that the Sensor only continues to collect events until the last configuration message has expired, since there will not be any new configuration messages sent by the Analyzer. You can change the configuration message expiry on the Analyzer settings page in the TransactionVision UI.

Additional Analyzer Configuration

The following configurations can be performed as needed.

➤ Changing Heap Settings for SonicMQ Broker

➤ Message Expiring Configuration

➤ Set Analyzer Thread Count

➤ Error Logging

➤ Running the Analyzer with a DB2 Type 2 or Oracle oci JDBC Driver

➤ Multithreaded Servlet/JMS Events

➤ Disabling Unneeded Analyzer Beans

➤ Local Transaction Matching

➤ Analyzer Failure Mode

➤ JDBC Sensor Database Resolution

➤ Optimizing for .NET Agents

Changing Heap Settings for SonicMQ Broker

1 Start the SonicMQ Management Console using the script:

<TVISION_HOME>\Sonic\MQ7.5\bin\startmc.bat (Windows)

<TVISION_HOME>/Sonic/MQ7.5/bin/startmc.sh (UNIX)

2 Click on the **Configure** tab near the top of the SonicMQ Management Console Window.

3 Click on the second '+' button named 'Containers' to expand the list of Containers configured in SonicMQ.

4 Select the Container named the same as the hostname and right-click to view the drop-down menu; select **Properties**.

5 On the Edit Container Properties Window, press the tab named "Environment".

6 In the box named "Java VM Options" increase the heap size setting by changing the default "-Xms32m -Xmx256m".

7 Exit out of the Management Console.

8 Shutdown the nanny using the command:

<TVISION_HOME>\bin\SupervisorStop.bat (Windows)

<TVISION_HOME>/bin/run_topaz stop (UNIX)

9 Restart the nanny using the command:

<TVISION_HOME>\bin\SupervisorStart.bat (Windows)

<TVISION_HOME/bin/run_topaz start (UNIX)

For more information about setting the SonicMQ Broker heap size, see the *Progress SonicMQ Performance Tuning Guide*.

Message Expiring Configuration

Configuration Message Expiry should be set to a value small enough so that the amount of events generated by an "orphaned" configuration message can fit into the event queues without causing major production issues. For more information, see *Using TransactionVision*.

Set Analyzer Thread Count

Analyzer thread count should be set to match the test results from the DBMS insert test. For more information, see *Using TransactionVision*.

Error Logging

Once the Analyzer is running, it logs its errors to TVISION_HOME/logs/ analyzer.log. If an error occurs while initializing the service portion of the Analyzer, however, they can be found in the analyzer_startup.log file.

Running the Analyzer with a DB2 Type 2 or Oracle oci JDBC Driver

The TVisionSetupInfo utility sets up the DB2 Universal JDBC driver(type 4) or the Oracle thin client driver. To use other drivers such as the DB2 Universal Driver(type 2) or the Oracle oci driver, perform the following steps:

1 Run TVisionSetupInfo at least once to input required parameters such as the database host name, database name, etc.

2 To use the DB2 Universal Driver(type 2), add the property jdbc_url in the property file <TVISION_HOME>/config/datamgr/ Database.properties in the following format:

    jdbc:db2:<database-name>,

where <database-connection-name> is the database alias name used by the Analyzer. Refer to "Configuring Database Access" on page 60 to determine the database connection name.

3 To use the Oracle oci driver, add the property jdbc_url in the property file <TVISION_HOME>/config/datamgr/ Database.properties in the following format:

> jdbc:oracle:oci:<user>/<password>@<database-name>

Where <database-connection-name> is the Oracle Net service name used by the Analyzer, and <user>, <password> are the user name and passwords required for the Oracle connection. Refer to "Configuring Database Access" on page 60 to determine the database connection name.

4 Once the jdbc_url property has been changed, both the Analyzer and the application server will use the property value to connect to the database. Application servers such as WebSphere or WebLogic use 32-bit JVMs and do not require any changes to the library path environment variable.

Multithreaded Servlet/JMS Events

By default, if a servlet spins off a thread to make some JMS calls, the servlet passes tracking information to the child thread. The result is that both the servlet and JMS events belong to the same business transaction. However, there may be some cases in which you wish to separate these events into different transactions. For example, a servlet may spin off a long-running thread that you do not want to be part of the same transaction as the servlet.

To change the default behavior so that the child thread is not considered by TransactionVision to be part of the same business transaction as the servlet that spins it, perform the following steps:

1 Open the <TVISION_HOME>/config/services/Analyzer.properties file.

2 Add the name of the program that spins off threads to the separate_child_thread_txns property, as in the following example:

> separate_child_thread_txns=program1, program2

Separate multiple program names with a comma.

3 Close and save Analyzer.properties.

4 Restart the Analyzer.

Disabling Unneeded Analyzer Beans

If you configured the Analyzer for JMS event collection during TVisionSetupInfo, the JMSPubSubRelationBean will get enabled by default. This bean is needed to correlate JMS Publish and Subscribe events during event analysis.

If you:

➤ do not collect any PubSub JMS events from your Sensors, AND -

➤ require maximum performance of the Analyzer, then you can disable this bean to avoid the overhead related to handling the Publish/Subscribe mechanism.

To Disable the JMSPubSubRelationBean

Edit <**TVISION_HOME**>/**config/Beans.xml** in the following way:

```
<!--  this bean is used to correlate Publish/Subscribe events -->
<Module type="Bean"
class="com.bristol.tvision.services.analysis.eventanalysis.JMSPubSubRel
ationshipBean"/>
```
To:

```
<!--  this bean is used to correlate Publish/Subscribe events -->
<!--Module type="Bean"
class="com.bristol.tvision.services.analysis.eventanalysis.JMSPubSubRel
ationshipBean"/-->
```

Local Transaction Matching

By default, the Analyzer uses strict local transaction matching to group WebSphere MQ events into local transactions. However, there may be times when you want to disable strict local transaction matching and instead use the default MQ local transaction bean, which groups events into local transactions according to unit of work.

To disable strict local transaction matching for certain WebSphere MQ events, you must define criteria in the TransactionVision MQStrictLocalTxnExclude.xml file. If an event matches any of the criteria specified in this file, the Analyzer will not put it into a separate local transaction, but will use the default MQ local transaction bean instead.

Criteria consist of a program name, queue manager, and object name combination. The following example disables strict local transaction matching for events that meet either of the following criteria:

➤ Program name amqcrsta, queue manager QM1, and object TEST.Q1

➤ Program name amqcrsta, queue manager ALT_QM, and object TEST.Q

```
<Criterias>
 <Match id="0">
   <value xpath="/Event/StdHeader/ProgramName">amqcrsta</value
   <value xpath="/Event/Technology/MQSeries/MQObject/@queueManager">
QM1</value>
   <value xpath="/Event/Technology/MQSeries/MQObject/@objectName">
TEST.Q1</value>
 </Match>

 <Match id="1">
   <value xpath="/Event/StdHeader/ProgramName">amqcrsta</value
   <value xpath="/Event/Technology/MQSeries/MQObject/@queueManager">
ALT_QM</value>
   <value xpath="/Event/Technology/MQSeries/MQObject/@objectName">
TEST.Q1/value>
 </Match>
</Criterias>
```

Analyzer Failure Mode

The Analyzer can operate in two modes: standard mode and failure mode. In failure mode, the Analyzer will only store event data of failed business transactions (or transactions violating SLA). For successful business transactions, only the corresponding business transaction rows will be saved. Using failure mode reduces the data storage requirement for a project. For more information about enabling failure mode, see *Using TransactionVision*.

JDBC Sensor Database Resolution

All JDBC events report the current database connection URL.

For DB2, the URL syntax can be one of the following:

➤ jdbc:db2:<db2 database alias> for type 2 driver.

➤ jdbc:db2://<host>:<port>/<db2 database alias> for type 4 driver.

For Oracle, the URL syntax may take the following forms:

➤ jdbc:oracle:thin:@//<host>:<port>/<service name> for type 4 driver.

➤ jdbc:oracle:thin:<host>:<port>:<SID> for type 4 driver.

➤ jdbc:oracle:oci:@<TNSName> for type 2 driver.

➤ jdbc:oracle:oci:@ for local default connection.

The JDBC Sensor uses the connection URL to identify which database the corresponding events are associated with. In many cases the URL contains all the information necessary to identify the database. This is typically the case with a type 4 style URL that contains the host, port and database name. In such a case making changes to this file is not needed. It is also possible that the URL does not have this information, then, in order to identify the database name the TransactionVision Analyzer follows a set of rules defined in the **JDBCSystemModelDefinition.xml**, Knowing whether you need to configure this file depends on how your application connects to the database, and whether it is desired that any database aliases that may be used are resolved to the actual name of the underlying database.

TVISION_HOME/config/services/ JDBCSystemModelDefinition.xml

This file contains data to associate JDBC connection URLs to database level objects. It contains these four sections:

➤ Default database instance name for different vendors (optional).

➤ A list of database instance objects in the monitored environment that can be referred to in the local default database and mapping section.

➤ A list of default local database connections. This applies to Oracle only.

➤ Mapping between JDBC URL attributes to the corresponding database.

Default Database Instance

This optional section defines the default database instance string to be used for a DB vendor and platform:

➤ <defdb2instwin>db2</defdb2instwin> This defines the default database instance name for DB2 on a Microsoft Windows platform.

➤ <defdb2inst>db2inst1</defdb2inst> This defines the default name for DB2 on all other platforms.

➤ <deforacleinst>1521</deforacleinst> This defines the default port number for the oracle instance.

If these entries do not exist, TransactionVision assumes the default values shown above.

Database Instance Objects

The second section defines the different database instances in the environment:

```
<dbinstance vendor="oracle" host=" myhost " name="1544">
  <protocol name="tcp">
    <param name="host" value=" myhost "/>
    <param name="port" value="1544"/>
  </protocol>
</dbinstance>
```

Each database instance element should have the following attributes: vendor (db2 or oracle), host, and instance name. The instance can have one or more protocol sections.

For DB2, the instance name is set to the DB2 instance. For Oracle, it is set to the port number of the listener process.

Default Local Database Connections

The third section defines the default database for the given host and database instance. Note that there should be a database instance definition (see above section) for the instance referred to in the local default database definition.

```
<localdefdb vendor="oracle" host="myhost" inst="1544"
        name="mydb"/>
```

An example use case of the default local database setting is if a JDBC program made a connection using a URL like ''jdbc:oracle:oci:@'. In this case TransactionVision has no way of identifying the database name being used, so it will look in the **JDBCSystemModelDefinition.xml** file and see what is defined as the default database for the host the event is occurring under. Assuming this was running on 'myhost' from the example above, it would associate this event with the Oracle database named 'mydb'.

JDBC URL Mapping

The fourth section defines the mapping between the database alias in the JDBC URL and the actual database name.

The mapping element looks like the following:

```
<mapping vendor="vendor">
  <src host="host name" dbalias="db alias"/>
  <dst host="host name" inst="instance name" dbname="db name"/>
</mapping>
```

➤ The vendor attribute's value can be set to "db2" or "oracle".

➤ The dbalias attribute in the <src> element, is set to the TNS name for Oracle, and database alias in DB2.

➤ The host of the <src> element is the host that the monitored application is running on.

➤ The instance attribute only is needed in DB2 mapping entries.

➤ The <dst> element contains the definition that a match will map to, and its host and instance attributes must match a <dbinstance> entry defined earlier.

For example, given the following mapping entry:

```
<mapping vendor="oracle">
<src host="apphost" dbalias="DBNAME"/>
<dst host="myhost" inst="1544" dbname="mydb"/>
</mapping>
```

In this situation, if a Sensored application running on 'apphost' used a JDBC URL connection such as 'jdbc:oracle:oci:@DBNAME', this would enable the Analyzer to resolve this database to the oracle 'mydb' database running on 'myhost'.

How a Database Name is Resolved

After defining rules in the **JDBCSystemModelDefinition.xml**, when the events are processed by the Analyzer it will follow a set of rules in determining the database name to report. The order of this resolution follows these steps:

Database Resolution for DB2 Type 2 Driver

1 Search the database mapping list and find if there is any entry with the source attributes matching the event host and database alias name.

2 If a matching entry can be found, create a database object using the information from the mapping destination section.

3 If no such entry can be found, create a database object using the event host, local database instance name (or default value if no such data exists in the event), and database alias name.

Database Resolution for DB2 Type 4 Driver

1 Follow Step 1 and 2 in the resolution instructions for DB2 type 2 driver.

2 If (1) fails, follow the below steps.

3 Parse the URL to retrieve the server name and listening port number.

4 Find an entry in the database instance list with matching host and port number.

5 If a matching entry can be found in (2), the event is to be associated with the found database instance. The database name is set to the one reported by the event URL.

6 If no such entry can be found in (2), and the event host is the same as the URL host, use the local database instance data from the event as database instance, or the default value if no such data exists in the event. Create a database object with the host, database instance, and URL database name as determined above.

7 If no such entry can be found in (2), and the event host is different from the URL host, create a database object using the URL host, default database instance name, and URL database name

Database Resolution for Oracle Type 2 Driver

1 If the URL contains a database name (TNS name), check if there is an entry in the database mapping list with the source host and database alias name matching the event host and database name.

2 If a matching entry can be found, use the mapping host section data (host, instance, database name) to create a database object for the event.

3 If no matching entry can be found, create a database object using (a) the JDBC event host, (b) default Oracle database instance (port), and (c) event database name.

4 If the URL contains no database name, check if there is an entry in the local default database section with matching host name. If more than one is found, use the first returned match. Create a database object using the host, database instance, and database name data from the matching entry.

5 If no matching entry can be found, create a database object using (a) the JDBC event host, (b) the default value for Oracle database instance, and (c) the string "default" as database name

Database Resolution for Oracle Type 4 Driver

1 No resolution is necessary in this case. The URL should provide the database host, listener port number, and SID/service name.

2 Check if the configuration data provides details on the Oracle database instance identified by the host and port number. Use the information if such record exists.

3 Create a database object in the system model table with the database instance and SID/service name information.

Optimizing for .NET Agents

If the Analyzer is used in a deployment environment that includes .NET Agents, event transport time and processing time can be improved by using WebSphere MQ as the messaging middleware provider instead of the SonicMQ messaging middleware included with TransactionVision.

Additionally, if the hits to the .NET application are sporadic, consider increasing the size of the event queue. For information about increasing the event queue size, see the documentation of the messaging middleware provider.

Reducing Event Database Size

TransactionVision provides an XML event compression bean. Use this bean to reduce the database size for each event.

Note: If the XML Event Compression bean is enabled, it is not possible to query on user buffer data.

After running TVisionSetupInfo, open the file <TVISION_HOME>/config/services/Beans.xml, locate and change the content as follows:

Original Segment:

```
<Module name="DBWriteEventCtx" type="Context">

  <!-- This context contains beans that write the XML event (or part of it) to the
database. -->
  <!-- Each registered bean in the chain is called. -->

  <Module
class="com.bristol.tvision.services.analysis.dbwrite.DBWriteEventDefaultBea
n" type="Bean"/>
```

```
   <!-- Replace the default bean with this one if you want ZIP compression for
the XML event -->
   <!--Module type="Bean"
class="com.bristol.tvision.services.analysis.dbwrite.DBWriteEventCompress
edBean" /-->
</Module>
```

Modified Segment:

```
<Module name="DBWriteEventCtx" type="Context">

   <!-- This context contains beans that write the XML event (or part of it) to the
database. -->
   <!-- Each registered bean in the chain is called. -->

   <!--<Module
class="com.bristol.tvision.services.analysis.dbwrite.DBWriteEventDefaultBea
n" type="Bean"/-->


   <!-- Replace the default bean with this one if you want ZIP compression for
the XML event -->
   Module type="Bean"
class="com.bristol.tvision.services.analysis.dbwrite.DBWriteEventCompress
edBean" />
</Module>
```

Note: Make sure only one bean is enabled at one time; otherwise, when both DBWriteEventDefaultBean and DBWriteEventCompressedBean are enabled, an exception will be thrown.

When using the compression bean, your DatabaseDef.xml must be updated to store the event data in BLOB format. The EVENT and EVENT_OVERFLOW table definitions must be updated, and the event_data column type changed from CLOB to BLOB.

After this change has been made you will need to restart your Analyzer and UI/Job Server, and create a new project for the changes to take effect.

The following example shows DatabaseDef.xml before the change to store event data in BLOB format:

```
<Table name="EVENT" volatile="true">
[…]
 <Column name="event_data" type="CLOB" size="1M"/>
[…]
 </Table>
 <Table name="EVENT_OVERFLOW" volatile="true">
 […]
 <Column name="event_data" type="CLOB" size="1M" notNull="true"/>
 […]
</Table>
```

8

Configuring Analyzer Logging

This chapter includes:

➤ Log Files on page 97

➤ Circular Logging on page 98

➤ Using Windows and UNIX System Logs on page 100

➤ Enabling SMTP Logging on page 101

➤ Enabling SNMP Logging on page 103

➤ Enabling JMS Logging on page 103

Log Files

By default, all TransactionVision components log error and warning messages to the appropriate log files. The location of log files is specified when you run TVisionSetupInfo or SensorSetup and stored in the Setup.properties file.

The Analyzer logs error messages to the analyzer.log file. On Windows, the Analyzer uses three additional log files:

➤ **analyzer_startup.log** contains information about the running of the Windows service portion of the Analyzer. It typically contains information about what options the Analyzer started under. If errors are encountered during the initializing of the service portion of the Analyzer, they can be found in this file.

➤ **analyzer_stderr.log** and **analyzer_stdout.log** represent the standard output and error of the Analyzer process. If you have custom analysis beans that print to the console or to standard error, you can find their output in these files. These files should also be referred to for further information if you see problems in starting or running the Analyzer and the standard analyzer.log file does not contain anything indicating an error.

Circular Logging

By default, the Analyzer employs 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 Analyzer log file analyzer.log, reaches 10 MB in size, it is renamed analyzer.log.1 and a new analyzer.log file is created. If you change the configuration so that there are two backup files, the following events take place when analyzer.log reaches 10 MB:

➤ analyzer.log.2 is removed if it exists.

➤ analyzer.log.1 is renamed analyzer.log.2.

➤ analyzer.log is renamed analyzer.log.1.

➤ A new analyzer.log is created.

If you do *not* want 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/Hewlett-Packard/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.

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.

Enabling SMTP Logging

The SMTPAppender sends an email to the SMTP server when an error log event at the specified threshold reaches the appender. To enable the SMTPAppender, add the following to your Analyzer.logging.xml file:

```
<appender name="EMAIL"
class="tvision.org.apache.log4j.net.SMTPAuthenticateAppender">
  <param name="SMTPHost" value="smtp.myserver.net"/>
  <param name="To" value="analyzer_log4j@myserver.net"/>
  <param name="From" value="administrator@myserver.net"/>
  <param name="UserName" value="smtp_user"/>
  <param name="Password" value=""/>
  <param name="Authenticate" value="true"/>
  <param name="BufferSize" value="1"/>
  <param name="Threshold" value="info"/>
  <layout class="tvision.org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d [%t] %-5p %c %x - %m%n"/>
  </layout>
</appender>
```

The threshold parameter specifies the logging level that is allowed to append into the SMTPAppender.

To define a customized triggering event evaluator, add the EvaluatorClass parameter:

```
<param name="EvaluatorClass" class="tvision.org.apache.
log4j.spi.TriggeringEventEvaluator"/>
```

This interface provides the following function for determining when an email should be sent:

```
public boolean isTriggeringEvent(LoggingEvent event) {
   long l = 0;
   synchronized(lock) {
    l = (msgCount ++);
   }
   return (((l + 1)%msgPkgSize) == 0); // fire email on every msgPkgSize events.
 }
```

Enabling SNMP Logging

The JoeSNMPTrapSender appender sends email when a specified error level occurs. To enable JoeSNMPTrapSender, add the following definition to the Analyzer.logging.xml file:

```
<!-- SNMP TRAP appender !-->
<appender name="TRAP_LOG"
class="tvision.org.apache.log4j.ext.SNMPTrapAppender">
  <param name="ImplementationClassName"
value="tvision.org.apache.log4j.ext.JoeSNMPTrapSender"/>
  <param name="ManagementHost" value="127.0.0.1"/>
  <param name="ManagementHostTrapListenPort" value="162"/>
  <param name="EnterpriseOID" value="1.3.6.1.4.1.24.0"/>
  <param name="LocalIPAddress" value="127.0.0.1"/>
  <param name="LocalTrapSendPort" value="161"/>
  <param name="GenericTrapType" value="6"/>
  <param name="SpecificTrapType" value="12345678"/>
  <param name="CommunityString" value="public"/>
  <param name="ForwardStackTraceWithTrap" value="true"/>
  <param name="Threshold" value="DEBUG"/>
  <param name="ApplicationTrapOID" value="1.3.6.1.4.1.24.12.10.22.64"/>
  <layout class="tvision.org.apache.log4j.PatternLayout">
  <param name="ConversionPattern" value="%d,%p,[%t],[%c],%m%n"/>
  </layout>
</appender>
```

Note: You must add joesnmp.jar to your CLASSPATH because it is required by JoeSNMPTrapSender. This JAR file can be downloaded from the JoeSNMP project at http://sourceforge.net/projects/joesnmp.

Enabling JMS Logging

TransactionVision provides a mechanism to send log messages via a JMS messaging provider. This is done through the com.bristol.tvision.appender.JMSAppender appender configured in the **Analyzer.logging.xml**.

The JMS appender can be configured one of two ways. Typically you use JNDI settings to configure the JMS connectivity, but direct WMQ JMS configuration is also allowed.

Choose whether you are configuring using:

➤ JNDI

-or

➤ Direct WMQ JMS.

Note: If both methods are specified, the WMQ JMS settings will take precedence and the JNDI settings are ignored.

In order to manually configure the BPI JMS connectivity, or to configure a separate logging facility to publish logs through JMS queues, use the following JMSAppender options:

➤ **ConnectionRetryDelay** is the time before a retry is made if connection fails.

➤ **ConnectionRetryTimeout** is the time it will wait before an unresponsive connection times out.

➤ **QueueName** is the name of the JNDI object (if JNDI is used), or the actual name of the queue (in the case of WMQ JMS).

➤ **Username** and **Password** are optional settings if authentication to the JMS provider is required.

For JNDI Settings

➤ **InitialContextFactoryName** is the classname of your JNDI context factory. This value will depend on which JMS vendor you use (see their documentation for details). Some examples are com.sun.jndi.fscontext.RefFSContextFactory, or com.tibco.tibjms.naming.TibjmsInitialContextFactory.

➤ **ProviderURL** is the url to connect to the JNDI repository, and depends on which JMS vendor you use. A RefFSContextFactory has a URL similar to file:/C:/jndi. For TIBCO, you might use something like tibjmsnaming://host:7222.

➤ **QueueConnectionFactoryName** is the name of the Queue Connection Factory JNDI object.

WMQ JMS Settings

The WMQ JMS specific settings correspond to the queue manager name, host, port and channel that you are using to connect to WMQ JMS. TargetMQClient enables/disables whether MQ uses RFH2 headers in its JMS message.

```
<appender class="com.bristol.tvision.appender.JMSAppender"
name="JMS_APPENDER">
  <!--connection retry interval in millseconds -->
 <param name="ConnectionRetryDelay" value="0"/>
 <param name="ConnectionRetryTimeout" value="0"/>
 <param name="QueueName" value=""/>
 <param name="UserName" value=""/>
 <param name="Password" value=""/>
<!-- enable the following to provide JNDI context parameters for
    JMS connection -->
<!--<param name="InitialContextFactoryName" value="" />
<param name="ProviderURL" value="" />
      <param name="QueueConnectionFactoryName" value=""/>
-->


<!-- enable the following to provide WebSphere MQ parameters for
     JMS connection -->
<!--
<param name="MQQueueManagerName" value="" />
<param name="MQClientConnectionHost" value="" />
<param name="MQClientConnectionPort" value="" />
<param name="MQClientConnectionChannel" value="" />
<param name="TargetMQClient" value="false"/>
->


 </appender>
```

Part III

UI/Job Server Installation and Configuration

9

Preparing to Install the TransactionVision UI/Job Server

This chapter includes:

➤ About the TransactionVision UI/Job Server on page 109

➤ The TransactionVision UI/Job Server in the Deployment Environment on page 109

About the TransactionVision UI/Job Server

The TransactionVision UI/Job Server is a web application bundled with the Apache Tomcat Servlet/JSP Container. The TransactionVision UI/Job Server supports communication between the TransactionVision Analyzer and the Business Availability Center Gateway Server.

The TransactionVision UI/Job Server in the Deployment Environment

In most deployment environments, the TransactionVision UI/Job Server should be installed on the same host on which the Business Availability Center Gateway Server is installed.

For information about the system requirements of the TransactionVision UI/Job Server, see "Supported TransactionVision UI/Job Server Platforms" on page 31.

10

Installing the UI/Job Server on UNIX Platforms

This chapter includes:

➤ Installation Files on page 111

➤ Starting the UI/Job Server Installation Program on UNIX on page 111

➤ Initial Installation on page 113

➤ Upgrade Installation on page 113

➤ Uninstalling the UI/Job Server on page 114

Installation Files

The following table shows the installation file names for the TransactionVision UI/Job Server package for the supported platform.

Platform	Files
Solaris	tvuijs_800_sol_sparc.tar.gz

Starting the UI/Job Server Installation Program on UNIX

The UI/Job Server is installed to the following directory on Solaris and Linux: **/opt/HP/TransactionVisio**n. On AIX, the UI/Job Server is installed to: **/usr/lpp/HP/TransactionVision**.

To install the TransactionVision UI/Job Server on UNIX platforms, perform the following steps.

1 Change to the directory location of the TransactionVision installation files (either a DVD device or download directory). NOTE: On Solaris and HP-UX, you must instead copy the installation files from the DVD device to a temporary directory on your host's hard drive.

2 Unzip and untar the packages for your platform; see "Installation Files" on page 111. For example:

```
gunzip tvuijs_800_sol_sparc.tar.gz t
```

3 Log in as superuser:

su

4 Enter the following command to begin the installation procedure:

./tvinstall_800_unix.sh

After the package is unzipped, a menu representing the available components is displayed. For example:

```
The following TransactionVision packages are available for installation:

1. TransactionVision Analyzer
2. TransactionVision UI/Job Server
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:
```

If this is the first time installing TransactionVision components on this computer, continue with Initial Installation. If an earlier version of TransactionVision is installed on this computer, continue with "Upgrade Installation" on page 113.

Initial Installation

1 Type 2 and press Return.

The installation script installs the package, and displays the following message:

> Installation of <TVUIJOB> was successful.

The TransactionVision component menu is displayed.

2 Type **q** and press **Return** to quit the installation process.

To install additional components on this host, see the installation instructions for those components.

Upgrade Installation

1 Type 1 and press Return.

If the installation script determines that a previous version of the UI/Job Server is installed, it displays the following message:

> There is an earlier version of TransactionVision installed on the system.
> The earlier version has to be uninstalled before installing the current package(s).
> Continue with the uninstallation? (Y/N) [N]:

2 Type **Y** and press **Return** to uninstall the previous version. The following prompt is displayed:

> Before uninstalling the previous version, TransactionVision provides the option of migrating configuration to the new installation:
> Installation has detected a previous installation of TransactionVision.
> Do you want to migrate existing TransactionVision configuration files? (y/n) [y] :

3 Type **Y** and press **Return** to migrate configuration to the new version. The installation script automatically creates a backup copy of existing UI/Job Server configuration files for the migration. It then uninstalls the previous version of the UI/Job Server and displays the TransactionVision component menu. Continue to step 4.

If you type **N** and press **Return**, the installation script provides the option of making a backup copy of configuration files for future reference:

Although migration will not be performed at this time, you may optionally back up configuration files from your previous installation for reference purposes. Note that answering N will overwrite these files, causing any existing setup information to be lost. Do you wish to back up configuration files from the previous installation? (y/n) [y] :

Type **Y** and press **Return** to create a backup copy of your configuration files. The installation script prompts you to specify a backup location:

Enter the directory to which existing TransactionVision configuration files should be backed up [/opt/TVision/migrate_tv780_date_time]:

Press **Return** to use the default location, or enter the desired backup directory location. The installation utility performs the following tasks:

➤ Copies the current configuration files to the specified directory.

➤ Uninstalls the previous version of TransactionVision.

➤ Displays the TransactionVision component menu again.

4 Type **q** and press **Return** to quit the installation procedure.

Uninstalling the UI/Job Server

To uninstall any TransactionVision component, perform the following steps:

1 Log in as superuser:

su

2 Enter the following command:

./tvinstall_800_unix.sh -u

The following menu is displayed (note that actual options depend on the TransactionVision components that are installed.

> The following TransactionVision packages are available for installation:
>
> 1. TransactionVision UI/Job Server
> 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:

Note: The options and numbers depend on the installation files available on your computer.

---

3 Enter **1** and press **Return**.

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

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

4 Type **q** and press **Return** to quit the installation

11

Installing the UI/Job Server on Windows

This chapter includes:

➤ Starting the UI/Job Server Installation Program on Windows on page 117

➤ Initial Installation on page 118

➤ Uninstalling the UI/Job Server on page 118

Starting the UI/Job Server Installation Program on Windows

To install the UI/Job Server, perform the following steps:

1 Ensure that you are logged into the target system either as Administrator or as a user with Administrator privileges.

2 If you are upgrading the UI/Job Server from a previous release, be sure that the JAVA_HOME environment variable is set or ensure that Java is included in your path.

3 Close all Windows programs currently running on your computer, including automatic backup programs. Antivirus, antispyware, and threat protection programs do not need to be shut down.

4 From Windows Explorer, double-click **tvuijs_800_win.exe**. The InstallShield Welcome screen appears.

5 Click **Next**> and wait until the TransactionVision Setup Welcome screen appears.

6 If the InstallShield Save Files screen appears, click **Next>** to use the default folder for extracting installation files (for example, C:\TEMP\Hewlett-Packard\TransactionVision), or click **Change** to select the desired folder and click **Next>** to continue.

If this is the first time installing the TransactionVision UI/Job Server on this host, continue with "Initial Installation" on page 118. If an earlier version of the Analyzer is installed on this computer, continue with "Uninstalling the UI/Job Server" on page 118.

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 use the default installation folder (C:\Program Files\HP\TransactionVision), click **Next>**. To choose a different installation folder, click **Browse...**, select the desired installation folder, then click **Next>**.

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

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

Uninstalling the UI/Job Server

To uninstall the Analyzer, perform the following steps:

1 From the Start menu, choose **Control Panel**.

2 Double-click **Add/Remove Programs**.

3 Select the HP TransactionVision package you want to uninstall 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 want 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**.

6 The Uninstallation Complete screen appears. Click **Finish** to complete the uninstallation procedure.

Note: After uninstalling the TransactionVision web user interface, you must clean up its temporary cache and distribution directory. WebSphere does not do this automatically, and any old files could cause a new installation to work incorrectly.

12

Configuring the UI/Job Server

This chapter includes:

About Configuring the UI/Job Server

TransactionVision provides the **TVisionSetupInfo** utility to guide you through the UI/Job Server configuration process. This utility prompts you to enter information it needs for the setup process and sets required environment variables.

Important: Configure the Analyzer before you configure the UI/Job Server. See Chapter 7, "Configuring the Analyzer".

After running the utility, you can perform some optional configuration as described in this chapter.

After all configuration is done, you must start the UI/Job Server as described in this chapter.

Files Modified by TVisionSetupInfo

When configuring the UI/Job Server, the **TVisionSetupInfo** utility modifies the following files:

➤ TVISION_HOME/config/datamgr/Database.properties

➤ TVISION_HOME/config/setup/Setup.properties

In addition, **TVisionSetupInfo** does the following:

➤ Saves the installation path for software tools in **TVISION_HOME/config/setup/DefaultInstallPath.xml**.

➤ Generates **TVISION_HOME/bin/SetupEnv.**[**sh**|**bat**], which is run by TVisionSetup to set the JAVA_HOME, CLASSPATH, and system library path environment variables required by TransactionVision.

Information Required by TVisionSetupInfo

The **TVisionSetupInfo** utility prompts you to enter the following information necessary to complete the configuration. You can review these information requirements before running **TVisionSetupInfo** so that you will be prepared to enter appropriate responses.

Log File Location

You must supply the pathname of the directory where TransactionVision is to store the UI/Job Server log files. The default value is TVISION_HOME/logs. See Chapter 8, "Configuring Analyzer Logging".

Database Information

You must specify the type of database you plan to use with TransactionVision (IBM DB2, Oracle, or SQL Server), as well the database-specific properties.

The information collected is the same information that is collected when setting up the Analyzer. See "Database Information" on page 71.

Registration with Business Availability Center

You must specify the location of the Business Availability Center Gateway Server used by the TransactionVision components.

Running TVisionSetupInfo

1 Ensure that you are logged into the target system either as root, Administrator, or as a user with Administrator privileges.

2 Enter TVisionSetupInfo.[sh|bat] to run the TVisionSetupInfo script.

Operating System	Script Command
AIX, Linux, Solaris	<TVISION_HOME>/bin/TVisionSetupInfo.sh
Windows	<TVISION_HOME>\bin\TVisionSetupInfo.bat

Note: The installation sets the TVISION_HOME environment variable to the absolute path of the installation directory. For example, on Solaris, TVISION_HOME would be /opt/HP/TransactionVision; on AIX it would be / usr/lpp/HP/TransactionVision.

TVisionSetupInfo prompts you to enter information required to configure the following:

➤ Log files

➤ Database properties

➤ Registration with Business Availability Center

When responding to prompts from TVisionSetupInfo, press **Enter** to accept the default value shown in brackets; otherwise, type the correct value and press **Enter**. To specify an empty value, press the spacebar, and then press **Enter**. In the following sections, sample input is shown in italics.

As **TVisionSetupInfo** completes each configuration section, it displays messages indicating which files have been updated.

Example

The following shows an example session of **TVisionSetupInfo** on Windows.

This program collects configuration information in order to set up the TransactionVision environment. This includes:
- Location of software that TransactionVision depends upon such as the
 messaging middleware and the relational database system
- Parameters required to connect to the messaging middleware
- Parameters to connect to the database system
- Setup parameters for installed TransactionVision components

You will be prompted to input required configuration parameters.
If a default value is provided in [], pressing <Enter> will set the parameter to this default value. Pressing <Space><Enter> will set the parameter to an empty value.

Please specify name of the directory where you want to store your log files
 [C:/PROGRA~1/HP/TRANSA~1\logs]:
TransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\config\setup\Setup.properties" has been successfully updated
Modifying *.Logging.xml files to use log file directory C:/PROGRA~1/HP/
TRANSA~1\logsTransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\config\logging\bpitveventimporter_loggingconfig.prope
rties" has been successfully updated

Please provide your TransactionVision license key: PreSalesTraining@hp.com-020EF
5-815082021D0482C
TransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\config\license\License.properties" has been
successfully updated

 Database Settings

Retrieving database configuration parameters...

  Type of Database? (DB2/Oracle/SQLServer) [Oracle]:

  Name of the host the database is running on [your_database_host]: ros89891duh.
rose.hp.com
Name of database TransactionVision connects to.
  This is the Oracle database SID [your_database_name]: orcl

Enter the port number that the database listener is on [1521]:

Database user name []: system

  User password []: rpmtest
TransactionVision Info(FileUpdated): File
"C:\PROGRA~1\HP\TRANSA~1\config\datamgr\Database.properties" has been
successfully updated

  LW-SSO (Lightweight Single Sign-On) Settings

Enter the LW-SSO init string, used for the initialization of the enc algorithm. This value
needs to be the same in all applications integ with LW-SSO. Unless you have modified
the default value in other app such as BAC, you can just accept the default. [Xy6stqZ]:

Enter the LW-SSO application domain, used for LW-SSO cookie creation.This is the full
or partial domain part of the fully-qualified hostn of the machine running the
TransactionVision UI. (for example, 'mydomain.com') [hpqcorp.net]:

Enter a comma delimited list of LW-SSO protected domains. Applicable if other
applications integrated with LW-SSO are running in differen domains. It is not necessary
to include the TransationVision UI doma (for example, if BAC is in domain bac.com and
Diagnostics is in domadiag.com, enter 'bac.com,diag.com'):TransactionVision
Info(FileUpdated): File "C:\PROGRA~1\HP\TRANSA~1\cproperties" has been
successfully updated

Generating script file for environment setup...
TransactionVision Info(NewFileCreated): File "C:\PROGRA~1\HP\TRANSA~nv.bat"
has been successfully created

  Initialize and verify database setup for TransactionVision

Do you wish to initialize the TransactionVision Database ? [y]:

Verifying database initialization for TransactionVision...TransactionVision
Info(DBInitialized): Database has been properly initialized

  Register TransactionVision with Business Availability Center
----
Do you wish to register TransactionVision with Business Availability
Center at this time? [n]: y

Note: If your BAC server has Lightweight Single Sign On enabled, the hostnames of the TransactionVision UI and BAC servers must be a fully qualified domain name.

Please enter the hostname of the BAC server [] : ovresx1-vm11.rose.hp.com

Please enter the port of the BAC server [80] :

Do you wish to configure TransactionVision to access BAC using HTTPS? [n] :

TransactionVision has been configured to use the BAC server at: http://ovresx1-vm11.rose.hp.com:80

Do you wish to register the TransactionVision UI with BAC? [y] :

Note: If your BAC server has Lightweight Single Sign On enabled, the hostnames of the TransactionVision UI and BAC servers must be a fully qualified domain name.

Enter TransactionVision UI host [ROS89891DUH.americas.hpqcorp.net] :

Do you wish to configure BAC to access TransactionVision using HTTPS? [n] :

Enter TransactionVision UI port [21000] :

Do you wish to configure BAC to access TransactionVision using HTTPS? [n] :

Enter TransactionVision UI port [21000] :

Do you wish to register a TransactionVision Analyzer for RUM Data Publishing with BAC? [y] :

Enter the TransactionVision Analyzer host for RUM Data Publishing [ROS89891DUH.americas.hpqcorp.net] :

Enter RUM data publishing port [21113] :

User name for Basic Authentication to TransactionVision Analyzer used by Real User Monitor :

Password for Basic Authentication to TransactionVision Analyzer used by Real User Monitor :

The BAC server at http://ovresx1-vm11.rose.hp.com:80 will be configured with:

TransactionVision Web Server: http://
ROS89891DUH.americas.hpqcorp.net:21000
TransactionVision URL for RUM: http://
ROS89891DUH.americas.hpqcorp.net:21113/tv_rum
TransactionVision User for RUM:
TransactionVision Password for RUM:

Please review the above settings a verify that they are correct.

Proceed with registration to BAC? [y] :

Registration with BAC was successful.

TransactionVision Info(TVisionSetupInfoSuccess): TVisionSetupInfo has completed successfully.

All program output and user input has been logged to
"C:/PROGRA~1/HP/TRANSA~1/logs/setup.log".

Please start HP Business Availability Center by running
%TVISION_HOME%\bin\SupervisorStart.bat

Completion Information

Upon completion, TVisionSetupInfo displays the following messages:

TransactionVision Info(TVisionSetupInfoSuccess): TVisionSetupInfo has completed successfully.

Restarting Business Availability Center

Business Availability Center may need to be restarted after the **TVisionSetupInfo** script is run. If this is necessary, the **TVisionSetupInfo** script displays the appropriate command to use as follows:

On Windows:

Please start HP Business Availability Center by running
%TVISION_HOME%\bin\SupevisorStart.bat

On UNIX:

Please start HP Business Availability Center by running $TVISION_HOME/bin/run_topaz start

Managing the UI/Job Server

The UI/Job Server service is managed by the HP Business Availability Center service. See the the *HP Business Availability Center Deployment Guide* PDF.

13

Configuring UI/Job Server Logging

This chapter includes:

➤ Log Files on page 129

➤ Circular Logging on page 129

➤ Trace Logging on page 131

➤ Using Windows and UNIX System Logs on page 131

Log Files

By default, all TransactionVision components log error and warning messages to the appropriate log files. The location of log files is specified when you run TVisionSetupInfo or SensorSetup and stored in the Setup.properties file.

The UI/Job Server logs error messages to the ui.log file.

Circular Logging

By default, the UI/Job Server employs 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 UI/Job Server log file ui.log, reaches 10 MB in size, it is renamed ui.log.1 and a new ui.log file is created. If you change the configuration so that there are two backup files, the following events take place when ui.log reaches 10 MB:

➤ ui.log.2 is removed if it exists.

➤ ui.log.1 is renamed ui.log.2.

➤ ui.log is renamed ui.log.1.

➤ A new ui.log is created.

If you do not want 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\ 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 Analyzer is doing internally. It is used mainly to troubleshoot problems and should not be turned on in production environments.

To enable trace logging for the TransactionVision web user interface, set the value of the trace property in the <TVISION_HOME>/config/ui/ UI.properties file to on. After modifying this configuration file, you must restart the web user interface for the change to take effect. To restart the TransactionVision web user interface, use your application server administration console.

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 catagory 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.

Part IV

Sensor and Agent Installation and Configuration

14

Preparing to Install TransactionVision Sensors

This chapter includes:

➤ Applications That Can Be Monitored on page 136

➤ Available Sensor and Agent Types on page 137

Applications That Can Be Monitored

In the following diagram, shaded areas represent the parts of a web application for which TransactionVision can track events.
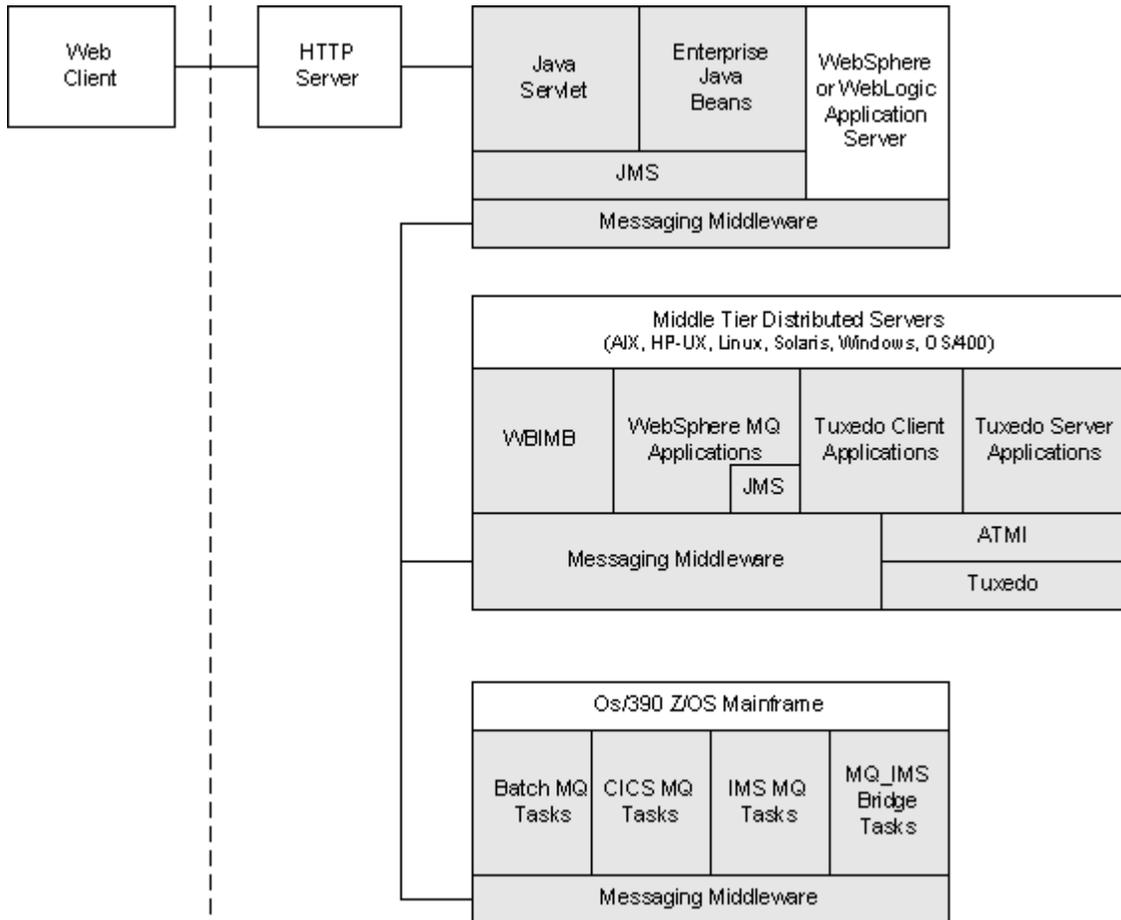
ASP.NET applications can also be monitored:
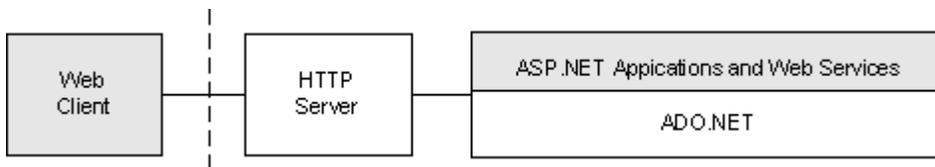
.NET Remoting client and server applications can also be monitored.

NonStop TMF applications can also be monitored:

Applications accessing an Enscribe file	NonStop Guardian
	TMF /Enscribe

Available Sensor and Agent Types

TransactionVision provides the following types of Sensors and Agents:

➤ WebSphere MQ (WMQ) Sensors

➤ Java Agent

  ➤ Servlet Sensor

  ➤ JMS Sensor

  ➤ EJB Sensor

  ➤ JDBC Sensor

➤ CICS Sensor

➤ .NET Agent

➤ BEA Tuxedo Sensor

➤ NonStop TMF Sensor

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 Agent

➤ 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.

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 15, "Installing and Configuring the Java Agent" for details.

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.

.NET Agent

The .NET Agent tracks Webservices in the ASP.NET environment. The .NET Agent tracks these Webservice method invocations by instrumenting the .NET code to collect events at the entry and exit of Webservice methods on the server and outgoing webservice calls on the client.

The .NET Agent also tracks HTTP calls, such as HTTP_POST, HTTP_GET, HTTP_PUT in the ASP.NET environment and ADO and remoting events.

BEA Tuxedo Sensor

The BEA Tuxedo Sensor monitors applications making Tuxedo ATMI calls in C and C++ environments. It intercepts and collects ATMI methods, the minimum set includes tpenqueue, tpdequeue, and tpcall.

For the collected method, two collection modes are supported: API + technology data, API + technology data + payload data.

This Sensor also supports data collection filtering on criteria common across all technologies, plus Tuxedo ATMI specific criteria including (but not limited to) Tuxedo queue space, queue name, and Tuxedo service name.

NonStop TMF Sensor

The NonStop TMF Sensor tracks audited Enscribe file system access. All audited transactions on the HP NonStop are logged in TMF audittrails. Because TMF protects any audited files on the NonStop system, it acts as a repository of all changes (adds, deletes, modifies) to data on the system as a whole.

This Sensor reads the TMF audittrails and tracks all access to Enscribe files that match the filter conditions configured by a user.

15

Installing and Configuring the 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 on page 143
➤ Installing and Configuring the Java Agent on Windows on page 145
➤ Installing and Configuring the Java Agent on UNIX on page 157
➤ Silent Installation of the Java Agent on page 166
➤ Running the JRE Instrumenter on page 167
➤ Configuring the Application Servers on page 177
➤ Configuring Messaging System Providers on page 178
➤ Configuring Custom User Events on page 180

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 Java 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 145 and "Installing and Configuring the Java Agent on UNIX" on page 157.

 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 147 and "Configuring the Java Agent to Work as a TransactionVision Java Sensor on UNIX" on page 159.

 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 167) and configure the application server (see "Configuring the Application Servers" on page 177).

Installation Files

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

Platform	Files
Windows	JavaAgentSetup_win_8_00.exe
AIX	JavaAgentSetup_ibm_8_00.bin
Linux	JavaAgentSetup_linux_8_00.bin
Solaris	JavaAgentSetup_sol_8_00.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 HP- BTO Software.

2 Enter TransactionVision in the Keyword field and "Trial Software" in the "Refine Search by Resource Type" field, and click Search.

3 Continue with "Running the Installation on Windows" on page 146.

To launch the installer from Business Availability Center:

4 Select **Admin > Platform** from the top menu in Business Availability Center, and click the **Setup and Maintenance** tab.

5 On the Downloads page, click the appropriate link to download the Java Agent installer for Windows.

6 Continue with "Running the Installation on Windows" on page 146.

To launch the installer from Business Availability Center 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 Business Availability Center product installation DVD:

3 From the HP Business Availability Center Installation DVD, select and run the executable file for your platform. See "Installation Files" on page 144.

4 Continue with "Running the Installation on Windows" on page 146.

Running the Installation on Windows

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

Note: Close all Windows programs currently running on your computer, including automatic backup programs. Antivirus, antispyware, and threat protection programs do not need to be shut down.

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.

Directory names must contain English characters only.

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.
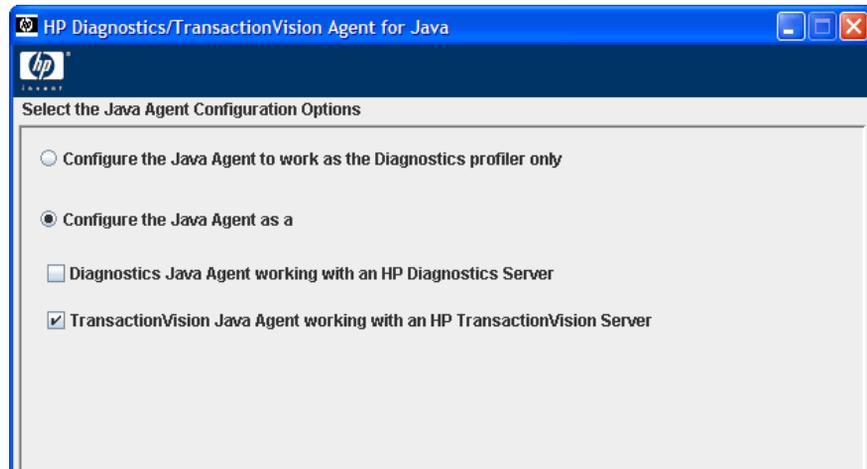
You can also choose from the following options:

➤ **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.
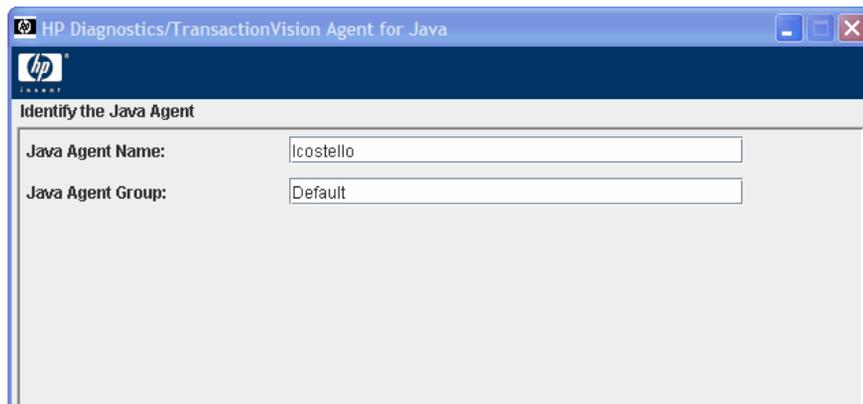
➤ For the Java Agent name, enter a name that uniquely identifies the agent within TransactionVision. The following characters can be used in the name: - , _ , and all alphanumeric characters. The agent name is assigned to be the Java Sensor name.

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 Select the application server to be monitored and its installation directory. Choose the event transport provider and specify the credentials.

➤ You can right-click inside the text box to open a file selection dialog.

➤ To use the SonicMQ event transport provider that is included with TransactionVision, specify the host name on which the Analyzer is running.

➤ Change the name of the configuration queue if the TransactionVisionanalyzer uses a different queue.

➤ Enter the user name and password for the event transport provider if they are required.

➤ Click **Next** to continue.

4 Configure the JMS and JDBC implementation 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,
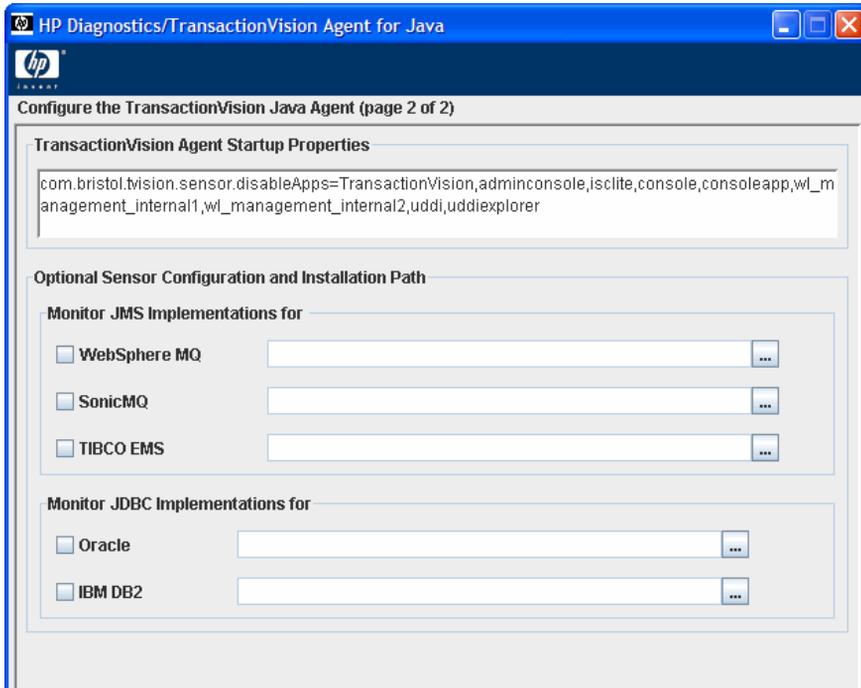
The Transaction Agent Startup Properties field contains directives as to the operation and instrumentation of the Agent. Typically you should accept the default unless instructed otherwise by HP Customer Support. The properties are separated by semicolons.

Note: 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. The dialog that appears next depends on the messaging middleware you are using:

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

Note: 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.
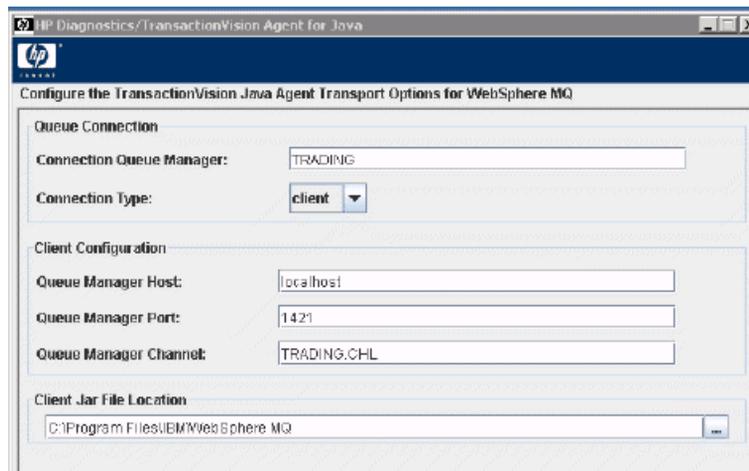
➤ Enter the configuration queue manager name.

➤ 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.
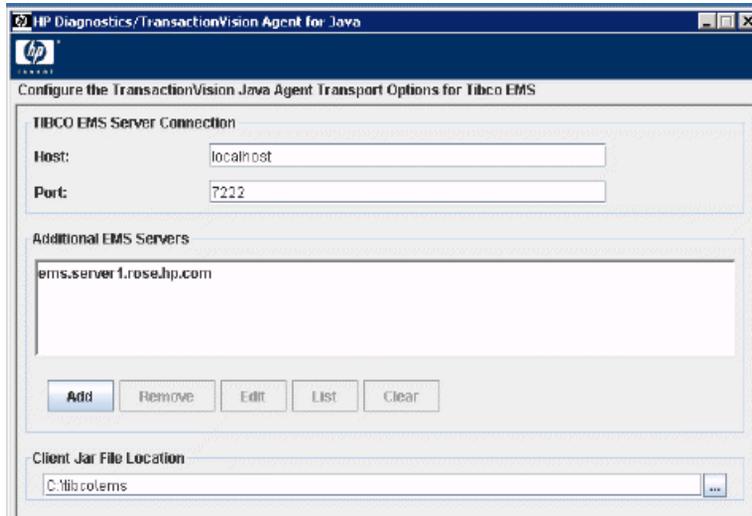
➤ Enter the host name. You can change the default port if you want.

➤ Enter or browse for the TIBCO EMS installation location.

➤ You can add and define EMS servers. Click **Add** to open the add dialog.
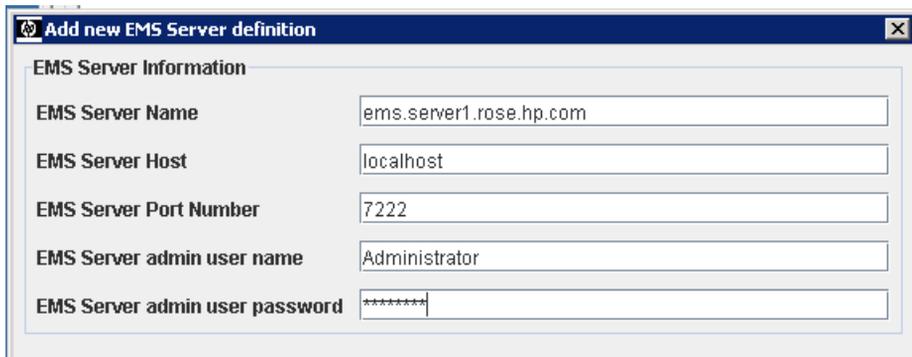
Enter the server definitions and click **OK**. To change the definitions, select the server and click **Edit**. To see what the definitions are for a selected server, click **List**.

➤ If you choose SonicMQ as your communication link transport, configure the transport settings for SonicMQ.
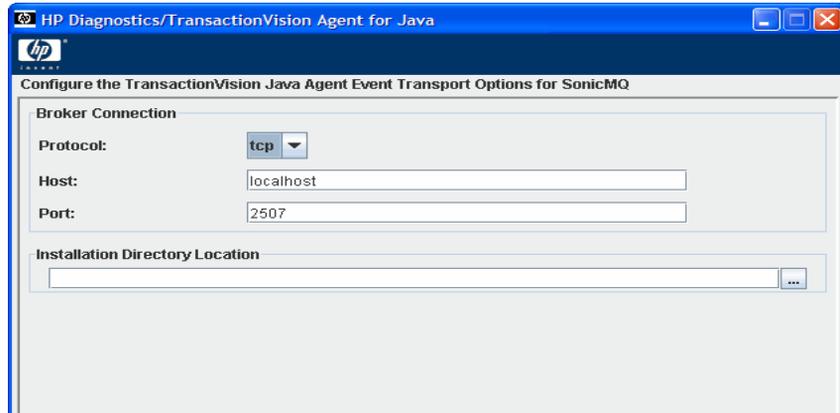
➤ Use the default tcp protocol unless a different protocol is used.

➤ Enter the host name. You can change the default port if you want.

➤ 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.
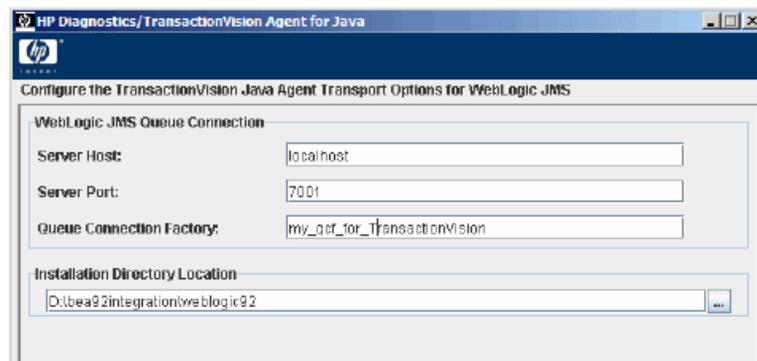
➤ Enter the host name. You can change the default port if you want.

➤ 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 executes a series of tests to validate and test the configuration settings:
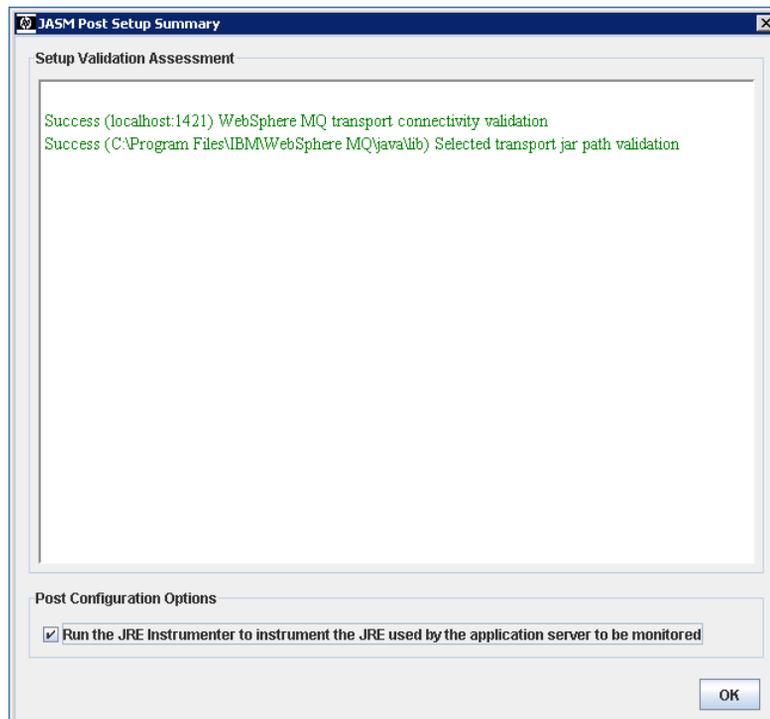
If any validation fails, check your transport settings and make sure that your JMS server or queue manager is running with proper settings.

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 167.

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 177.

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 167 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 177.

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 167 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>_8_00.bin** to the machine where the TransactionVision Server is to be installed.

2 Continue with "Running the Installation on UNIX" on page 158.

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

3 Select **Admin** > **Platform** from the top menu in Business Availability Center, and click the **Setup Maintenance** tab.

4 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>_8_00.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 145.

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 147.

Configuring the Java Agent on UNIX in Console Mode

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

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

➤ 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 step 2.

Press **Enter** to continue.

4 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_A rose.hp.com]:systemArose.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.

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

```
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/AppServer]:/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.

6 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,isclite,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
**************************************************************************
```

Note: The Transaction Agent Startup Properties field contains directives as to the operation and instrumentation of the Agent. Typically you should accept the default unless instructed otherwise by HP Customer Support. The properties are separated by semicolons.

> ➤ Press **Enter** at the Additional TransactionVision Properties prompt.

> ➤ Select one or more JMS transports by entering capital X for yes or capital O for no next to each transport (if you want, 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 by entering capital X for yes or capital O for no next to each

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

7 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 WebSpher
e 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 EM
S
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/bea/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.

8 When prompted to save your changes to the Java Agent Setup Module, enter **Y**.

9 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 VMs]:
IBM 1.5.0 (/usr/java5/jre)
IBM 1.4.2 (/usr/java14/jre)
IBMJ9 1.4.2 (/usr/java14/jre/lib/jclSC14)
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 174.

10 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.

11 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 177.

12 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 <JASMResponseFileName>

➤ On UNIX:

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

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
<JASMResponseFileName>

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.

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 169

➤ "Running the JRE Instrumenter on a UNIX Machine" on page 174

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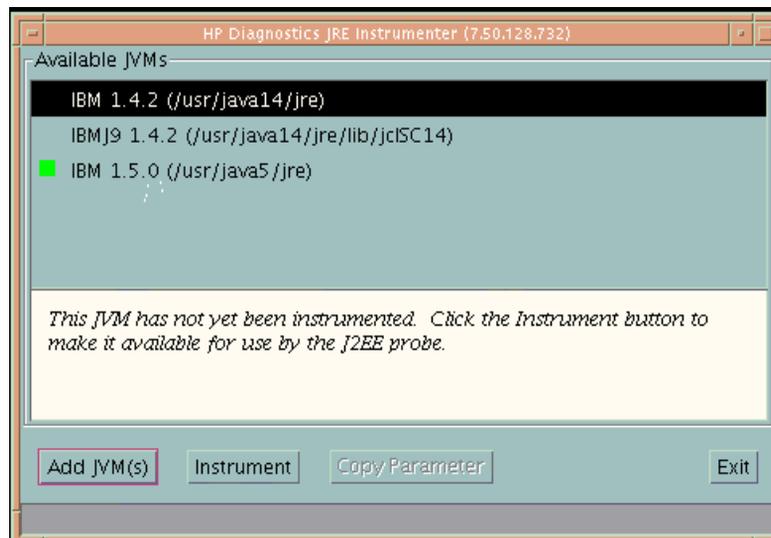
The Instrumenter lists the JVMs that were discovered by the Instrumenter and are available for instrumentation. The JVMs that have already been instrumented are listed with a green square preceding the name of the JVM.

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 170.

➤ 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 173.

➤ 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 173.

➤ 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.
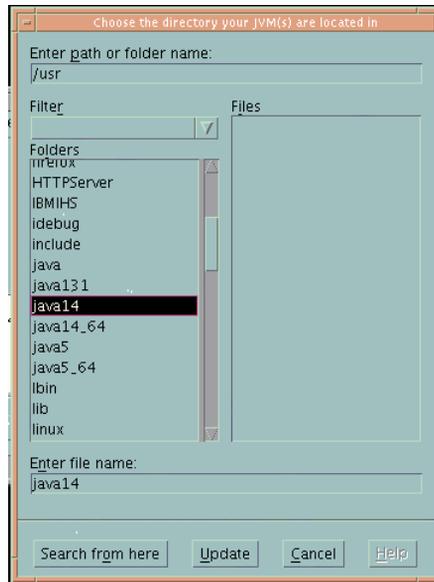
2 Enter the directory location where you would like the Instrumenter to begin searching for JVMs.

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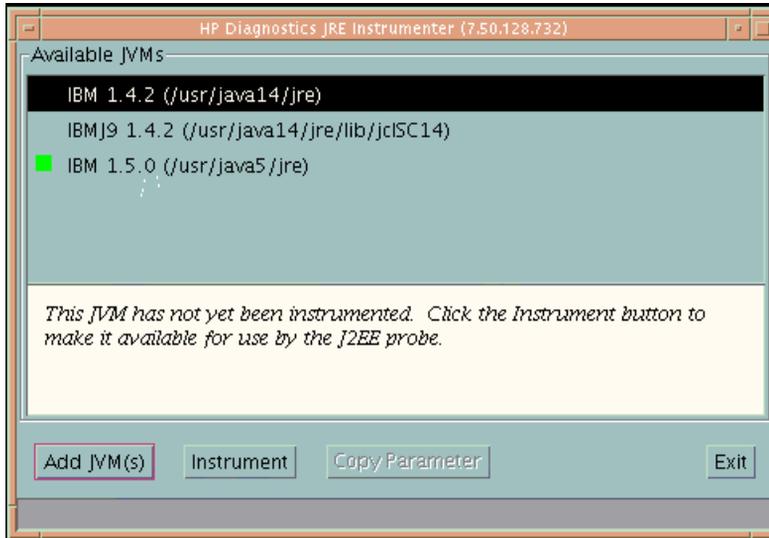
When the Instrumenter has completed the search, it enables the command buttons on the dialog. If the selected row is a JVM that has already been instrumented, the Instrument button is disabled. The green square indicates that the JVM has been instrumented.

Instrumenting a Selected JVM

Select a JVM that has not been instrumented from the **Available JVMs** list and click **Instrument**.
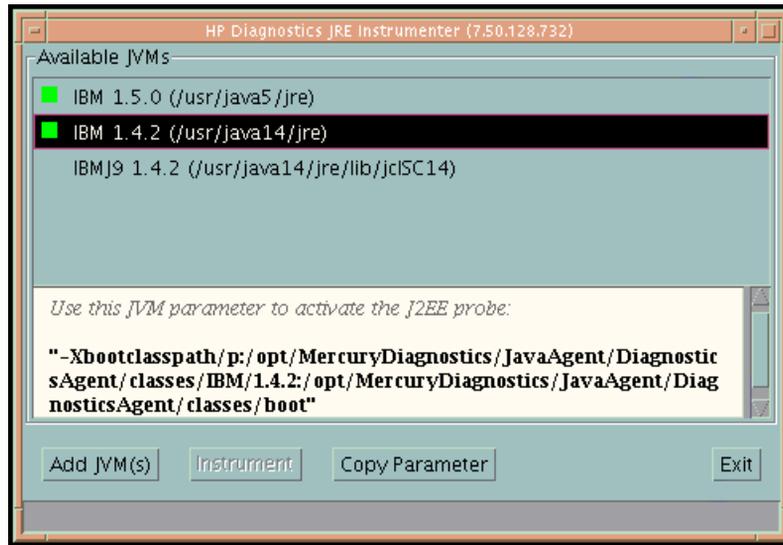
The JRE Instrumenter displays the JVM parameter in the box below the Available JVMs list, which must be used when the application server is started.

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.

Note: 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 169.

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
jreinstrumenter -1	Display the list of known JVMs. See "Displaying the List of Instrumented JVMs" on page 175 for details.
jreinstrumenter -a DIR	Look for JVMs below the DIR directory. See "Adding JVMs to the Available JVMs List" on page 175 for details.

Instrumenter Function	Description
jreinstrumenter -i JVM_DIR	Instrument the JVM in JVM_DIR. See "Instrumenting a Listed JVM" on page 176 for details.
jreinstrumenter -b JVM_DIR	Instrument the JVM in JVM_DIR and put the ClassLoader in **<java_agent_install_dir>/ DiagnosticsAgent/classes/boot**. See "Instrumenting a Listed JVM" on page 176 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.

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:
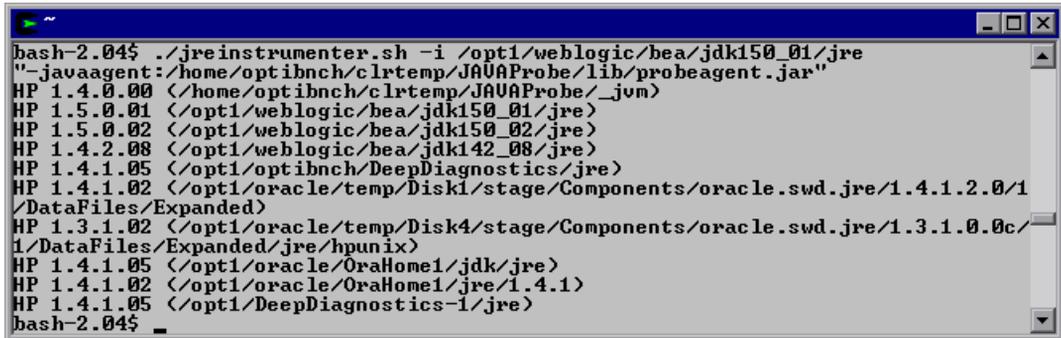
```
bash-2.04$ ./jreinstrumenter.sh -i /opt1/weblogic/bea/jdk150_01/jre
"-javaagent:/home/optibnch/clrtemp/JAVAProbe/lib/probeagent.jar"
HP 1.4.0.00 (/home/optibnch/clrtemp/JAVAProbe/_jvm)
HP 1.5.0.01 (/opt1/weblogic/bea/jdk150_01/jre)
HP 1.5.0.02 (/opt1/weblogic/bea/jdk150_02/jre)
HP 1.4.2.08 (/opt1/weblogic/bea/jdk142_08/jre)
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

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.

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 "Managing Communication Links" in *Using TransactionVision*.

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…**.

Configuring Custom User Events

You can include custom methods as part of the Transaction path to be presented as events. These methods are not by default included in the Event History unless they are part of the standard Java Application framework such as JMS, EJB, Servlets, JSP, etc.

Modify the **auto_detect.points** file and either create a TransactionVision only point or modify an existing Diagnostics point to specify a TransactionVision user event by specifying the tv:user_event tag on the details line.

For example:

```
#
# GENERIC EVENT
#
[GenericEvent]
class    = !com.company.importantclass
method   = !.*
signature = !.*
detail    = tv:user_event
```

For more information about creating and modifying existing instrumentation points, see custom instrumentation in the *HP Diagnostics Installation and Configuration Guide*.

16

Installing WebSphere MQ and User Event Sensors on Windows

This chapter includes:

Starting the Installation Program 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 15, "Installing and Configuring the 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, including automatic backup programs. Antivirus, antispyware, and threat protection programs do not need to be shut down.

2 In the Windows Explorer, double-click **tvwmque_800_win.exe**. The InstallShield Welcome screen appears.
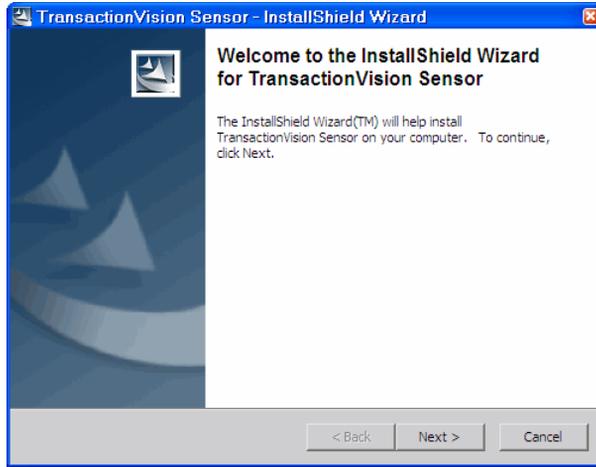
3 Click **Next**> to display the InstallShield Save Files screen.

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 **tvwmque_800_win.exe** and click
Next> on the InstallShield Welcome screen to display the Sensor setup
maintenance menu:
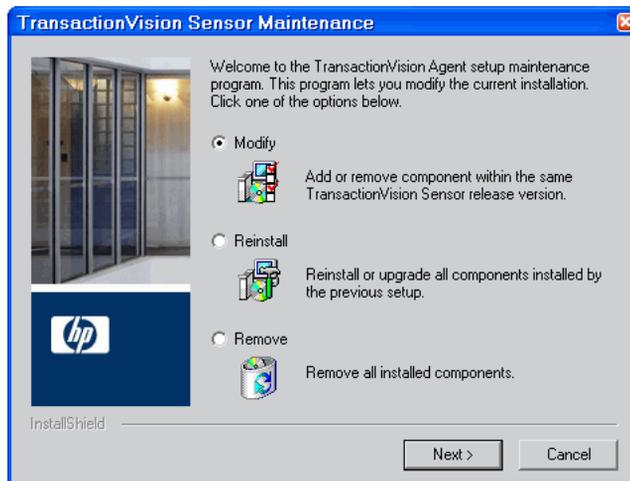
 1 Select one of the following to upgrade the TransactionVision installation (to
modify the installation, see "Modifying the Installation" on page 185):

 ➤ If you want 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:

Configuration File Migration

Installation has detected a previously installed version of TransactionVision.

Do you want to migrate the configuration files from the previous installation?

Clicking "No" will overwrite those files, causing any setup information to be lost.

[Yes]   [No]

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.

Migrate?

Do you want to backup the configuration files from the previous installation?

Clicking "No" will overwrite those files, causing any setup information to be lost.

[Yes]   [No]

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 want 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 **tvwmque_800.exe** and click **Next>** on the InstallShield Welcome screen to display the TransactionVision Sensor Maintenance screen:
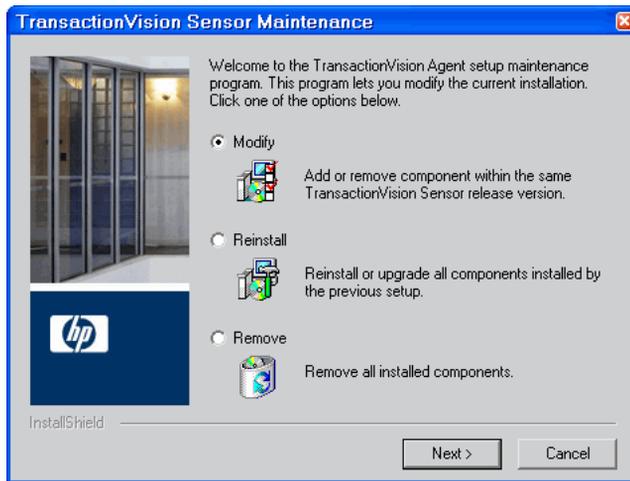
2 To install an additional Sensor or remove an installed Sensor, select **Modify** and click **Next**> to display the Select Components screen.
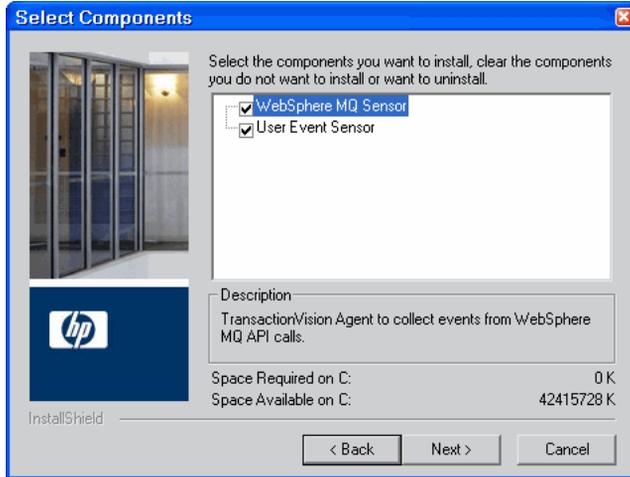
3 Select the Sensors you want to install and click **Next**> to run the modified installation.

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 want 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.

17

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 15, "Installing and Configuring the Java Agent".

Note that TransactionVision provides unique packages for each Sensor.

This chapter includes:

➤ Installing Sensors on page 189

➤ Uninstalling Sensors on page 192

Installing Sensors

Installation Files

The following table shows the installation file names for the WebSphere MQ Sensor.

Platform	Files
AIX	tvwmq_800_aix_power.tar.gz
HP-UX	tvwmq_800_hpux_parisc.tar.gz tvwmq_800_hpux_ia64.tar.gz
Linux	tvwmq_800_linux_x86.tar.gz

Platform	Files
Solaris	tvwmq_800_sol_sparc.tar.gz
IBM i5/OS	tvmq_800_i5os_iseries.savf

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

Platform	Files
AIX	tvue_800_aix_power.tar.gz
Linux	tvue_800_linux_x86.tar.gz
Solaris	tvue_800_sol_sparc.tar.gz

Installation Steps

1 Change to the directory location of the TransactionVision installation files (either a DVD device or download directory).

Note: On Solaris and HP-UX, you must copy the installation files from the DVD 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:

./tvinstall_800_unix.sh

The following menu is displayed:

> 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:

Note: 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 want 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 sensored 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:

   ./tvinstall_800_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 want 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.

18

Installing Sensors on i5/OS

This chapter includes:

➤ Starting the Installation Program on i5/OS on page 195

Starting the Installation Program 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 tvwmq_800_i5os_iseries.savf from the CD_ROM and rename it to sensor800.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 tvwmq_800_i5os_iseries.savf sensor800.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 sensor800.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.

19

Installing and Configuring Sensors on z/OS

This chapter includes:

About Sensors in the z/OS Environment

There are two TransactionVision Sensor types for deployment in the IBM z/OS environment, each of which has its own three character product code:

➤ SLD Sensor components are designed to monitor: CICS, WebSphere MQ (WMQ) Batch, and WMQ IMS events.

➤ SLM Sensor components are designed to monitor: WMQ CICS Bridge and WMQ IMS Bridge events.

This chapter provides instructions to perform the base component installation and post-install configuration tasks for each Sensor type. The base installs are virtually identical and therefore described once. Sensor component configuration tasks differ significantly and are therefore handled separately.

For additional z/OS ESM (External System Requirements), see Appendix D, "Additional z/OS Settings."

Base Component Installation Summary

1 Transfer files from the installation media to the zSeries platform. Two options are available:

  ➤ Use the **tvinstall_800_zos_zseries.bat** script found on the installation media to assist with the transfer (recommended).

  ➤ Manually FTP the install files.

2 TSO RECEIVE the transmitted files to create product datasets. Two options are available:

  ➤ Use the (TSORECV) Rexx script created by the **tvinstall_800_zos_zseries.bat** script in Step 1 above to execute the TSO RECEIVE commands (recommended).

  ➤ Manually execute the needed TSO RECEIVE commands.

3 Determine the best installation approach for your environment. Two options are available:

  ➤ SMP/E (recommended). You can either create a new SMP/E global zone (recommended) or use an existing SMP/E global zone.

  ➤ Non-SMP/E

4 Tailor and execute the installation jobs; the degree of customization needed will vary depending on the installation approach (SMP/E or Non-SMP/E).

5 Review installation libraries.

Base Component Installation Procedure

Step 1: Transfer the Files from the Installation Media to the zSeries Host

Two options are available:

➤ Run the **tvinstall_800_zos_zseries.bat** script.

➤ Manually FTP the install files.

File Name Conventions

File names on the installation media adhere to the following pattern:

tv<prodcode><filequal>_<ver>_zos_zseries.xmit

The ftp output dataset file names adhere to the following pattern:

<&custhlq>.<prodcode>|<ver>.<filequal>

where:

<filequal> = f1 | f2 | f3 | mcs

<prodcode> = SLD | SLM

<ver> = 800

<&custhlq> = a customer-provided high level dataset qualifier for output of the FTP PUT sub-command on the target z/OS system.

For example, a file name on the installation media is:

tvsldrf1_800_zos_zseries.xmit

An ftp output dataset name might be:

TVISION.SLD800.F1

Option 1: Run the tvisntall_800_zos_zseries.bat script

1 Logon to a workstation running Microsoft Windows that is capable of connecting to the target z/OS system using TCP/IP FTP.

2 Start an MS-Windows Command Prompt session (**Start** > **All Programs** > **Accessories** > **Command Prompt**). Adjust the windows properties, increasing its size and buffering, as necessary.

199

3 Navigate, using the change directory (cd) command, to the folder containing installation media.

4 Review the information that will be collected by the script by entering:

tvinstall_800_zos_zseries.bat   /h | more

5 Invoke **tvinstall_800_zos_zseries.bat** with no parameters and respond to the prompts.

Note: If you are not satisfied with the values you enter, you can quit tvinstall_800_zos_zseries.bat without initiating a file transfer. In addition, you can terminate the script at any time by entering **CTRL+C** repeatedly.

---

6 Carefully review the results of the FTP command output issued by **tvinstall_800_zos_zseries.bat**.

➤ There should be no errors.

➤ Check for the existence of installation RELFILES with the high-level qualifiers specified on the target z/OS system.

➤ Check for the existence of the Rexx script (TSORECV) in the PDS library specified.

➤ If any datasets are missing or FTP errors were detected, they should be investigated, corrected, and the tvinstall_800_zos_zseries.bat process repeated.

Option 2: Manually FTP the install files:

Don't perform this step if you installed the files by using the **tvisntall_800_zos_zseries.bat script**.

1 Logon to a workstation running Microsoft Windows that is capable of connecting to the target z/OS system using TCP/IP FTP.

2 Start an MS-Windows Command Prompt session (**Start** > **All Programs** > **Accessories** > **Command Prompt**). Adjust the windows properties, increasing its size and buffering, as necessary.

3 Navigate, using the change directory (cd) command, to the folder containing installation media.

4 Using a z/OS account with permissions sufficient to create datasets with the desired high level qualifier, initiate an FTP session with the target z/OS system, for instance:

> > ftp hostname

5 Issue the following FTP sub-commands:

> **a** ftp> **quote site fixrecfm 80 lrecl=80 recfm=fb blksize=3120 vol=&custvol u=&custunit pri=30 sec=5 tr**
>
> where **&custvol** is a valid disk volser and **&custunit** is a valid disk device type or esoteric
>
> **b** ftp> **bin**
>
> **c** ftp> **put tv**<prodcode><filequal>_<ver>_zos_zseries.xmit
> '<&hlq>.<prodcode><|ver>.<filequal>'
>
> where
> <prodcode> = slm | sld
> <&hlq> = customer select high level qualifier
> <filequal> = f1 | f2 | f3 | mcs
> <ver> = 800
>
> For example:
>
> put tvsldf1_800_zos_zseries.xmit 'TVISION.SLD800.F1'
>
> **d** Issue ftp **put** commands for any remaining product installation files with a filename suffix of **fn** or **mcs**.
>
> **e** It is important that ftp command output be carefully examined for errors. If detected, the target dataset on the z/OS system may need to be deleted and the ftp put command re-executed.
>
> **f** ftp> **quit**

Step 2: TSO RECEIVE the Transmitted Files to Create Product Datasets

Two options are available:

➤ Run the (TSORECV) Rexx script created by the tvinstall_800_zos_zseries.bat in step "Step 1: Transfer the Files from the Installation Media to the zSeries Host" on page 199.

➤ Manually execute the needed TSO RECEIVE commands

Option 1: Execute the (TSORECV) Rexx script

From a TSO command line or TSO/ISPF command line, execute the (TSORECV) Rexx script created in "Step 1: Transfer the Files from the Installation Media to the zSeries Host" on page 199. This results in the execution of TSO RECEIVE commands for all z/OS target datasets specified.

The RECEIVEd datasets will be allocated using the same high level qualifier, however the character 'A' will be added as a prefix to the prodcode. For example, if input pattern=TVISION.SLD800.filequal, then the output pattern would be TVISION.ASLD800.filequal

Option 2: Manually execute the needed TSO RECEIVE commands

From a TSO command line or TSO/ISPF command line, prefixing with 'TSO' as necessary, execute RECEIVE commands as shown in the example table below. In response to prompt 'INMR906A Enter restore parameters or 'DELETE' or 'END'" enter '**DSN(&custhlq.A**<prodcode>**800.filequal)**'.

Command	Filename
RECEIVE INDSNAME('&hlq.SL_800.F1')	DSN('&hlq.ASL_800.F1')
RECEIVE INDSNAME('&hlq.SL_800.F2')	DSN('&hlq.ASL_800.F2')
RECEIVE INDSNAME('&hlq.SL_800.F3')	DSN('&hlq.ASL_800.F3')
RECEIVE INDSNAME('&hlq.SL_800.MCS')	DSN('&hlq.ASL_800.SMPMCS')

Step 3: Determine the Best Installation Approach

Two options are available:

➤ SMP/E (recommended). You can either create new SMP/E global zone (recommended) or use an existing SMP/E global zone.

➤ Non-SMP/E

Step 4: Tailor and Execute the Installation Jobs

Two options are available: SMP/E or non-SMP/E.

Option 1: SMP/E installation jobs

Tailor and execute the jobs in the order presented. Jobs are in **dataset &custhlq.A**<prodcode>**800.F3.** After substituting the 3rd character of the product code for the '_' underscore character in the list below; proceed to tailor the individual members to meet your site requirements. Read the comments at the beginning of each job and customize accordingly.

Following execution, carefully review all output making sure all steps completed successfully before moving on to the next job. If installing into an existing SMP/E global zone, carefully review SMP/E steps, parameters, and inputs.

SMP/E installation jobs:

Order	Job	Description
1.	**SL_ALLOC**	Allocate target and distribution libraries
2.	**SL_GZON**	Defines SMP/E global zone (not needed if using existing global zone)
3.	**SL_DZON**	Defines SMP/E distribution zone
4.	**SL_TZON**	Defines SMP/E target zone
5.	**SL_DDDEF**	Defines SMP/E DDDEFs
6.	**SL_RECV**	SMP/E Receive
7.	**SL_APPLY**	SMP/E Apply
8.	**SL_IVP**	Installation Verification
9.	**SL_ACCPT**	SMP/E Accept **Important!** Run this job only after the installation has been configured and fully tested, since it is not possible to remove the product from target or distribution libraries once the SMP/E ACCEPT function has been executed.

Option 2: Non-SMP/E installation jobs

Tailor and execute the following jobs in the order presented to perform a non-SMP/E installation. Jobs may be found in dataset **&custhlq.A**<prodcode>**800.F3**. Read the comments at the beginning of each job and customize accordingly. Following execution, carefully review all output making sure all steps completed successfully.

Non-SMP/E installation jobs:

Order	Job	Description
1.	**SL_INSTL**	
2.	**SL_IVP**	Installation verification

Step 5: Review the Installation Libraries

DS#	Dataset Name	Description
1	TVISION.SSL_AUTH	Sensor Load Modules (APF Auth)
2	TVISION.SSL_INST	Sensor Installation JCL Samples
3	TVISION.SSL_LOAD	Sensor Load Modules
4	TVISION.SSL_PROC	Sensor Sample JCL
5	TVISION.SL_800.F1	SMP/E JCLIN
6	TVISION.SL_800.F2	Inst RELFILE - Load Modules
7	TVISION.SL_800.F3	Inst RELFILE - Installation JCL
8	TVISION.SL_800.SMPMCS	Inst RELFILE - SMP/E MCS stmts

➤ Not shown in the above table are the installation SMP/E distribution libraries (DLIBS). These will not be populated until execution of SMP/E Accept processing.

➤ 'TVISION' is the default high level qualifier, yours may differ.

Step 6: Repeat Base Component Installation Procedure (Steps 3-5 as Needed)

Depending upon the mainframe sensor components licensed at your installation, it may be necessary to repeat steps 3-5 of the base component installation procedure using another product code (SLM or SLD).

Configuring the SLD Sensor Components on z/OS: CICS, WebSphere MQ Batch, and WebSphere MQ IMS

Also see Chapter 20, "Configuring the CICS, WMQ Batch, and WMQ-IMS Bridge Sensor" for additional information pertaining to Sensor operation, behavior, and configuration. Reading through this material will enhance your understanding of TransactionVision Sensor operation and interaction with other system components.

To configure the SLD Sensor Components on z/OS: CICS, WebSphere MQ Batch, and WebSphere MQ IMS, follow these steps:

1 Update your CICS CSD file with the required resource definitions for the SLD Sensor by customizing and running job SLDCICSD, found in DS#2 - Sensor Installation JCL Samples. 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.

2 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. If you run multiple CICS regions with separate startup JCL, repeat this step for each CICS region to be monitored by the Sensor. The CICS modules are:

> SLDPCCX
>
> SLDPCMX
>
> SLDPCPX
>
> SLDPCSX
>
> SLDPDSX

SLDPFCX

SLDPICX

SLDPPCX

SLDPPSX

SLDPTCX

SLDPTDX

SLDPTSX

3 Optional. To automatically enable the sensor's CICS exit programs at CICS startup time, define SLDPCSX as a second pass PLTPI. Refer to CICS Resource Definition Guide, DFHPLT section. Add the definition to your existing DFHPLTxx. If a new PLT is defined, add references to it 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. (Note - if this optional step is not performed, CICS transaction SLDS can be executed to enable Sensors.)

Sample SIT entry:

```
...
PLTPI=BI,
...
```

Sample definition of PLT with the suffix entry:

```
//DFHPLTPI    EXEC DFHAUPLE
//ASSEM.SYSUT1 DD *
  DFHPLT TYPE=INITIAL,SUFFIX=BI
  DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
  DFHPLT TYPE=ENTRY,PROGRAM=SLDPCSX
  DFHPLT TYPE=FINAL
  END
//*
```

4 APF-authorize the TransactionVision library, &hlq.SSLDAUTH. Please refer to MVS Initialization and Tuning Reference, PROGxx or IEAAPFxx system parameters for additional details. (Note - library &hlq.SSLDLOAD should NOT be APF authorized.)

5 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 job SLDCRTQS in DS#2 - Sensor Installation JCL.

6 Customize the sample Sensor startup procedures, TVISION and TVISIONC, found in DS#4 - Sample Sensor JCL, and copy them to an appropriate procedure library for your site. See Chapter 20, "Configuring the CICS, WMQ Batch, and WMQ-IMS Bridge Sensor" for a detailed discussion on startup procedure keyword parameters and their meaning.

Configuring the SLM Sensor Components on z/OS: WebSphere MQ CICS Bridge, and WebSphere MQ IMS Bridge

Chapter 20, "Configuring the CICS, WMQ Batch, and WMQ-IMS Bridge Sensor" contains additional information pertaining to Sensor operation and behavior. Reading through this material will enhance your understanding of TransactionVision Sensor operation and interaction with other system components.

➤ Some of the following 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.

➤ Some of the following configuration steps only apply to the WebSphere MQ Sensor for IMS; they are noted as "IMS only." If you do not intend to use the WebSphere MQ Sensor for IMS, omit those steps.

➤ Steps not noted as "CICS only" or "IMS only" are common to both Sensor types and should be executed independent of CICS or IMS considerations.

Also see Chapter 24, "Configuring WebSphere MQ Sensors"for additional information regarding Sensor setup and operation.

1 (CICS only) Please read 'Background Information: WebSphere MQ Sensor for CICS' found at the end of this configuration section.

2 (CICS only) Update your CICS CSD file with the required resource definitions for the SLM Sensor by customizing and running job SLMCICSD, found in DS#2 - Sensor Installation JCL Samples.   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.

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.

3 (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.

4 (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:

  a CKQC MODIFY N E to enable the crossing exit (E for enable) or

  b CKQC MODIFY N D to disable the crossing exit (D for disable)

5 APF-authorize the TransactionVision library, &hlq.SSLMAUTH. Please refer to MVS Initialization and Tuning Reference, PROGxx or IEAAPFxx system parameters for additional details. (Note - library &hlq.SSLMLOAD should NOT be APF authorized.)

6 Create appropriate Sensor configuration and event queues on the WebSphere MQ queue manager to be used for communication between the Sensor Driver component and the TransactionVision Analyzer. Customize and run job SLMCRTQS in DS#4 - Sample Sensor JCL.

Note: This step may not be needed if JOB SLDCATQS was previously executed.

---

7 (CICS only) Configure SLMC for CICS. This step is not required, but strongly recommended as it can significantly improve the performance of WMQ application programs run in your CICS region when the TransactionVision Analyzer component is not monitoring the region. Please refer to section titled 'Configure SLMC for CICS' at the end of this chapter for details.

8 (IMS only) Please read 'Background Information: WebSphere MQ IMS Bridge Sensor' found at the end of this configuration section.

9 (IMS only) 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.

Background Information: 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.

Configure SLMC for CICS

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

Background Information: WebSphere MQ IMS Bridge Sensor

The WebSphere MQ IMS Bridge Sensor is implemented as three components which communicate with one another via cross memory program calls:

➤ An OTMA Input/Output Edit exit routine, DFSYIOE0, which runs in the IMS control region.

➤ A control function, referred to as TVISIONB, which runs as a started task in a separate address space.

➤ An event dispatcher function, referred to as TVISIOND, which runs as a started task in another separate address space.

For instructions on completing Sensor setup and operating the WebSphere MQ IMS Bridge Sensor, see Chapter 24, "Configuring WebSphere MQ Sensors."

20

Configuring the CICS, WMQ Batch, and WMQ-IMS Bridge Sensor

Once the TransactionVision CICS Sensors are installed, you must configure and then start the Sensors.

This chapter includes:

➤ Overview on page 213

➤ Common Sensor Components on page 214

➤ CICS Sensor Commands on page 215

➤ Sensor Operations on page 222

➤ Buffer Queue Considerations on page 224

➤ TransactionVision Manager Startup Procedure on page 225

➤ Sensor Driver Startup Procedure on page 226

Overview

All TransactionVision z/OS Sensors are implemented as a combination of a common base component—the TransactionVision Manager, an environment/subsystem specific Sensor manager component and the Sensor itself— WMQ Batch Sensor and the WMQ-IMS Sensor.

TransactionVision Manager (TVM)

TVM is an authorized program that runs as a started task. It controls the other components and provides common routines and structures used by all Sensors. After TVM is started, you issue MVS modify commands to the TVM started task job name to request it to start, stop, query, or control Sensors. The available commands are listed in "CICS Sensor Commands" on page 215. section. You issue the MVS stop command to shutdown TVM, in which case all Sensors under its control are also shutdown. See "TransactionVision Manager Startup Procedure" on page 225 for more information.

Common Sensor Components

Sensor Manager

There are two Sensor managers, one for WebSphere MQ Sensors and one for the CICS Sensor, but both conform to a generic model and perform common functions required by all Sensors. The Sensor manager is started as a result of a start Sensor command to TVM and runs as a subtask of TVM. Startup parameters identify the Sensor type and the target application set to be monitored. The Sensor manager type is MQBATCH, WMQ-IMS, or CICS.

For WMQ-IMS and CICS Sensor types, you must also specify the IMSID or CICS SYSID of the IMS or CICS system to be monitored. You must start a separate Sensor manager for each WMQ-IMS or CICS system to be monitored.

The Sensor manager creates a buffer queue data space to store WebSphere MQ or CICS events and starts the Sensor driver component. A single Sensor handles the events generated by all the applications in all the dependent regions for a WMQ-IMS system.

Sensor Driver

There are two Sensor drivers, one for WebSphere MQ Sensors and one for the CICS Sensor, but both conform to a generic model and perform common functions required by all Sensors. The Sensor driver is an unauthorized program that runs as a started task in its own address space.

The Sensor driver is automatically started by the Sensor manager when it starts up and terminated when it shuts down. A parameter on the start Sensor command specifies the procedure to be used to start the driver. See "Sensor Driver Startup Procedure" on page 226 for more information.

The Sensor driver communicates with the TransactionVision Analyzer via WebSphere MQ. In response to configuration messages from the Analyzer, the driver indicates to the Sensors which event types to capture and what filtering criteria to apply. It also retrieves the captured events from the buffer queue data space and, subject to the filtering criteria, sends them to the Analyzer.

Sensor Event Capture Components

Each Sensor has a unique event capture component that runs in the application environment to intercept transaction activity and store the associated event data into the buffer queue.

WebSphere MQ Sensors

Batch and IMS programs to be monitored must be linked to a TransactionVision stub program, which replaces the WebSphere MQ stub as follows:

WebSphere MQ Stub	Replacement Stub	Description
CSQBSTUB	SLDPSTBB	Batch
CSQBRRSI	SLDPSTBB	Batch RRS
CSQBRSTB	SLDPSTBB	Batch RRS
CSQQSTUB	SLDPSTBB	IMS

CICS Sensor Commands

The command to start and stop the TransactionVision Manager are standard MVS start and stop commands.

Start TransactionVision Manager (TVM)

```
START procname[.jobname],TVID=tvid
```

where

➤ **procname** is the name of a cataloged procedure to start TVM.

➤ **jobname** is the MVS jobname to be assigned to the started task. If not specified the jobname defaults to the procedure name.

➤ **tvid** is the unique system id of this instance of TVM, consisting of four or fewer characters. Note that **tvid** may be an optional parameter, depending on the procedure definition. See "TransactionVision Manager Startup Procedure" on page 225.

Example:

```
S TVISION, TVID=TV01
```

The TransactionVision Manager component is started. The following messages will be displayed:

```
SLDS434I TVISION Bristol TransactionVision for z/OS – V5.0.0
SLDS400I TVISION TransactionVision Manager startup in progress.
SLDS401I TVISION TV01 TransactionVision Manager startup complete.
```

Stop TransactionVision Manager (TVM)

```
STOP jobname
```

where

➤ **jobname** is the MVS jobname specified or defaulted on the start command for the TVM job to be stopped.

Example:

```
P TVISION
```

The TransactionVision Manager component is stopped. Any Sensors that have been started within the stopped TVM are also stopped. The following messages will be displayed:

```
SLDS402I TVISION TV01 STOP command received.
SLDS404I TVISION TV01 TransactionVision Manager termination in progress.
SLDS405I TVISION TV01 TransactionVision Manager termination complete.
```

If Sensors are running within the TVM, their shutdown messages will also be displayed.

Note: All other Sensor commands are issued as standard MVS modify commands. Most of the command and operand names can be abbreviated to as few characters as required to make the name unique. If the abbreviation is not unique, the alphabetically first command or operand name that fits is assumed. However, some names are reserved for future use.

Start Sensor

```
MODIFY tvm_jobname,START sensortype [SYSID(sysid) | IMSID(imsid)]
[DRVRPROC(drvrproc)]
[QMGR(qmgr)] [CONFIGQ(configq)]
[QBLKSIZE(qblksize)] [MAXQBLKS(maxqblks)]
```

where:

➤ **tvm_jobname** is the MVS jobname specified or defaulted on the start command for the TVM job that will control the started Sensor.

➤ **START** is the command name and is required.

➤ **sensortype** is the Sensor type and is required. Specify **MQBATCH**, **MQIMS**, or **CICS**.

If the **sensortype** is **MQBATCH**, do not specify either **IMSID** or **SYSID**.

➤ **sysid** is required when the sensortype is **CICS**. **sysid** must specify the SYSID of the CICS region to be monitored.

➤ **imsid** is required when the sensortype is **MQIMS**. **imsid** must specify the IMSID of the IMS system to be monitored.

➤ **drvrproc** is optional. drvrproc specifies the name of the cataloged procedure to start the Sensor driver. See "Sensor Driver Startup Procedure" on page 226 for more information. The default name is **TVISIONC**.

➤ **qmgr** is optional. **qmgr** specifies the name of the WebSphere MQ queue manager through which the Sensor driver component will communicate with the TransactionVision Analyzer. The default is specified as a parameter in the Sensor driver startup procedure. See "Sensor Driver Startup Procedure" on page 226 for more information on the relationship of this parameter with the Sensor driver startup procedure.

➤ **configq** is optional. **configq** specifies the name of the WebSphere MQ queue from which the Sensor driver component will receive configuration messages from the TransactionVision Analyzer. The default is specified as a parameter in the Sensor driver startup procedure. See "Sensor Driver Startup Procedure" on page 226 for more information on the relationship of this parameter with the Sensor driver startup procedure.

➤ **qblksize** is optional. **qblksize** specifies the size, in megabytes, of each block in the buffer queue data space. The minimum for **qblksize** is **1**; the maximum is **100**, and the default is **3**. See "Buffer Queue Considerations" on page 224 for more information.

➤ **maxqblks** is optional. **maxqblks** specifies the maximum number of buffer queue blocks that the Sensor is allowed to allocate in its data space. Each queue block is of the size specified or defaulted by the **qblksize** parameter. The minimum value for maxqblks is **3**; the maximum is **2046**, and the default is **5**. See "Buffer Queue Considerations" on page 224 for more information.

Example 1:

```
F TVISION,S MQB DR(TVISIONM) QM(CSQ1)
  CO(TVISION.CONFIGURATION.QUEUE)
```

The WebSphere QM Batch Sensor is started: a Sensor manager subtask of
TVM is started and a Sensor driver task is started in a separate address space.
The following messages will be displayed:

```
SLDS400I TVISION MQBATCH TransactionVision sensor startup in progress.
SLDS401I TVISION MQBATCH TransactionVision sensor startup complete.
IEF403I TVISIONM - STARTED […and other MVS messages issued when starting a
job]
+SLMS278I : TransactionVision Sensor: WMQ **** Sensor Driver startup completed.
QMGR=CSQ1, CONFIGQ=TVISION.CONFIGURATION.QUEUE.
```

Example 2:

```
F TVISION,S MQI IMSID(IVP1) DR(TVISIONM) QM(CSQ1)
  CO(TVISION.CONFIGURATION.QUEUE)
```

The WebSphere QM IMS Sensor is started: a Sensor manager subtask of TVM
is started and a Sensor driver task is started in a separate address space. The
following messages will be displayed:

```
SLDS400I TVISION MQIMS IVP1 TransactionVision sensor startup in progress.
SLDS401I TVISION MQIMS IVP1 TransactionVision sensor startup complete.

IEF403I TVISIONM - STARTED […and other MVS messages issued when starting a
job]

+SLMS278I : TransactionVision Sensor: WMQ IVP1 Sensor Driver startup completed.
QMGR=CSQ1, CONFIGQ=TVISION.CONFIGURATION.QUEUE.
```

When starting the CICS Sensor, the following TransactionVision error
SLDS499S might occur:

```
SLDS499S TVISION CICS xxxx TVISION system error. Diagnostic data follow:
SLDS499S 01 07050224 19472077 0000000C 00000034
SLDS499S 02 00E00080 00000000 0000000C 6C000611
```

If the last two values on the third line are as listed in this message, the error is due to an insufficient number of **SCOPE=COMMON** data spaces available in the current system. Increase the value of the **MAXCAD** parameter in the appropriate **IEASYSnn** member of SYS1.PARMLIB and then IPL the MVS image. For a detailed description of **MAXCAD** parameter tuning, please refer to the IBM *MVS Initialization and Tuning Reference* manual.

Stop Sensor

MODIFY *tvm_jobname*,STOP *sensortype* [SYSID(*sysid*) | IMSID(*imsid*)]

where:

➤ **tvm_jobname** is the MVS jobname specified or defaulted on the start command for the TVM job that controls the Sensor to be stopped.

➤ **STOP** is the command name and is required.

➤ **sensortype** is the Sensor type and is required. Specify **MQBATCH**, **MQIMS**, or **CICS**.

➤ If the **sensortype** is **MQBATCH**, do not specify either **SYSID** or **IMSID**.

➤ **sysid** is required if sensortype is **CICS**. **sysid** must specify the same SYSID that was specified on the start Sensor command for this Sensor.

➤ **imssid** is required if sensortype is **MQIMS**. **imsid** must specify the same IMSID that was specified on the start Sensor command for this Sensor.

Example 1:

```
F TVISION,STO MQB
```

The WebSphere MQ Batch Sensor manager and Sensor driver are stopped.
The following messages will be displayed:

```
SLDS404I TVISION MQBATCH TransactionVision sensor termination in progress.
SLDS443I TVISION MQBATCH sensor quiescing: 14 events in buffer queue.
…
SLDS445I TVISION MQBATCH Sensor quiesce completed.
SLDS448I TVISION MQBATCH Sensor statistics:
  Events in queue:           0
  Events collected         474
  Events dispatched        474
  Events_lost                0

+SLDS27BI : TransactionVision Sensor: MQBATCH Sensor Driver is ending
IEF404I TVISIONM - ENDED […and other MVS messages issued when stopping a job]
SLDS405I TVISION MQBATCH TransactionVision sensor termination complete.
```

Example 2:

```
F TVISION,STO MQI IMSID(IVP1)
```

The WebSphere MQ Batch Sensor manager and Sensor driver are stopped.
The following messages will be displayed:

```
SLDS404I TVISION MQIMS IVP1 TransactionVision sensor termination in progress.
SLDS443I TVISION MQIMS IVP1 sensor quiescing: 14 events in buffer queue.
…
SLDS445I TVISION MQIMS IVP1 Sensor quiesce completed.
SLDS448I TVISION MQIMS IVP1 Sensor statistics:
  Events in queue:           0
  Events collected         474
  Events dispatched        474
  Events_lost                0

+SLDS27BI : TransactionVision Sensor: MQIMS IVP1 Sensor Driver is ending
IEF404I TVISIONM - ENDED […and other MVS messages issued when stopping a job]
SLDS405I TVISION MQIMS IVP1 TransactionVision sensor termination complete.
```

Inquire

```
MODIFY tvm_jobname,INQUIRE STATISTICS [sensortype] [SYSID(sysid) |
IMSID(imsid)]
```

where

➤ **tvm_jobname** is the MVS jobname specified or defaulted on the start
command for the TVM job to which the inquire is directed.

➤ **INQUIRE** is the command name and is required.

➤ **sensortype** is the Sensor type and is required. Specify MQBATCH, MQIMS,
or CICS. If sensortype is omitted, all Sensor types are queried.

➤ **imsid** or SYSID may be specified to identify the MQIMS or CICS Sensor of
which you wish to inquire. If IMSID and SYSID are omitted, all Sensors of
the specified type are queried.

Example:

```
F TVISION,I ST
```

The inquire statistics command will display the following:

```
SLDS448I TVISION MQBATCH Sensor statistics:
  Events in queue:            56
  Events collected:          530
  Events dispatched:         474
  Events_lost:                 0
SLDS448I TVISION MQIMS IVP1 Sensor statistics:
  Events in queue:           175
  Events collected:         1068
  Events dispatched:         893
  Events_lost:                 0
```

Sensor Operations

1 The TransactionVision Manager must be started before any Sensors are
started.

2 The TransactionVision Manager is started with a TVID parameter that uniquely identifies the instance. Multiple concurrent TransactionVision Managers may be started, which may be desirable for test and production environments, for example. However, within the same environment, resources will be most efficiently utilized and operations simplified by running multiple Sensors under control of one TransactionVision Manager.

3 Only one Sensor at a time is allowed to use a particular IMSID or SYSID.

4 The prohibition of IMSID/SYSID rules will be enforced across all instances of the TransactionVision Manager, so you cannot start a Sensor to monitor a specific IMS system or CICS region if a Sensor is already monitoring that IMS system or CICS region regardless of which TransactionVision Manager instances are involved. Also, only one TransactionVision Manager at a time can monitor WebSphere MQ batch jobs.

5 The TVID must not be the same value as the IMSID or SYSID value of any Sensor. For future considerations, it is highly recommended that the TVID be unique among all TransactionVision Manager TVIDs, CICS SYSIDs, IMS IDs, and all MVS subsystem IDs at your site.

6 Stopping the TransactionVision Manager will automatically stop all the Sensors under its control.

7 When a Sensor is stopped, the Sensor manager will signal its corresponding Sensor application environment component to cease further data capture, then enter quiesce mode until the Sensor driver retrieves all the events in the buffer queue. When all events have been retrieved, which may take some time, the Sensor driver will terminate, then the Sensor manager will terminate.

8 TransactionVision Manager termination will not complete until all Sensors under its control have terminated.

9 If you cancel the TransactionVision Manager, all Sensors under its control will immediately terminate and all remaining events in the buffer queue will be discarded. Furthermore, canceling the TransactionVision Manager leaves some common storage control blocks in place with the result that the first subsequent WebSphere MQ call from a running batch or IMS applications that is bound to a TransactionVision WebSphere MQ stub will cause a Sensor error message to be displayed. The application, itself, is otherwise unaffected; the WebSphere MQ call will proceed normally—as it would have without the Sensor. All subsequent calls to the stub within the

same job will not generate the error message. The Sensor disables itself and passes through all calls to WebSphere MQ. Running subsequent jobs or restarting the IMS dependent regions without fixing the problem will again result in an error message on the first call by an application. Restarting the TransactionVision Manager cleans up the leftover control blocks and avoids any subsequent errors in the Sensors. Alternatively, you may run a reset procedure that cleans up the control blocks without starting the TransactionVision Manager. A sample reset procedure is provided in the TVISIONR member of the hlq.SSLDPROC library.

Buffer Queue Considerations

To minimize the overhead of Sensor components in the application environments, the Sensors store all captured transaction events in a data space referred to as the buffer queue. Each Sensor has exclusive use of a data space.

The buffer queue is configured as a number of queue blocks of a certain size. Both of these dimensions are specified in parameters on the start Sensor command. MAXQBLKS specifies the maximum number of queue blocks and QBLKSIZE specifies the size of each queue block in megabytes. The maximum size of the data space used, in megabytes, is the product of MAXQBLKS and QBLKSIZE and must not exceed 2 GB. The default MAXQBLKS is 5 and the minimum is 3. The default QBLKSIZE is 3 and the minimum is 1. Therefore, the minimum data space size requirement is 3MB, which is suitable only for low volume testing.

The correct size of the buffer queue varies widely depending on the transaction throughput of the monitored application environment; the number, types, and sizes of events collected; the throughput of the Sensor driver; and the responsiveness of the buffer queue management functions in the Sensor manager. These variables will be discussed in more detail later, but before attempting laborious calculations using what may be mere guesses as your parameters, consider taking a trial and error approach. A generous allocation at startup can compensate for a lack of precision in estimating the event workload while not costing any extra overhead.

At Sensor startup, three queue blocks are allocated. When the TransactionVision application environment components are run, they store events into the first queue block and then to the next, etc. Concurrently, the Sensor driver is retrieving events from the buffer queue in the same sequential order that they were stored (FIFO). The buffer queue management functions are invoked every hundredth of a second to check the state of the buffer queue. If the queue block currently receiving events is the last queue block allocated, an additional queue block is allocated provided that the total number allocated doesn't exceed the MAXQBLKS specification. Any queue block that has had all its events retrieved is freed, reducing the total number of allocated blocks. Therefore, the size of the buffer queue expands and contracts as traffic dictates and, regardless of the maximum allowed, no more space is used than is required-beyond the minimum of three queue blocks. A generous allocation at startup can compensate for a lack of precision in estimating the event workload while not costing any extra overhead.

TransactionVision Manager Startup Procedure

When you start the TransactionVision Manager (TVM) you must assign it a TVID that is unique among all concurrent instances of TVM. The TVID is passed as the single parameter to the main program, SLDPTVM. A sample procedure, TVISION, is provided to start TVM as follows:

```
//TVISION  PROC TVID=TV01,TVHLQ=TVISION
//TVISION  EXEC PGM=SLDPTVM,TIME=1440,PARM='&TVID'
//STEPLIB  DD   DISP=SHR,DSN=&TVHLQ..SSLDAUTH
//       PEND
```

TVM requires one parameter, a four-character TVID value that is unique for all concurrently running instances of TVM. The TVID must also be different from the ids assigned to any concurrently running Sensor. For future considerations, it is highly recommended that the TVID be unique among all TVMs, CICS SYSIDs, IMSIDs, and all MVS subsystem IDs at your site.

The TVID is defined as a procedure keyword parameter and defaults to TV01. A START TVISION operator command will start TVM with a TVID of TV01. A START TVISION,TVID=TV02 operator command will start TVM with a TVID of TV02. If both commands are issued two instances of TVM are started and both have the same jobname, TVISION.

To stop TVM or to start and stop Sensors under control of TVM requires issuing STOP or MODIFY commands specifying the jobname of the TVM to which the command is directed. If multiple TVMs are started with the same jobname, the command is directed to all of them. To avoid this problem specify a unique jobname on the START command. It is recommended to use the TVID as the jobname. For example:

```
START TVISION,JOBNAME=TV01,TVID=TV01
START TVISION,JOBNAME=TV02,TVID=TV02
```

Sensor Driver Startup Procedure

When you issue a start Sensor command to TVM, the Sensor driver component is started in a separate address space. The Sensor manager automatically starts the Sensor driver using the procedure specified on the start Sensor command (see "Start Sensor" on page 217). The following parameters from the start Sensor command are passed to the Sensor driver procedure: **SYSID**, **QMGR**, **CONFIGQ**.

The **DRVRPROC** parameter on the start Sensor command specifies the procedure to be invoked to start the Sensor driver. The default procedure names are **TVISIONM** for WebSphere MQ Sensors and **TVISIONC** for the CICS Sensor, samples of which are provided in the hlq,SSLDPROC library as follows:

```
//TVISIONM   PROC SYSID=,QMGR=CSQ1,
//              CONFIGQ=TVISION.CONFIGURATION.QUEUE,
//              TVHLQ=TVISION,MQHLQ=CSQ520
//TVISIONM   EXEC PGM=SLDPMDR,TIME=1440,
//              PARM='&SYSID &QMGR &CONFIGQ'
//STEPLIB    DD  DISP=SHR,DSN=&TVHLQ..SSLDLOAD
//              DD  DISP=SHR,DSN=&MQHLQ..SCSQAUTH
//              PEND
//TVISIONC   PROC SYSID=,QMGR=CSQ1,
//              CONFIGQ=TVISION.CONFIGURATION.QUEUE,
//              TVHLQ=TVISION,MQHLQ=CSQ520
//TVISIONC   EXEC PGM=SLDPCDR,TIME=1440,
//              PARM='&SYSID &QMGR &CONFIGQ'
//STEPLIB    DD  DISP=SHR,DSN=&TVHLQ..SSLDLOAD
//              DD  DISP=SHR,DSN=&MQHLQ..SCSQAUTH
//              PEND
```

The **SYSID**, **QMGR**, and **CONFIGQ** are defined as procedure keyword parameters with the indicated defaults. Remember that this procedure is automatically invoked by the Sensor manager; you should not invoke this procedure by any other means. For example, you should not issue a **START TVISIONM** command.

You may customize this procedure or create multiple procedures according to the communications links between the Sensor drivers and TransactionVision Analyzers.

The Websphere MQ queue manager (**QMGR**) and configuration queue (**CONFIGQ**) parameters determine which Analyzer(s) communicate with the Sensor drivers. In the sample procedure, the **QMGR** and **CONFIGQ** parameters can be overridden by specification of the same parameters on the **start Sensor** command and the **SYSID** parameter must be supplied from the **start Sensor** command. The **SYSID**, **QMGR**, and **CONFIGQ** parameters

are passed to the Sensor driver main programs, SLDPMDR or SLDPCDR, as space-delimited positional parameters, which allows the omission of only the last parameter, **CONFIGQ**. If you omit this parameter, it defaults to **TVISION.CONFIGURATION.QUEUE.** When customizing these procedures or creating your own Sensor driver procedures, you should observe the following conventions:

➤ Always specify all three parameters—**SYSID**, **QMGR**, and **CONFIGQ**—as procedure parameters.

➤ Always specify a **SYSID** parameter default of **null**.

➤ Always specify a non-null default **QMGR** parameter.

➤ Always pass at least the **SYSID** value and the **QMGR** value in the **PARM** field to **SLDPCDR**.

Note: The **SYSID** parameter in the Sensor driver procedures is used generically—as a **SYSID** or an **IMSID**.

21

Installing and Configuring Sensors on BEA Tuxedo

This chapter includes:

➤ Preparing for the Installation on page 229

➤ Running the Installation on page 230

➤ Rebinding the Tuxedo Sensor on page 231

➤ Uninstalling Sensors on page 231

Preparing for the Installation

You start the Sensor installation from the Business Availability Center product installation disk or from the Downloads page in Business Availability Center.

The following table shows the installation file names for the BEA Tuxedo packages for each supported platform:

Platform	Files
AIX	tvtux_800_aix_power.tar.gz
HP-UX	tvtux_800_hpux_parisc.tar.gz
Solaris	tvtux_800_sol_sparc.tar.gz

Running the Installation

1 Change to the directory location of the TransactionVision installation files (either a DVD device or download directory). NOTE: On Solaris and HP-UX, you must instead copy the installation files from the DVD device to a temporary directory on your host's hard drive.

2 Log in as superuser:

   su

3 Unzip and untar the packages for your platform; see "Preparing for the Installation" on page 229. For example:

gunzip tvtux_800_hpux_parisc_tar.gz

4 Enter the following command to begin the installation procedure:

   ./tvinstall_800_unix.sh

After the package is unzipped, a menu representing the available components is displayed. For example:

```
The following TransactionVision packages are available for installation:

1. TransactionVision Analyzer
2. TransactionVision UI/Job Server
3. TransactionVision WebSphere MQ Agent
4. TransactionVision BEA Tuxedo Agent
5. TransactionVision User Event Agent

99. All of above
q. Quit install

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

5 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 want 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.

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

Rebinding the Tuxedo Sensor

Unlike the WMQ Sensor, the Tuxedo Sensor installation does not automatically call the rebind script.

Run the **rebind_tux_sensor** script to relink the Sensor library. For more information about this script, see Appendix A, "Utilities Reference".

Uninstalling Sensors

To uninstall the Sensor or any other TransactionVision components on the host, perform the following steps:

1 Log in as superuser:

> **su**

2 Enter the following command:

> ./tvinstall_800_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 available for installation:

1. TransactionVision Analyzer
2. TransactionVision UI/Job Server
3. TransactionVision WebSphere MQ Agent
4. TransactionVision BEA Tuxedo Agent
5. TransactionVision User Event Agent

99. All of above
q. Quit install

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

3 Type the number associated with the TransactionVision package you want 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.

Configuring BEA Tuxedo Sensors

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 BEA Tuxedo Sensor library.

The required configuration consists of the following:

➤ Configure the application environment to load the Sensor library

➤ Set the Analyzer Configuration Service URL

An optional configuration is:

➤ Filter sensitive data

Configure the Application Library to Load the Sensor Library

The BEA Tuxedo Sensor library is dynamically loaded at runtime by including its library search path before the standard BEA Tuxedo Sensor library search path. The standard BEA Tuxedo Sensor library is dynamically loaded by the Sensor library.

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 BEA Tuxedo Sensor 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
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

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

Platform	Environment Variable	Default Directory
HP-UX	SHLIB_PATH	/opt/HP/TransactionVision/lib64:/opt/mqm/lib64
IBM AIX	LIBPATH	/usr/lpp/HP/TransactionVision/lib64:/usr/lpp/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 BEA Tuxedo library path. See Troubleshooting in *Using TransactionVision*.

Setting the Sensor Connection URL

You must modify the BEA Tuxedo Sensor property file (TVISION_HOME/config/TuxedoSensor.properties) and set the **transport** option in the "static configuration" section to the Sensor connection URL.

The Sensor Connection URL is defined as part of the communication link definition. To obtain the URL, you can access the communication link template and copy from the Connection URL field:

Filter Sensitive Data

The BEA Tuxedo Sensor is configured by default to collect payload (user) data from monitored calls. You may wish to disable this capability if the user data contains sensitive information. To do so, modify the BEA Tuxedo Sensor property file (TVISION_HOME/config/TuxedoSensor.properties) and set the **collectUserData** option in the "static configuration" section to "no".

22

Installing and Configuring the .NET Agent

The .NET Agent combines the capabilities of the Diagnostics .NET Probe and the TransactionVision .NET Sensor into a single component. The .NET Agent can simultaneously serve as both the TransactionVision Sensor and the Diagnostics Probe on a .NET host.

The .NET Agent provides a low-overhead capture solution that works with HP Software's Business Availability Center applications. The .NET Agent captures events from a .NET application and sends the event metrics to the TransactionVision Analyzer or to the Diagnostics Server, or both.

This chapter includes:

About .NET Agent Installer

During the .NET Agent installation the following setup and configuration is done for you:

➤ Discovery of ASP.NET applications. The installer attempts to automatically detect the ASP.NET applications on the system where the agent is installed.

➤ Default agent configuration.

   ➤ The installer configures the agent to capture basic ASP.NET/ADO/WCF workload for each of the ASP.NET application detected. The agent configuration is controlled using the **probe_config.xml** file. See "Configuring the .NET Agent" on page 245.

   ➤ Standard .NET application instrumentation. Default aspnet.points and adonet.points files are installed and enabled providing standard instrumentation to allow you to start monitoring ASP.NET applications. The points files control the workload that the agent will capture for the application.

   The default points ASP.NET.points and Ado.Points need to be enabled to generate TransactionVision events. These are enabled by default.

   To generate .NET Remoting Events you will also need Remoting.points and will have to setup the application for instrumentation. Refer to the Remoting documentation for this

➤ Optional configuration. Modify the agent configuration in the probe_config.xml file. See "Configuring the .NET Agent" on page 245.

Installing the .NET Agent

The following section provides detailed instructions for installing the .NET Agent.

This section includes:

➤ "Installing the Agent to Work in a TransactionVision Environment" on page 243

Preparing for the Installation

You install the .NET Agent on the host machine of the application that you want to monitor.The overhead that the Sensor for .NET (.NET Probe) imposes on the system being monitored is extremely low.

The following are the recommendations for memory and disk space that support the Sensor's processing:

Platform	All Platforms.
Memory	60 MB Additional RAM
Free Hard Disk Space	200 MB Additional Space
.NET Framework	1.1 or later

Important: The .NET Agent includes a SOAP Extension Handler. Installing the .NET Agent may cause existing Web Applications that are using SOAP to restart.

Starting the Installation

You start the installer from the Business Availability Center product installation disk or from the Downloads page in Business Availability Center.

To start the installation from the product installation disk:

1 Locate the installation package file for your platform:

Platform	Files
32-bit .NET	DotNetAgentSetup_x86_8_00.msi
64-bit .NET	DotNetAgentSetup_x64_8_00.msi

2 Run the file in the root directory of the installation disk.

To start the installation from the Download Center - HP - BTO Software page:

1 Enter **TransactionVision** in the Keyword field, and **Trial software** in the Refine Search by resource type box.

2 Click the appropriate link to download the .NET Agent installer for your platform.

Running the Installation

After you have launched the installer, you are ready to begin the main installation procedure.

To install the .NET 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 Provide the location where you want the Agent installed.

By default, the Agent is installed in **C:\MercuryDiagnostics\.NET Probe**.

Accept the default directory or select a different location either by typing in the path to the installation directory into the **Installation HP Diagnostics/ TransactionVision Agent for .NET to** box, or by clicking **Browse** to navigate to the installation directory.

Click **Next** to continue.

3 Select the .NET Agent features that you want to install.
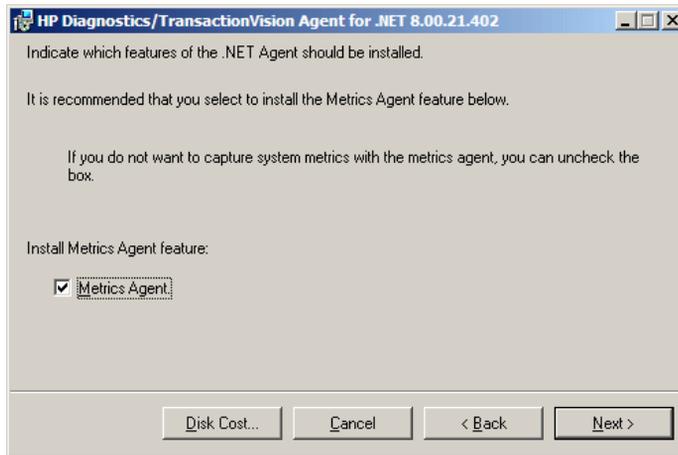
To capture system metrics on the host machine without monitoring any applications, select **Metrics Agent** only.

If you want to check the amount of available disk space on the drives of the host, click **Disk Cost**. Use this functionality to make sure that there is enough room for the Agent installation.

4 Select whether you want to install the Agent as the Diagnostics Profiler only, as a probe reporting to a Diagnostics Server, or as an Agent reporting to the TransactionVision Analyzer.
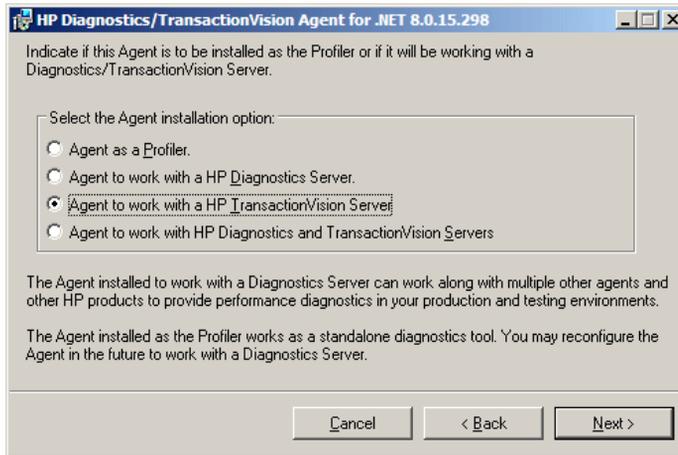
➤ Select **Agent to work with an HP TransactionVision Server**.

Click **Next** to continue.

Note: If you are installing the Agent as the Diagnostics Profiler or to work with a Diagnostics Server, see the *Diagnostics Installation and Configuration Guide*.

Installing the Agent to Work in a TransactionVision Environment

If you are installing the Agent to work with in a TransactionVision environment, continue with the following procedure.

1 The Configure the .NET Agent for TransactionVision dialog appears.
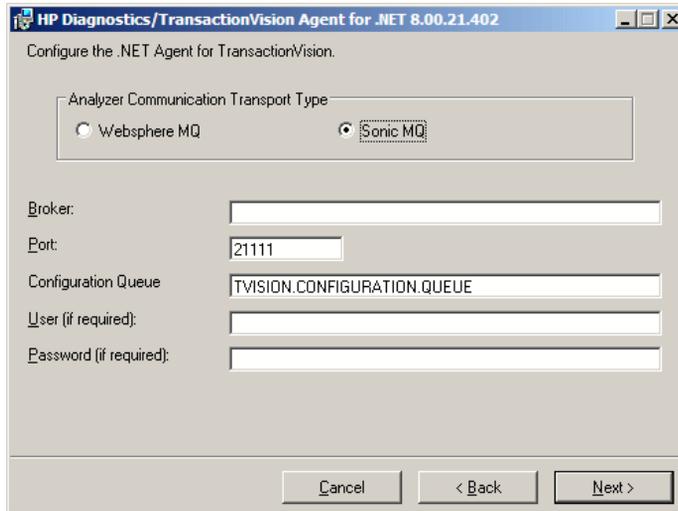
2 Choose the Messaging Middleware Provider. Options are: WebSphere MQ and SonicMQ.

SonicMQ is included with the .NET Agent. If you specify this option, the Sonic MQ .NET client (Sonic.Client.dll - Progress SonicMQ .NET Client, version 7.6.0.112) is installed as part of the Agent installation.

A third-party WebSphere MQ installation can be used instead. In this cae, you must install the MQ series .NET client (amqmdnet.dll - WebSphere MQ Classes for .NET, version 1.0.0.3) on the host being monitored.

By default, SonicMQ is selected.

3 For SonicMQ, enter the following:

Broker. Host name on which the Sonic broker is running. Typically this will be the Analyzer hostname.

Port. The port on which the broker communicates. By default, 21111.

Configuration Queue. Name of the configuration queue. By default, TVISION.CONFIGURATION.QUEUE.

User. User id if required by SonicMQ installation for connection. By default, no username is required.

Password. Password if required by SonicMQ installation for connection. This is in the obfuscated form created by using the PassGen utility. By default, no password is required.For more information about **PassGen**, see "Utilities Reference" on page 327.

4 For WebSphere MQ, enter the following:

Host. The host on which the WebSphere MQ queue manager resides.

Port. Port number for WebSphere MQ queue manager.

Configuration Queue. Name of the configuration queue.

User. User id if required by WebSphere installation for connection.

Password. Password if required by the WebSphere MQ installation for connection. This is in the obfuscated form created by using the PassGen utility. For more information about **PassGen**, see "Utilities Reference" on page 327.

Websphere MQ channel. Channel name for WebSphere MQ queue manager.

Websphere MQ Q Manager. CCSID for WebSphere.

Click **Next** to continue.

5 The pre-installation summary dialog appears.

Click **Install** to proceed with the installation.

After the installation completes, you must restart IIS (see "Restarting IIS" on page 251). You can perform any custom configuration if desired as described in the section that follows.

Configuring the .NET Agent

The default configuration of the .NET Agent allows you to begin capturing performance metrics for the monitored application. The Agent's configuration can be customized to suit the configuration of your environment and the performance issues that you would like to diagnose.

To override the default configuration, access the **<sensor_install_dir>/etc/ probe_config** file.

Use the following elements:

➤ <tv>, <time skew>, <transport>

These elements are supported for TransactionVision only.

➤ <logging>, <modes>

These elements are supported by TransactionVision and Diagnostics.

For complete information on the **probe_config** file, see the *HP Diagnostics Installation and Configuration Guide*.

<tv> element

Purpose

Configure the .NET Agent for use with TransactionVision.

Attributes

Attributes	Valid Values	Default	Description
eventthreads	number	3	(Read on startup) The number of threads spawned by the Agent to send events to the Analyzer.
eventthreadsleep	number	100	(Dynamic) The time in milliseconds the event thread sleeps after sending a message(event package).

Attributes	Valid Values	Default	Description
eventmemorythreshold	number	250,000,000	(Dynamic) The memory consumed by the internal buffer(Q) after which the Agent will try and send the message on the application thread.
configthreadsleep	number	10,000	(Dynamic) The time in milliseconds the event thread sleeps after browsing the configuration queue.

Elements

Number of Occurrences	1 (one)
Parent Elements	ProbeConfig
Child Elements	transport, timeskew

Example

```
<tv eventthreads="3" eventthreadsleep="80"
eventmemorythreshold="25000000" configthreadsleep="10000" >
    <timeskew historysize="24" checkinterval="300000" latencythreshold="100"
        retrythreshold="8"/>
        <transport type="sonicmq"
        connectionstring="broker=myhost.mydomain.com;
        port=21111; user=; password=;
        configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
</tv>
```

<timeskew> element

Purpose

Calculates the time difference between the time server and the host on which the .NET Agent is running. The frequency of checking with the time server can be configured.

Attributes

Attributes	Valid Values	Default	Description
historysize	number	24	(Read on startup) number of time skew samples to store and compare for best sample.
checkinterval	number	300,000 ms.	(Dynamic) The time in milliseconds to wait before checking the time server for the skew time calculation.
latencythreshold	number	100 ms.	(Dynamic) The maximum time in milliseconds a reply from a time server can take for a valid time skew value.
retrythreshold	number	8	(Dynamic) Number of times to try when request to time server fails.

Elements

Number of Occurrences	1 (one)
Parent Elements	tv
Child Elements	none

Example

<timeskew historysize="24" checkinterval="300000" latencythreshold="100" retrythreshold="8"/>

<transport> element

Purpose

Configure the events channel used by TransactionVision.

Attributes

Attributes	Valid Values	Default	Description
type	mqseries sonicmq	sonicmq	The event transport provider being used by the Agent.
connectionString	See below.		The connection information for the event transport provider.

conectionString Syntax when type=sonicmq

```
broker = <broker>; port = <port>; user = <user>; password =<password>;
configurationQueue = <configurationQueue>
```

Where:	Is:
broker	Host name on which the Sonic broker is running. Typically this will be the Analyzer hostname.
port	The port on which the broker communicates. By default, 21111.
user	User id if required by SonicMQ installation for connection. By default, no username is required.
password	Password if required by SonicMQ installation for connection. This is in the obfuscated form created by using the PassGen utility. By default, no password is required. For more information about **PassGen**, see "Utilities Reference" on page 327.
configurationQueue	Name of the queue which has the configuration messages for the .NET TransactionVision Agent.

conectionString Syntax when type=mqseries

host= <host>; queuemanager=<queuemanager>; port= <port>; channel=,channel>

Where:	Is:
host	Host on which the TransactionVision configuration queue is hosted.
queuemanager	Name of the queuemanager.
port	MQSeries port on which the QueueManager communicates.
channel	MQSeries channel which will be used to communicate.
configurationQueue	Name of the queue which has the configuration messages for the .NET TransactionVision Sensor.

Elements

Number of Occurrences	1 (one)
Parent Elements	tv
Child Elements	None

Example

For SonicMQ:

```
<transport type="sonicmq" connectionstring="broker=brokerHost;
    port=21111; user=; password=;
    configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
```

For MQ Series:

```
<transport type="mqseries" connectionstring="host=mqHost;
    queuemanager=; port=1414; channel=TRADING.CHL;
    configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
```

<logging> Element

The "TVDEBUG" tag can be specified for tracing TransactionVision specific code in the .NET Agent.

For example,

```
<logging level="TVDEBUG"/>
```

For more information about the **<logging>** element, see the *HP Diagnostics Installation and Configuration Guide*.

<modes> Element

The "tv" modes tag specifies the TransactionVision mode for the Agent.

For example,

```
<modes tv="true"/>
```

When this is the only mode specified, the Agent works in a "TV only" mode, that is, the Diagnostics profiler and the Diagnostics probe are disabled and only TransactionVision events are generated.

When other modes are specified, both TransactionVision and Diagnostics events are generated.

For more information about the **<modes>** element, see the *HP Diagnostics Installation and Configuration Guide*.

.NET Remoting Client and Server Applications

The following example shows the configuration changes required for TransactionVision Transport in a .NET Remoting environment.

Changes to probe_config.xml:

```
<tv eventthreads="5" eventthreadsleep="100"
eventmemorythreshold="25000000" configthreadsleep="60000" >
    <timeskew historysize="20" checkinterval="300000" latencythreshold="100"
        retrythreshold="6"/>
        <transport type="sonicmq"
```

```
      connectionstring="broker=myhost.mydomain.com;
      port=21111; user=; password=;
      configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
</tv>
<process name="TVRemotingClient">
   <points file="Remoting.points"/>
   <points file="TVRemotingClient.points"/>
   <modes enterprise="true" pro="true" tv="true"/>
</process>
```

Enabling Correlation of .NET Events

By default, .NET Events are not correlated. To enable the user correlation bean, make the following changes to the configuration on the Analyzer host:

➤ uncomment the .NET Agent section in EventCorrelationDefinition.xml file

➤ uncomment the following tag in Beans.xml :

```
<Attribute name="UserCorrelationBean"
value="com.bristol.tvision.services.analysis.eventanalysis.XMLRuleCorrelationB
ean"/>
```

For more information about the EventCorrelationDefinition.xml file, see the *TransactionVision Advanced Customization Guide*.

Restarting IIS

After installing the .NET agent and modifying the configuration or creating custom instrumentation, as needed, restart IIS before using the .NET Agent with ASP.NET applications.

To restart IIS from the command line or from the Start > Run menu, type:

```
iisreset
```

This command restarts the Web publishing service but does not immediately start the Sensor. The next time that a Web page in the application is requested, the Sensor is started, the applications are instrumented, and the Sensor reads the Configuration Queue Messages from the Analyzer.

Note: ASP.NET automatically restarts applications under various circumstances, including when it has detected that applications have been redeployed, or when applications are exceeding the configured resource thresholds.

Determining the Version of the .NET Agent

When you request support, it is useful to know the version of the components that you have installed.

To determine the version of the .NET Agent:

➤ Right-click the file **<.net_agent_install_dir>\bin\HP.Profiler.dll**, and display the component version information by selecting **Properties** from the menu.

Uninstalling the .NET Agent

To uninstall the .NET Agent, follow these steps:

1 Stop all Web applications that are using SOAP.

2 Use the **Add/Remove Programs** utility in Windows.

Remove the HP TransactionVision/Diagnostics Agent for .NET program.

3 Restart the Web applications.

SSL Configuration for .NET Agents

If the .NET Agent is using SonicMQ for the messaging middleware, SSL can be enabled. See the *HP Business Availability Center Hardening Guide* PDF.

23

Installing and Configuring Sensors on NonStop TMF

This chapter includes:

➤ About the NonStop TMF Sensor on page 256

➤ Preparing for the Installation on page 256

➤ Installing the NonStop TMF Sensor on page 257

➤ Startup/Shutdown on page 258

➤ Configuring the NonStop TMF Sensor on page 259

About the NonStop TMF Sensor

All audited transactions on HP NonStop Guardian systems are logged to audit trail files. This include every kind of database access from indexed (b-tree) files to SQL databases. The product that generates the audited trail data is called Transaction Monitoring Facility (TMF), and because TMF protects any audited files on the NonStop system, it is the repository of all changes. To support On Line Transaction Processing (OLTP) applications, TMF can monitor thousands of complex transactions sent by hundreds of users to the audited trail database. TMF also provides transaction protection, database consistency, and database recovery critical in high-volume transaction processing.

The NonStop TMF Sensor monitors the TMF audited trail database for changes (adds, deletes, and modifies) on Enscribe databases. The NonStop TMF Sensor does not monitor changes to SQL relational databases, only Enscribe file monitoring is supported.

Because TMF records all audited changes to the audited trail database, the NonStop TMF Sensor is able to read the audited trail records of interest and create TransactionVision user events which are forwarded to the TransactionVision Analyzer for analysis.

Preparing for the Installation

You start the Sensor installation from the Business Availability Center product installation disk or from the Downloads page in Business Availability Center.

The following table shows the installation file names for the NonStop TMF packages for each supported platform:

Platform	Files
Guardian	tvtmf_800_nonstop_guardian.zip

Installing the NonStop TMF Sensor

1 Unzip the install package to a temporary directory.

2 Open FTPSCRIPT.txt and make the following changes:

➤ <NonStop Username> to a valid NonStop user.

➤ <NonStop Userpassword> to the user's password.

➤ <NonStop destination volume.subvolume> to the destination volume and subvolume.

3 FTP using the modified FTP script.

For example:

ftp -s:FTPSCRIPT.txt <NonStop host DNS/IP>

4 Logon to the NonStop host as super.super and go to the temporary installation $volume.subvol.

5 Run the "INSTALL" TACL macro and provide the destination installation volume/subvolume, and the host DNS name or IP address. For example:

```
135> run install
This program collects configuration information in order to set up the TransactionVision
sensor for the NonStop environment. This includes:
- Installation volume and subvolume where to install the sensor
- Location of the TransactionVision Analyzer Configuration Service

You will be prompted to input required configuration parameters.
If the previous value is provided in (), pressing <Enter> will set
the parameter to the previous value.


Enter the installation volume (): $data02.tvision


Enter the analyzer configuration service DNS or IP address (): 15.178.196.101

Sensor installation complete!
```

Startup/Shutdown

The NonStop TMF Sensor has three TACL macros that are used for startup and shutdown:

➤ **COLDMON**. Starts the Sensor for the first time,

➤ **STRTMON**. Starts the Sensor after it has been stopped.

➤ **STOPMON**. Stops the Sensor.

COLDMON

This macro purges all temporary data used by the Sensor (queued events) and establishes the first audit file to begin looking for events that meet the User Event filter criteria.

Start the cold start macro, by going into the $volume.subvol where the NonStop Sensor is installed, and run the macro, (e.g. run "COLDMON").

After starting "COLDMON" you will be prompted to input the audit trail file to begin searching for events.

For example:

```
Select TMF Audittrail file to commence scan from:
File 1:  $AUDIT.ZTMFAT.AA000006
File 2:  $AUDIT.ZTMFAT.AA000007
File 3:  $AUDIT.ZTMFAT.AA000008
Select number from list above :
```

If you know you would like the Sensor to begin searching in a specific audit trail file, input that file, otherwise, simply begin with the first file displayed (select '1').

The Sensor begins searching for events that fit the filter criteria up to the present. The first time the Sensor is started; there will not be any events to collect until a filter is configured. The Sensor will scan through the audit files, and wait for new audit trail records.

STRTMON

When the Sensor is stopped, it saves the last position of the record in the audit trail file it was reading. That way, when it is restarted, it will start scanning for events from the position where it let off. To start the NonStop TMF Sensor, after it has been stopped, run the TACL macro STRTMON, for example:

```
run STRTMON
```

STOPMON

To stop the NonStop Sensor, run the TACL macro STOPMON, for example:

```
run STOPMON
```

Configuring the NonStop TMF Sensor

The NonStop TMF Sensor is configured from two sources:

➤ The INSTALL TACL macro, which sets up values in the startup macros, (COLDMON/STRTMON).

➤ Data received from the Analyzer.

The TACL INSTALL macro sets all required values in the COLDMON/ STRTMON macros so in most cases users do not need to make any additional changes to the startup macros. After installation of the Sensor, the Sensor will work with all of the values set by the INSTALL macro.

The NonStop Sensor reads the Analyzer configuration service to determine, among other things, transport details for sending events and what data collection filtering to perform. On the Analyzer, transport details are part of the Communication Link configuration. The data collection filtering information comes from the Data Collection Filter configuration. See Communication Links in *Using TransactionVision*.

24

Configuring WebSphere MQ Sensors

TransactionVision provides two types of WebSphere MQ Sensors:

➤ WebSphere MQ Sensor library

➤ WebSphere MQ API Exit

This chapter includes:

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\HP\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 Troubleshooting in *Using TransactionVision* 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, 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: 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 ADDLIBLE or CHGLIBL 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: 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 26, "Configuring Agent and 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 284.

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: 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 269.

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

Note: To use a different configuration queue name for each CICS region, see "Using Separate Configuration Queues for Each CICS Region" on page 269

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 && (dfheiprt->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 "Communication Links" in *Using TransactionVision*.

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 279 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.

Note: 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 273). 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 278.

➤ 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 273 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.

Note: When using the WebSphere MQ API Exit Sensor, if the exit_sensor.allow file exists and is left empty, no programs will be monitored.

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
runmqlsr
```

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: 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 "Data Collection Filters" in *Using TransactionVision*.

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 MQCONNX.

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:
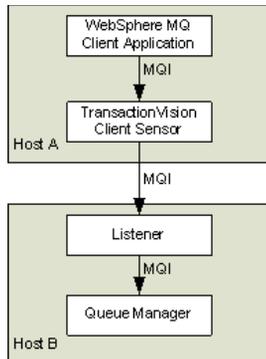
In general, applications that make use of WebSphere MQ client binding will communicate with a "listener" process (also known as the channel responder) that runs on the same host as the targeted queue manager. All WebSphere MQ activities (that is, MQI calls) are forwarded to and processed by the listener 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:
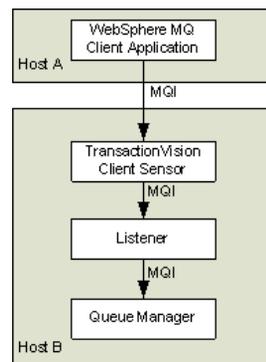
In this case, the Sensor is deployed on the host where the listener process and queue manager reside. Instead of direct monitoring of the client application, the Sensor monitors the listener program instead. Note that the TransactionVision Server Sensor is deployed. The Sensor intercepts and reports 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 145.

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

Note: 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 287.

➤ 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)l, 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 UI/Job Server 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 UI/Job Server. However, you may disable the Sensor from the system console with the MODIFY command: F TVISIONB,DISABLE MQIMSBDG. 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 MQIMSBDG.

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 by 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.IMSBridgeEntryModifer Bean 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 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.

25

Configuring the Proxy Sensor

This chapter includes:

➤ About the Proxy Sensor on page 293

➤ Application Requirements on page 294

➤ Enabling the Proxy Sensor on page 294

➤ Configuring the Proxy Definition File on page 294

➤ Configuring the User Interface on page 297

About 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.

293

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 want to include in your TransactionVision analysis.

Note: 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.

Element	Required?	Attributes	Description
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.
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 *matchMsgIdToCorrelId* or *matchCorrelIdToMsgId*

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."

26

Configuring Agent and Sensor Logging

This chapter includes:

➤ Log Files on page 299

➤ Circular Logging on page 300

➤ Trace Logging on page 302

➤ Configuring Separate Log Files for Multiple Sensor Instances on page 302

➤ Using Windows and UNIX System Logs on page 303

Log Files

Java Agents

By default, the TransactionVision Java Agents log error and warning messages to the logs directory where you installed the Java Agent. This location is stored in the Sensor.Logging.xml file.

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

WebSphere MQ Sensors

By default, the 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 Java Agents 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* want 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 Java Agents, trace messages are sent to the Agent's log4j TraceLog.

To enable or disable Sensor trace logging in the communication link, see "Communication Links" in *Using TransactionVision*.

Configuring Separate Log Files for Multiple Sensor Instances

If multiple applications servers or JVM processes on the same machine have the Agent enabled, the Agent 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.property file. For example, for serverA set the JFM property as follows:

```
com.bristol.tvision.sensor.properties=sensor/Sensor_ser verA.properties
```

For serverB, set the JVM property as follows:

```
com.bristol.tvision.sensor.properties=sensor/Sensor_ser verB.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_se
rverA.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 catagory 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.

Part V

Security

27

Security

This chapter provides information about the security setup procedures of TransactionVision.

This chapter includes:

➤ About Security in TransactionVision on page 309

➤ SSL Configuration for TransactionVision on page 310

➤ TransactionVision User Rights Management on page 312

➤ Default Roles on page 316

➤ TransactionVision Authentication and Light Weight Single Sign On on page 317

➤ Securing TransactionVision Configuration Files on page 318

➤ Securing the TransactionVision Analyzer on page 320

➤ Securing the TransactionVision Database on page 322

About Security in TransactionVision

Security in TransactionVision is managed by the following tasks:

➤ Enabling SSL between TransactionVision components.

➤ Controlling user rights.

➤ Securing the Analyzer, configuration files, and database.

SSL Configuration for TransactionVision

The following diagram shows the paths in the TransactionVision deployment environment that are candidates for SSL.
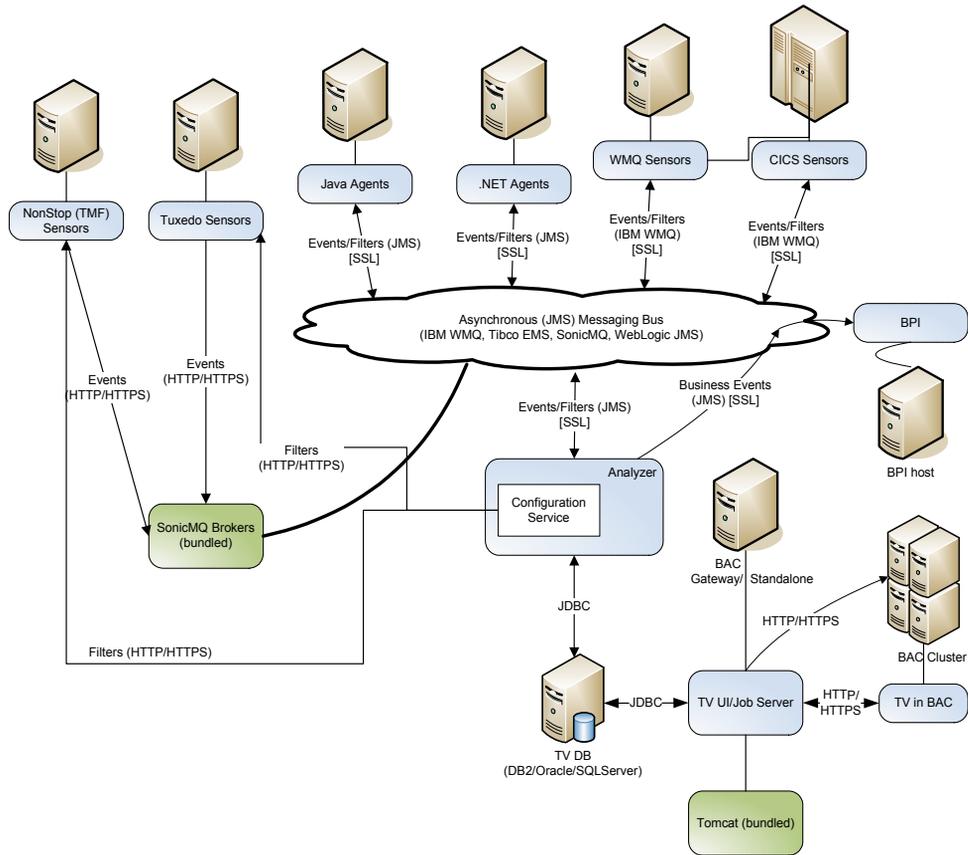
The general process to enable SSL is summarized below.

Step 1: Generate required certificates for encrypting traffic

The TransactionVision Analyzer, UI/Job Server, and SonicMQ Server each require a private key that is created as a result of generating the required certificates. If two or more of these components reside on the same host, they can share the same private key.

Step 2: Configure the TransactionVision SonicMQ Server

The TransactionVision Analyzer is bundled with Progress SonicMQ 7.5, which is installed when the Analyzer is installed.

Configure the TransactionVision SonicMQ Server to use a certificate generated in step 1.

Step 3: Configure the Analyzer

a If using the HTTPS Configuration messages service (for use with NonStop TMF Sensors), configure the certificate generated in step 1.

b To communicate with the TransactionVision SonicMQ Server, configure the Analyzer to use the certificate set up in step 2.

Step 4: Configure the UI/Job Server

a To support incoming SSL requests, configure the TransactionVision UI/Job Server with the certificate generated in step 1.

b To support outgoing SSL communication to Business Availability Center, configure the TransactionVision UI/Job Server to use a Business Availability Center public certificate and set it to use https protocol to communicate with Business Availability Center.

Step 5: Configure the Sensors/Agents

a If communicating directly with the TransactionVision SonicMQ Server, configure Sensors/Agents with the certificate set up in step 2.

b If using the HTTPS Configuration message service (for NonStop TMF Sensors), configure the Sensor to use certificate configured in Step 3a.

Details about each step are provided in the *HP Business Availability Center Hardening Guide* PDF.

TransactionVision User Rights Management

User rights control which TransactionVision pages can be seen by a particular user and whether the user can make administrative changes or view specific data. The users rights in a particular session within TransactionVision are determined by the user's login and authentication in Business Availability Center. For additional information, see the *HP Business Availability Center Hardening Guide* PDF.

Setting User Rights

Setting user or group permissions on TransactionVision resources is done by selecting the TransactionVision context in the Business Availability Center Users and Permissions tab. User or group permissions can be set on resources by assigning a pre-defined role or by granting individual operations on resources. Granting permission to the Projects or User-created Reports resource container grants the same permission to child resource instances and any subsequent instances that are automatically added.

By default, TransactionVision permission changes go into effect the next time the user logs in. To have permissions go into effect during the user's login session, update the TransactionVision PermissionsCacheMaxAge property in <TVISION_HOME>/config/ui/UI.properties to the number of minutes before TransactionVision should retrieve permission updates from Business Availability Center.

Note: Setting this value will cause TransactionVision to requery Business Availability Center for its user permissions. If the TransactionVision UI/Job Server is installed on a separate machine, and particularly if it's on a separate network from the Business Availability Center server, it may contribute to some delay seen in the response of the TransactionVision application in Business Availability Center if it is set to reload the permissions too frequently.

Within the TransactionVision context, the follow resources are available:

Note: Projects and User-create reports are containers that dynamically populated based on which projects and/or user created reports are available in TransactionVision. Project child resource instances are added/deleted from the TAS tree as new projects are created/deleted using the project wizard. The project name provided in the project wizard will be displayed in the TAS tree and the creator of the project will be granted FULLCONTROL on the new project resource instance.

User-created Report child resource instances are added/deleted from the TAS tree whenever the <TVISION_HOME>/config/ui/Reports.xml file is updated with custom report definitions. The Report element's title attribute will be displayed in the TAS tree.

Name	Operations (permissions)
Projects	If no operations are select, project will not be accessible by user VIEW (user is allowed to only view the project, which will also grant access to all built-in reports as well as view and modify queries.) CHANGE (user is allowed to edit/delete/create projects and start/stop jobs on a project) FULLCONTROL (user is allowed all available operations as well as the ability to grant and revoke operations)
User-created Reports	VIEW (user is allowed to view the report) FULLCONTROL (user is allowed all available operations as well as ability to grant and revoke operations)
Analyzer	EXECUTE (User is allowed to start/stop the analyzer) FULLCONTROL (user is allowed all available operations as well as ability to grant and revoke operations)
User Data	VIEW (user is allowed to view user data) FULLCONTROL (user is allowed all available operations as well as ability to grant and revoke operations)

User Data Permissions

Some transaction and event content collected by TransactionVision may contain data that is considered sensitive to the business. This might include items such as CC number, social security number, etc. If it is determined that TransactionVision has sensitive data that only certain people are allowed to view, this can be controlled through the User Data resource. By disabling access to user data, the information shown by TransactionVision is limited to the generic data fields that are available on all event and/or transaction data.

If User Data view permission is denied, the following will be affected:

➤ Event Details window for an event will just display: "You do not have permission to view user data."

➤ SQL Statistics Report will not display the full SQL statement.

➤ Transaction Tracking Report will not display custom columns.

➤ Detail for Business Transaction will not display custom data in the Summary section.

➤ Transaction Definition Editor Edit Match Condition page will not allow selection of a txn and will not populate value pane with txn values if classification has existing associated txn information

---

Note: The Business Availability Center Administrator role has rights to grant and revoke all TransactionVision permissions, thus Business Availability Center Administrator is effectively granted access to all of TransactionVision. If you require extra prevention to lock down User Data access but you still have a need for a user to be able to access the TransactionVision Administration tab, then you can assign the user the TransactionVision Administrator role rather than the Business Availability Center Administration role.

Default Roles

A number of default Roles are defined for assigning access to TransactionVision. These roles are described in the below table.

Role	Operations	TransactionVision Application Tabs Allowed
TransactionVision Administrator	FULLCONTROL on Projects, Analyzer and User-created Reports	Current Projects and Administration
TransactionVision Operator	VIEW on Projects and User-created Reports EXECUTE on Analyzer	Current Projects
TransactionVision User	VIEW on Projects and User-created Reports	Current Projects

In addition, pre-defined Business Availability Center roles, Administrator and System Viewer were updated to include the same operations as defined for TransactionVision Administrator and TransactionVision User respectively. TransactionVision Administrator and Business Availability Center Administrator are the only roles to have access to the Administration tab in the TransactionVision application of Business Availability Center.

Due to the sensitive nature of User Data and to minimize the possibility of accidentally granting access, only the Business Availability Center Administrator has been given User Data access (FULLCONTROL permission).

TransactionVision Authentication and Light Weight Single Sign On

In order to guarantee secure access to TransactionVision, Light Weight Single sign on (LWSSO) must be enabled. The security tokens used by LWSSO are passed to and from between TransactionVision and Business Availability Center and ensure only a correctly authenticated user accesses the pages they are authorized to access. If you intend to have Business Availability Center use LWSSO, no further configuration of TransactionVision is required, this will work automatically out of the box.

The requirements of LWSSO are that the TransactionVision and Business Availability Center servers must be accessed using a fully qualified domain name. In some deployments, it may be that using a fully qualified host name for accessing Business Availability Center and/or TransactionVision may not be desired. In these cases LWSSO in Business Availability Center is disabled. It is important to note that with regards to TransactionVision, in order for TransactionVision to be properly secured, LWSSO and fully qualified domain names are required to correctly determine user authentication and authorizations.

If for some reason you do wish to disable LWSSO, while TransactionVision will still work, in this case the credentials passed to the TransactionVision UI/Job Server will be passed in a non-secure way. Thus disabling LWSSO causes a vulnerability that can allow authentication to be bypassed, and would not be a recommended approach if you have sensitive data or need to insure that access to TransactionVision data and administration is only ever possible by those with the correct authorizations. If you do disable LWSSO in Business Availability Center, you must change the bac_lwsso_enabled setting in <TVHOME>/config/ui/UI.properties and set it to false—while bac_lwsso_enabled is set to false the TransactionVision UI/Job Server will allow less secure user authentication.

Securing TransactionVision Configuration Files

All the TransactionVision configuration files are located under the <TVISION_HOME>/config directory. When the <TVISION_HOME>/bin/ TVisionSetupInfo.[sh|bat] script is run, the location of the logs directory will be specified, where the TransactionVision log files will be found.

The TransactionVision application administrator, who should be the only person who has write access to the files, should own all TransactionVision configuration files. Generally speaking, only read permission should be granted to all the relevant TransactionVision groups.

The following table shows the suggested authorities for TransactionVision configuration files:

Directory	Transaction Vision UI/ Job Server	Transaction Vision Analyzer	Description
<TVISION_HOME>/ config/codepage/*	read	read	Codepage configuration
<TVISION_HOME>/ config/datamgr/*	read	read	Database connection configuration
<TVISION_HOME>/ config/job/*	read	none	Job manager configuration
<TVISION_HOME>/ config/license/*	read	read	TransactionVision license
<TVISION_HOME>/ config/logging/*	read	read	Logging policy
<TVISION_HOME>/ config/rmi/*	none	read	RMI policy
<TVISION_HOME>/ config/services/*	none	read	Analyzer configuration
<TVISION_HOME>/ config/setup/*	read	read	TransactionVision Setup configuration

Directory	Transaction Vision UI/ Job Server	Transaction Vision Analyzer	Description
<TVISION_HOME>/ config/ technologyconst/*	read	none	Various constant definitions
<TVISION_HOME>/ config/ typeconversion/*	read	none	Event details type conversion configuration
<TVISION_HOME>/ config/ui/*	read	none	User interface configuration, including reports settings
<TVISION_HOME>/ config/usermgr/*	read	none	User data configuration
<TVISION_HOME>/ config/xdm/*	read	read	Various xdm definitions
<TVISION_HOME>/ config/xmlschema/*	read	read	Various xsl definitions
<user defined dir>/ logs/*	write	write	TransactionVision logs

Note: <TVISION_HOME>/config/datamgr/Database.properties may contain a clear text DB2 username and password, so you must be especially careful when granting read permission to this file. For maximum security, assign the read authority only to the file owners and system administrator.

To define file access permissions for a file in Windows NT, perform the following steps:

1 Log in as the system administrator.

2 Open Windows Explorer.

3 Right-click on the properties or policy file and select Properties. The File Properties dialog opens.

4 Click the Security tab, and assign privileges to various user accounts or
security groups for the file.

To set configuration file permissions in UNIX, perform the following steps:

1 Log in as system administrator

2 Run chmod to grant the appropriate access rights to different
TransactionVision groups. Recommended settings are rw--r-- for /datamgr/
database.properties, and rw-r--r-- for all other configuration files.

3 If necessary, use chgrp to change file groups and chown to change the
ownership of the files.

Securing the TransactionVision Analyzer

The TransactionVision Analyzer communicates with TransactionVision
Sensors and Agents and processes collected event data into meaningful
analysis. The Analyzer runs as a windows service on Windows and as a
daemon on Unix.

TransactionVision uses RMI (Remote Method Invocation) to communicate
with the Analyzer on a specified port (default is 21100). The
TransactionVision UI/Job Server as well as the command line utilities then
use RMI to communicate to the Analyzer in order to control its behavior,
initiating service shutdown, allowing collection to be started, stopped, and
getting status information. For information about utilites that operate on
the Analyzer, see "Administration Utilities"in *Using TransactionVision*.

TransactionVision defines RMI security policy in the file <TransactionVision
Install Directory>/config/rmi/RMI.policy. By default, it grants the
permission java.security.AllPermission. Most significant in this is that it
allows remote access to the Analyzer to any user that has access to a
TransactionVision install. An Administrator of TransactionVision should be
aware of this, if more secure access to the Analyzer is desired, a combination
of other options can be taken to control access.

The simplest mechanism to control access to an Analyzer from the host it is running on, is file permissions for the TransactionVision install. These permissions can be adjusted as needed. For example, on UNIX, you should only allow read/write/execute permission to either the owner of the install, or those in a designated group that is authorized to manage TransactionVision.

To control access to the Analyzer from remote machines, the RMI.policy file previously mentioned can be used. By granting java.net.SocketPermission you can control from what machine remote access to the Analyzer is allowed. The below template can be used to achieve this. At minimum the RMI.policy file must grant socket permission to the machine on which the UI/Job Server is running, otherwise it will not be possible to control the Analyzer from the web interface.

```
grant {
permission java.net.SocketPermission "webserver:*", "accept, connect, listen, resolve";
  permission java.net.SocketPermission "remotepc:*", "accept, connect, listen, resolve";
  permission java.net.SocketPermission "localhost:*", "accept, connect, listen, resolve";
  permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete, execute";
  permission java.net.NetPermission "*";
  permission java.awt.AWTPermission "*";
  permission java.util.PropertyPermission "*", "read, write";
  permission java.lang.reflect.ReflectPermission "*";
  permission java.lang.RuntimePermission "*";
  permission java.security.SecurityPermission "*";
  permission java.io.SerializablePermission "*";
};
```

In this example, the machines "webserver" and "remotepc" and the local machine the Analyzer is running on are the only machines from which commands to the Analyzer would be accepted.

Outside of "java.net.SocketPermission," the other permissions represent a minimum set of permissions required by the Analyzer itself, it is not recommended that these be changed. Additional information about the specifics of these permissions can be found at http://java.sun.com/j2se/1.3/ docs/guide/security/ permissions.html.

Securing the TransactionVision Database

The objectives of securing the TransactionVision database are:

➤ Preventing unauthorized access to classified data by anyone without a business need to know.

➤ Preventing unauthorized users from committing mischief by maliciously deleting or tampering with data.

➤ Monitoring user access of data through auditing techniques.

Database access is generally controlled by user authentication and user authorization. Authentication is the process of verifying the identity of a user, whereas authorization is the process of determining whether a user has the right to access a requested resource. The parameters to perform database authentication are configured in TransactionVision. During TVisionSetupInfo you will be asked to enter the database user name and password. Both will be stored in the configuration file <TVISION_HOME>/config/datamgr/Database.properties (the password in encrypted form). The underlying mechanisms for user authentication can be different for each database product. TransactionVision supports operating system based authentication for DB2, database authentication for Oracle, and 'SQL Server Authentication' for SQL Server.

TransactionVision tables and schemas

TransactionVision uses two different kinds of database schemas: the system schema, named TVISION, and the user-defined project schemas. The TVISION schema stores non-events related information, while project schema stores project events collected by TransactionVision sensors.

The TVISION system schema can either automatically be created during TVisionSetupInfo, or manually by executing the script "<TVISION_HOME>/bin/CreateSqlScript.[sh|bat] -c -e -system. After that, you can start working with TransactionVision and create TransactionVision project schemas from either the web application or CreateSqlScript.

Database privileges

In order to run TransactionVision successfully in the given database environment, you have to make sure that the database user configured for TransactionVision has the necessary database privileges to access all required database objects. These privileges consist of:

➤ SELECT privileges on the database system catalog tables

➤ SELECT, INSERT, UPDATE, and DELETE privileges on all tables in schema TVISION

➤ SELECT, INSERT, UPDATE, and DELETE privileges on all tables in the project schemas

➤ CREATE TABLE privileges for creating the TVISION system tables at installation time and the project schema tables from the TransactionVision UI project creation wizard.

If a dedicated database is used for TransactionVision it is often adequate to choose a user with predefined, sufficient database authority (e.g. a user with the DBA role) for accessing the database. If this is not possible, the above listed privileges have to be granted specifically. The CREATE TABLE privileges are not mandatory for running TransactionVision once the tables have been created, so it is possible to avoid granting these privileges by having a database DBA create the TVISION and project tables manually. The utility 'CreateSqlScript' can generate the necessary SQL scripts to create the TVISION and project tables, as well as scripts that grant the required access privileges for the configured user to TVISION and project tables:

➤ CreateSqlScript -create -system

➤ CreateSqlScript -create -schema SCHEMA

➤ CreateSqlScript -grant -system

➤ CreateSqlScript -grant -schema SCHEMA

If the configured database user does not have the CREATE TABLE privileges, you will not be able to create the tables for a new project from within the TransactionVision application of Business Availability Center. Instead, the tables will need to be created manually before the New Project Wizard is run.

Part VI

Appendixes

A

Utilities Reference

This chapter includes:

CreateSqlScript

Location
TVISION_HOME/bin/CreateSqlScript.[sh|bat]

Purpose
Allow the user to create and optionally execute a SQL script to create, drop, import or export TransactionVision system tables or project tables.

Note: The TransactionVision Analyzer and UI/Job Server need to be stopped before performing any database imports or exports. This is to avoid causing the database import/exports to fail because of database locks held by the Analyzer or UI/Job Server. If a schema is dropped, make sure that there are no active projects or users logged in that are using that schema

---

Syntax

```
CreateSqlScript
  {-create(-c) | -drop(-d) | -import(-i) | -export(-ex) |   {-system(-sys) |
-schema(-s)SCHEMA | -table(-t) TABLE SCHEMA}
  [[-noscript(-n)] -execute(-e)]
                [-noprompt(-np)] [-noinsert(-ni)] (
                [-tablespace(-ts) TABLESPACE] [-dbMove(-m)]
                [-fileType(-f) IXF|DEL] [-lobPath(-lp) PATH]
                [-noLob(-nl)] [-dbproperties(-db) FILE]
```

Options

Option	Description
-drop (-d)	Drop tables
-create (-c)	Create tables
-execute (-e)	Execute script
-noscript (-n)	No script generation
-system (-sys)	Create/drop system tables (schema TVISION)
-schema (-s) SCHEMA	Create/drop project tables in schema SCHEMA
-table (-t) TABLE SCHEMA	Create/drop table TABLE in schema SCHEMA
-tablespace (-ts) TBSPC	Use tablespace TBSPC
-dbproperties (-db) FILE	Use Database.properties file FILE
-import (-i)	Generates a database import script. To run database scripts, use the command db2 -n -t -f <sql script filename>. For a DB2 database, you can combine this option with the -fileType and -lobPath options to customize the data format and the location of LOB. You may NOT combine this option with the -noscript or -execute options.

Option	Description
-export (-ex)	Generates a database export script. To run database scripts, use the command db2 -n -t -f <sql script filename>. For a DB2 database, you can combine this option with the -fileType and -lobPath options to customize the data format and the location of LOB. You may NOT combine this option with the -noscript or -execute options.
-resetseq (-r)	Resets the sequence start number to match to match the imported data.
-dbMove (-m)	Display the corresponding db2move command for importing/exporting a schema instead of generating an IMPORT/EXPORT script (only in conjunction with -import/-export).
-noprompt (-np)	Do not prompt for confirmation when dropping database tables.
-noinsert (-ni)	Do not insert any initial table rows defined in the XDM file when creating database tables.
-fileType (-f) IXF|DEL	Specify the DB2 data output file format used for importing/exporting data. The default type is IXF. For the IXF file type, the import/export script uses the LOBFILE option for rows that contain greater than 32K data. For the DEL type, the import/export script exports LOBFILEs into a single file (requires FixPack 8).
-lobPath (-lp) PATH	Specifies the directory for DB2 LOBFILES. The default value is the current directory.
-noLob (-nl)	Do not generate DB2 export/import SQL with LOBINFILE option. This option will truncate LOB data to the first 32K bytes.

Examples

1 Create system tables with schema as TVISION and execute the procedure without generating SQL script:

> CreateSqlScript –e –n –c –sys

2 Generate SQL script for creating project tables with schema as PROJECT without executing the procedure:

> CreateSqlScript –c –s PROJECT

3 Drop table EVENT in schema PROJECT and execute the procedure without generating SQL script:

> CreateSqlScript –e –n –d –t EVENT PROJECT

DB2RunStats

Location

TVISION_HOME/bin/DB2RunStats.[sh|bat]

Description

Updates statistics about the physical characteristics of a table and the associated indexes. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

This command is called when a table has had many updates, such as when data is continuously collected into DB2 by the TransactionVision Analyzer. It could result in large performance gains in queries made by TransactionVision views and reports, as well as queries made internally by the TransactionVision Analyzer to correlate events.

➤ This script can be set up to run as a scheduled batch job using either the UNIX **cron** facility or the Windows scheduler.

➤ While this script is running, TransactionVision Analyzer processing slows down.

➤ This script typically should be run daily, though the frequency of execution could be higher for higher message rates.

➤ This script needs to be customized based on your system to set the correct DB2 installation location and the correct TVISION_HOME location

➤ If the user has additional tables defined for the project schema, new RUNSTATS statements must be added to cover the additional tables.

Syntax

DB2RunStats *username passwd database-name schema-name* [-v7]

Options

Option	Description
-username	The user account that has privilege to execute RUNSTATS and make database connections
-passwd	The password associated with user_name
-database-name	The name of the database to connect to
-schema-name	The name of the schema that the project reads and writes data to
-v7	Use in a DB2 7.x environment. The default operation is for DB2 8.1.

DB2Test

Location

com.bristol.tvision.admin.DB2Test

Description

Measures DB2 database INSERT performance.

The utility inserts sample event data into the RAW_EVENT table of the specified schema. Before running the test, create a new project schema with CreateSqlScript (which can be deleted after running the test). See the *TransactionVision Planning Guide* for details on how to set up the required test environment

Syntax

```
java com.bristol.tvision.admin.DB2Test databaseName user passwd schema
eventCount eventSize threadCount {-commit n}{-jdbcBatch}
```

Options

Option	Description
-databaseName	Name of the DB2 Database that contains the schema to be used for the test
-user	DB2 user name
-passwd	DB2 password
-schema	TransactionVision schema in which sample event data will be saved
-eventCount	Number of events to generate
-threadCount	Number of threads to use to generate events
-commit n	Executes every n insert statements as part of a single batch operation. Default is to commit each insert individually.
-jdbcBatch	Use JDBC standard batching

MigrateDB

Location

TVISION_HOME/bin/MigrateDB.[sh|bat]

Description

Migrates project database files from an older version of TransactionVision to the current version. This script has to be run after an upgrade installation process. It must be run in a configured TransactionVision environment; the **Database.properties** must be set correctly for communication with the database.

Syntax

MigrateDB

nanny

Location

TVISION_HOME/bin/nanny.[sh|bat]

Description

Manages the services controlled by the Nanny Manager. TransactionVision components such as SonicMQ Domain Manager, SonicMQ Broker, the Analyzer and the UI/Job Server are services which are managed (started/stopped/monitored) by the Nanny Manager.

The Nanny Manager is a service (named HP Business Availability Center) on Windows and a process on UNIX (nannyManager). This nanny utility can be used to find out which components are being managed by the NannyManager.

Syntax

```
nanny
-s <hostname> -p <port> -disableService <servicename> -enableService
<servicename> -getServiceInfo <servicename> -isServiceRunning <servicename>
-listAllDisabledServicesNames -listAllServicesNames -listDeadServicesNames
-listLiveServicesNames -listServiceInfo -listStartingServicesNames -restartService
-<servicename> -retrieveNannyManagerHTMLAdapterPort -showStackTrace
<servicename> -showStackTrace <servicename> -startAllServices -startService
<servicename> -stopAllServices -stopService <servicename>
```

Options

Option	Description
-s <hostname>	Hostname of the host where the Analyzer is installed. It defaults to localhost.
-p <port>	Port of the nanny server. It defaults to 11020.
-disableService <servicename>	Disables the service specified by <servicename>.
-enableService <servicename>	Enables the service specified by <servicename>.
-getServiceInfo <servicename>	Displays service information about <servicename>.
-isServiceRunning <servicename>	Checks if the service specified by <servicename> is running
-listAllDisabledServicesNames	Lists all disabled service names
-listAllServicesNames	Lists all service names
-listDeadServicesNames	Lists all services which are not running.
-listLiveServicesNames	Lists services which are running
-listServiceInfo	Lists attributes for all services.
-listStartingServicesNames	Lists all services that are in starting state.
-restartService <servicename>	Stops and starts the service specified by <servicename>.
-retrieveNannyManagerHTMLAdapterPort	Displays the nanny jmx port.

Option	Description
-showStackTrace <servicename>	Displays stacktrace for the service specified by <servicename>.
-startAllServices	Starts all services.
-startService <servicename>	Starts a particular service specified in the argument.
-stopAllServices	Stops all services
-stopService <servicename>	Stops a particular service specified in the argument.

Examples

To list all services being managed by the Nanny Manager, enter:

nanny.bat listAllServiceNames

The command generates the following output:

Executing: listAllServicesNames Service: on (localhost,11020)
Result = [tv_as, tv_message_broker, tv_domain_manager, tv_analyzer]

The **nanny** utility can enable or disable a component. Once a component is disabled, it is not managed by the NannyManager until it is enabled.

To disable the Analyzer, run the following command:

nanny.bat disableService tv_analyzer

To enable the Analyzer, run the following command:

nanny.bat enableService tv_analyzer

The **nanny** utility can also be used to start a component after it has been enabled.

To start the Analyzer component, run the following command:

nanny.bat startService tv_analyzer.

OracleRunStats

Location
TVISION_HOME/bin/OracleRunStats.[sh|bat]

Description

Note: Oracle 10g has a built-in job scheduler that automatically collects statistics. You do not need to run this script for version 10g.

Updates statistics about the physical characteristics of a table and the associated indexes. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

This command is called when a table has had many updates, such as when data is continuously collected into Oracle by the TransactionVision Analyzer. It could result in large performance gains in queries made by TransactionVision views and reports, as well as queries made internally by the TransactionVision Analyzer to correlate events.

This script can be set up to run as a scheduled batch job using either the UNIX **cron** facility or the Windows scheduler.

Note: While this script is running, TransactionVision Analyzer processing slows down.

This script typically should be run daily, though the frequency of execution could be higher for higher message rates.

Run this script from a user account that has privileges to perform database operations. This script must be run under an Oracle user account.

Note: This script needs to be customized based on your system to invoke the correct environment initialization script like .profile or .bashrc, and to set the correct Oracle installation location and the correct TVISION_HOME location. Additionally, the file OracleRunStats.sql must be in the current directory when running this script.

Syntax

```
OracleRunStats user_name passwd database_name schema_name
```

Options

Option	Description
-user_name	The Oracle user account to run the script under
-passwd	The password associated with user_name
-database_name	The name of the database to connect to
-schema_name	The name of the schema that the project reads and writers data to

OracleTest

Location
com.bristol.tvision.admin.OracleTest

Description
Measures Oracle database INSERT performance.

The utility inserts sample event data into the RAW_EVENT table of the specified schema. Before running the test, create a new project schema with CreateSqlScript (which can be deleted after running the test). See the *TransactionVision Planning Guide* for details on how to set up the required test environment

Syntax

```
java com.bristol.tvision.admin.OracleTest databaseName host port user passwd
schema eventCount eventSize threadCount [-VARCHAR | -BLOB |
-LONGRAW] {-commit n} {-jdbcBatch} {-OracleBatch} {-thin} {-parallel} {-url URL}
```

Options

Option	Description
-databaseName	Name of the Oracle database that contains the schema to be used for the test.
-host	Name of host system on which the Oracle server exists
-user	Oracle user name
-passwd	Oracle password
-schema	TransactionVision schema in which sample event data will be saved
-eventCount	Number of events to generate
-eventSize	Size of event user data buffer (default is 1024 bytes)
-threadCount	Number of threads to use to generate events
-VARCHAR	Use this option if the RAW_EVENT table has been created with a VARCHAR column definition.
-BLOB	Use this option if the RAW_EVENT table has been created with a BLOB column definition.
-LONGRAW	Use this option if the RAW_EVENT table has been created with a LONGRAW column definition.
-commit n	Executes every *n* insert statements as part of a single batch operation. Default is to commit each insert individually.
-jdbcBatch	Use JDBC standard batching.
-oracleBatch	Use Oracle update batching
-thin	User thin client driver. Default is to use oci client driver.

Option	Description
-parallel	Use the Oracle INSERT PARALLEL option instead of the standard INSERT INTO.
-url URL	By default, OracleTest will use an appropriate JDBC URL for thin or oci client drivers. However the default may be overwritten by specifying the JDBC URL here.

PassGen

Location
TVISON_HOME/bin/PassGen

Description
A password can be obfuscated by using the PassGen utility.

Syntax

```
PassGen /system <password>
```

Options

Option	Description
-password	A string of alpha-numeric characters with a maximum length of 128 characters.

Example
cd <installdir>/bin
$ PassGen /system TheLazyFoxJumpedHigh

password:
OBF:3q6r3xxz3y3r3xjs3wx03yc63n0r3lbr3vc03wd745893wre44u0413j3kn93zw
y40vi432i44fr3m453m894493439040pc40303kjd419r44na3wx0451h3wir3v6m3
lfr3mwj3yi03wre3xpi3xxz3y3r3q23

rebind_sensor

Location
TVISON_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

rebind_tux_sensor

Location

TVISION_HOME/bin/rebind_tux_sensor.sh

Description

This script rebinds the TransactionVision BEA Tuxedo Sensor on the AIX, Solaris and HP-UX platforms.

Syntax

rebind_tux_sensor.sh [-v|-s|-h]

Options

Option	Description
-v	Writes errors to the console. The default behavior is to write errors to TVISION_HOME/logs/tuxsensorbind.log
-s	Uses silent mode, which does not prompt before executing
-h	Displays usage message

Usage Notes

The BEA Tuxedo Sensor library must be linked to the BEA Tuxedo library using its full path. If this is not done, a BEA Tuxedo application monitored by the TransactionVision Sensor may fail with errors indicated unresolved library dependencies or unresolved symbols.

Hence, the rebind_tux_sensor.sh script needs to be run after TransactionVision Sensor installation and whenever BEA Tuxedo is upgraded or its installation location is changed.

The TVISION_HOME and TUXDIR environment variables must be set before running this script. TUXDIR should be set to the full absolute path of the BEA Tuxedo Sensor installation directory. The script will modify the TransactionVision Sensor libraries in TVISION_HOME/lib or TVISION_HOME/lib64 as necessary.

runSupportSnapshot

Location

<Diagnostics_probe_install_dir>/contrib/JASMutilities/snapins/
runSupportSnapshot.[sh|cmd]

Note: The <TransactionVision_sensor_install_dir> is in
<Diagnostics_probe_install_dir>.

Description

The **runSupportSnapshot** utility creates a .zip file containing the entire set
of files relevant to troubleshooting one or more instances of the Java agent
in a Diagnostics or TransactionVision deployment environment. This .zip
file is intended for Customer Support.

The .zip file contains the following:

➤ Files from the **<Diagnostics_probe_install_dir>\etc** directory

➤ Files from the **<Diagnostics_probe_install_dir>\log** directory

➤ Files from the **<TransactionVision_sensor_install_dir>\config** directory

➤ Files from the **<TransactionVision_sensor_install_dir>\logs** directory

➤ Property Scanner report, which compares a modified agent directory with
the release version, and reports differences between the property files, points
files, and for TransactionVision Sensors only, XML files.

➤ Probe or Sensor instance information, including property settings. For
agents running in 1.5 JVMs, environment variables, stack dumps, and class
loader information is also included.

Note: If a Diagnostics probe is configured so support SSL, this utility cannot
collect the instance information.

Basic Syntax

This utility uses defaults that apply to most agent environments. Therefore
the typical syntax is simply:

```
runSupportSnapshot.[sh|cmd] -console
```

Full Syntax

This utility also allows you specify options to override the defaults, allowing it to work in non-typical agent environments. This optional syntax is:

```
runSupportSnapshot.[sh|cmd] -console [ -Zipfile ] [ -AddFiles ]
[ -JAUser ] [ -JAPass ] [ -JAPort ] [ -JACount ]
[ -FTPSite ] [ -User ] [ -Pass ] [ -RemoteFile ]
[ -Source1] [ -Source2 ] [ -DiffOnly ] [ -SubDirs ] [ -Sort ]
```

Optional Syntax Options

Option	Description
-ZipFile	The name of the .zip file that is to be created. Dfault: **HPCustomerSupport.zip.**
-AddFiles	Any additional directories to include in the .zip. To include all subdirectories of a directory, append an "+" to the directory name.
-JAUser	The user name to be used to access the host on which the Agent is running. Default: admin
-JAPass	The corresponding password. Default: admin
-JAPort	The port on which to start polling for Agent instances. Default: 35000.
-JACount	The maximum number of Java Agent instances on this host for which data will be included in the .zip file. Default: 20.
-FTPSite	The name of the FTP site.
-User	The user name to be used to access the FTP site.
-Pass	The corresponding password.
-Remote File	The remote file name. Prefix the filename with directory path if desired. For example, **customer/diag/AgentSnapshot.zip**.
-Source1	The **etc** directory of the current Agent installation. Default: <Java_probe_install_dir>**etc.**
-Source2	An archived copy of the out-of-the box etc directory in the directory. Default: An internal directory to the Agent installation.
-DiffOnly	Only display properties with differences (yes or no). Default: Yes
-SubDirs	Specify whether subdirectories are also be compared (yes or no).Default: No.
-Sort	Specify whether the output is sorted by property name (yes or no). Default: No.

Example

```
> cd C:\MercuryDiagnostics\JavaAgent\DiagnosticsAgent\contrib\JASMUtilties
\Snapins
> java  -Dcom.hp.javaagent.diagnostics.home="..\..\.." -jar "..\..\..\lib\setupModule.jar"
-launchClass com.mercury.opal.javaprobe.setupModule.SetupModule  -launchMethod
launchSetupModule -importJarList
probe.jar,org.mortbay.jetty-jdk1.2.jar,javax.servlet.jar,mail.jar,activation.jar
-importJarsFrom "..\..\..\lib,lib" -customerSnapshot -console
INFORMATION-> [Looking for Java Agent Instances]: Looking for Java Agent
Instances...please wait...
INFORMATION-> [Java Agent found at http://localhost:35000/inst/
customerSupportSnapshot]: Java Agent Discovery
INFORMATION-> [Creating Probe Information file]: Java Agent Information
INFORMATION-> [Creating Configuration Information file]: Java Agent Configuration
Information
INFORMATION-> [Zip File Created]: Zip file
C:\HPCode\diag_head\javaprobe\build\contrib\JASMUtilities\Snapins\HPCustomerSup
portFile.zip Created with length 159215
```

If desired, review the .zip file for any sensitive data before transmitting it to
Customer Support.

ServicesManager

Location

TVISION_HOME/bin/ServicesManager.[sh|bat]

Purpose

Manage the TransactionVision Analyzer service. The Analyzer uses an
embedded RMI registry so that Analyzers may be controlled by the
TransactionVision web user interface running on remote hosts.

If the Analyzer is stopped or told to exit using the ServicesManager utility's
-keepcollect option, the only way to tell the Sensors to completely stop is to
start the Analyzer and project and then stop the project normally.
Otherwise, the Sensors will continue to produce events which will collect on
the event queue until the last configuration message sent by the Analyzer
has expired. See Configuration Message Expiry in *Using TransactionVision*.

Syntax

```
ServicesManager
 {(-start [-project (-proj) PROJECTNAME]) |
  (-stop [-quiesce][-project (-proj) PROJECTNAME]
[-keepcollect]) |
  (-exit [-quiesce][-keepcollect]) |
  (-status [-project (-proj) PROJECTNAME]) |
  (-killserver) |
  (-reconfig {classification | logging | analyzer}) |
  (-versioninfo)
  ([-host HOST][-rmiregp PORTNUMBER][-debug])
```

Options

Option	Description
-start	Starts the Analyzer process if it is not already running
-stop	The Analyzer stops collecting event data
-exit	The Analyzer stops collecting event data, then the process exits
-quiesce	The default behavior of the Analyzer on a stop or exit is to immediately close down collection. If this flag is set, the Analyzer clears out any pending events in the event queue before stopping or exiting. Note that if there is a large event backlog, the Analyzer may take some time to stop or exit.
-keepcollect	Do not send stop message to Sensor. Used in combination with the –stop or –exit option so that the Sensor keeps collecting.
-status	Reports the current Analyzer status

Option	Description
-reconfig	Reloads Analyzer configuration settings (classification/ logging/all settings) without stopping event collection. ➤ **logging** – Reloads the log4j XML configuration files in the directory <TVISION_HOME>/config/logging. ➤ **analyzer** – Re-initializes the Analyzer by reloading most Analyzer settings in the Analyzer.properties file, all beans described in the Beans.xml file, re-initializing logging and all XML based rule files. Settings in the"Collection Properties" section of the Analyzer.properties file, except the write_to_buffer_table property, are not reloaded by the Analyzer with the -reconfig option.
-versioninfo	Returns Analyzer version information
-killserver	Shuts down the Analyzer immediately. This flag is not recommended; -exit is the preferred method for shutting down the Analyzer cleanly.
-project (-proj) PROJECTNAME	Name of the project for the specified command. If the project name contains a space, enclose the project name in double quotation marks (for example, -proj "Project Name"). The project must have a communication link in order for the Analyzer to process events. Only the -start, -stop, and -status commands be performed on a single project. This option cannot be used in combination with the -host or -rmiregp options. When the -project option is used, the Analyzer host and port is looked up from the database.
-host HOST	Name of the host where the Analyzer runs. Can be either name or IP address. Local host is used if not specified.
-rmiregp PORTNUMBER	Port number on which the Analyzer listens for RMI connections. The default value is the value specified by the analyzer_port property in the Analyzer.properties file. This option only takes effect when communicating with an Analyzer that is already running; the Analyzer always uses the value specified in Analyzer.properties when starting up.
-debug	Start the Analyzer process in debug mode.

Examples

➤ Start project PROJECT:

ServicesManager –start –proj PROJECT

➤ Stop Analyzer on host HOST:

ServicesManager –stop –host HOST

SetupModule

Location

<java_agent_install_dir>\DiagnosticsAgent\lib

where <java_agent_install_dir> refers to the path of your Java Agent installation directory. The default path is C:\MercuryDiagnostics\JavaAgent on Windows and \opt\MercuryDiagnostics\JavaAgent on UNIX.

Description

This script starts the Java Agent Setup Module.

Syntax

```
SetupModule -recordFile name.rec -installFile name.rec -console
```

Options

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

SQLServerTest

Location

com.bristol.tvision.admin.SQLServerTest

Description

Measures SQL Server database INSERT performance.

The utility inserts sample event data into the RAW_EVENT table of the specified schema. Before running the test, create a new project schema with CreateSqlScript (which can be deleted after running the test).

Syntax

```
java com.bristol.tvision.admin.SQLServerTest databaseName user passwd schema
eventCount eventSize threadCount {-commit n}{-jdbcBatch}
```

Options

Option	Description
-databaseName	Name of the SQL Server Database that contains the schema to be used for the test
-user	SQL Server user name
-passwd	SQL Server password
-schema	TransactionVision schema in which sample event data will be saved
-eventCount	Number of events to generate
-threadCount	Number of threads to use to generate events
-commit n	Executes every n insert statements as part of a single batch operation. Default is to commit each insert individually.
-jdbcBatch	Use JDBC standard batching.

TVisionSetupInfo

Location

TVISION_HOME/bin/TVisionSetupInfo.[sh|bat]

Purpose

Collects information from the administrator about external tools used by TransactionVision so that CLASSPATH and library path can be setup correctly before running TransactionVision. This utility performs the following:

➤ Saves installation path for software tools in TVISION_HOME/config/setup/ DefaultInstallPath.xml.

➤ Modifies TVISION_HOME/config/datamgr/Database.properties based on user input.

➤ Modifies TVISION_HOME/config/ldap/Ldap.properties based on user input.

➤ Generates TVISION_HOME/bin/SetupEnv.[sh|bat] which sets the minimum set of environment variables (JAVA_HOME, CLASSPATH, shared library path for different platforms, etc.) required by TVision.

Syntax

TVisionSetupInfo.[sh|bat] - cleanweb

Options

Option	Description
-cleanweb	Remove the TransactionVision Web application.

Notes

TVISION_HOME/config/setup/Setup.properties is used to specify the XML file that stores the default software installation path and setup logging file.

User is free to modify any modified/generated files later on.

Since this utility modifies TransactionVision's configuration files, the user needs to have file modification privilege to run it (root on Unix or Administrator on Windows).

The format of TVISION_HOME/config/setup/DefaultInstallPath.xml is as follows:

```xml
<?xml version="1.0"?>
<DefaultInstallPath>
   OS name="Windows">
   <DB2>C:\Program Files\IBM\sqllib</DB2>
   <WebSphereMQ>C:\Program Files\IBM\WebSphere MQ</WebSphereMQ>
   <WebSphereMQJava>C:\Program Files\IBM\WebSphere MQ\Java</
WebSphereMQJava>
  <WebSphereAppServer>C:\Program Files\Websphere\ AppServer</
WebSphereAppServer>
  <WebLogic>C:\bea\weblogic81</WebLogic>
  </OS>
  <OS name="SunOS">
   <DB2>/opt/IBMdb2/V7.1</DB2>
   <Oracle></Oracle>
   <WebSphereMQ>/opt/mqm</WebSphereMQ>
   <WebSphereMQJava>/opt/mqm/java</WebSphereMQJava>
   <WebSphereAppServer>/opt/WebSphere/AppServer </WebSphereAppServer>
<WebLogic>/usr/local/bea/weblogic81</WebLogic>
  </OS>
  <OS name="AIX">
   <DB2>/usr/lpp/db2_07_01</DB2>
   <Oracle></Oracle>
   <WebSphereMQ>/usr/lpp/mqm</WebSphereMQ>
   <WebSphereMQJava>/usr/lpp/mqm/java</WebSphereMQJava>
   <WebSphereAppServer>/usr/WebSphere/AppServer </WebSphereAppServer>
<WebLogic></WebLogic>
  </OS>
  <OS name="Linux">
   <DB2>/usr/IBMdb2/V7.1</DB2>
   <Oracle></Oracle>
   <WebSphereMQ>/opt/mqm</WebSphereMQ>
   <WebSphereMQJava>/opt/mqm/java</WebSphereMQJava>
   <WebSphereAppServer>/opt/WebSphere/AppServer </WebSphereAppServer>
   <WebLogic></WebLogic>
  </OS>
</DefaultInstallPath>
```

353

B

Configuration Files

The TransactionVision setup utilities save Analyzer configuration information in the following configuration files. You can also modify these files directly if you need to make any changes to your configuration.

This chapter includes:

Note: If you modify property files, you must restart the associated application for you changes to take effect. For information about restarting the , see "Managing the Analyzer" on page 82.

Analyzer.properties

The **<TVISION_HOME>/config/services/Analyzer.properties** file provides general, collection, and analysis configuration information for the TransactionVision Analyzer.

General Properties

➤ **rmi_client_timeout.** This entry specifies the timeout for the RMI client in minutes. The default value is 2.

➤ **logging_xml.** Specifies the name of the logging configuration file used by the Analyzer. The default value is Analyzer.Logging.xml.

➤ **trace.** This property specifies whether Analyzer trace logging is on or off. The default value is ON.

➤ **debug.** This property specifies whether Analyzer debug logging is on or off. The default value is OFF. Enable this setting only if advised to do so by support.

➤ **service_jvm_flags.** Specifies any additional JFM flags you want the Analyzer to run with.

➤ **service_additional_classpath.** Specifies the classpaths for any custom beans to be used with your Analyzer.

➤ **jvm_dll**
service_classpath
service_libpath. On Windows, these properties are automatically generated by TVisionSetupInfo. It should not be necessary to change them. If you do change them, note that they will be reset if TVisionSetupInfo is run again. These properties have no effect on the TransactionVision environment other than the Analyzer running as a Windows service.

➤ **analyzer_port.** This entry specifies the default port number that the Analyzer runs on. The default value is 21100. The Analyzer always uses this port number when it starts. The ServicesManager.[bat|sh] -rmiregp command can be used to set the port number for an Analyzer that is already running.

➤ **rmi_server_port.** This entry specifies the default port number that the RMI server object is listening to.

The default value is 0, which means that the java runtime will use an arbitrary, random port whenever the Analyzer starts. If you are running the Analyzer behind a firewall and require a fixed port then you can set this property to a specific port value.

➤ **time_server.** Set to on to run the time server within the Analyzer. The time server is required for TIBCO EMS and generic JMS communication link types to calculate the time skew information. To run the time server on a different host, use the TimeServer.[sh|bat] utility.

➤ **time_server_port.** Specifies the listening port for the time server to use, if time_server is set to on. The default port is 9037.

Collection Properties

The following properties are used for the event collection operations of the Analyzer:

➤ **batch_commit_count.** Specifies the number of events to batch before a commit is issued. The default value is 50.

➤ **commit_time_threshold.** Specifies the amount of time, in seconds, after which a commit is forced. The default value is 1.

➤ **commit_byte_threshold.** Specifies the number of bytes after which a commit is forced. The default value is 1,000,000.

➤ **write_to_buffer_table.** Specifies whether to write a copy of raw events from the queue into the RAW_EVENT table, in addition to normal processing. The default value is False. The setting is global to all projects, opposed to the corresponding setting on the Communication Link page in the UI which allows to enable raw event writing for a specific Communication Link. This property should never be set to True if read_from_buffer_table is also set to True. Use this property to test custom beans, or if advised by support to troubleshoot issues. It should always be set to False in a production environment.

➤ **read_from_buffer_table.** Specifies whether to pull any events from the RAW_EVENT table in addition to reading from the queue. The default value is False. This property should never be set to True if write_to_buffer_table is also set to True. Use this property to test custom beans, or if advised by support to troubleshoot issues. It should always be set to False in a production environment.

➤ **keep_events.** Specifies whether the Analyzer should delete raw events after they have been successfully processed. Set it to true to keep the raw events in the RAW_EVENT table. Their event_status column will be set to PROCESSED so that they are not processed again. If you want to process the events again (for example, with a custom Java bean for a different type of analysis), you must first set the event_status to NEW. Set it to false (the default value) for the Analyzer to delete raw events from the RAW_EVENT table after it processes them.

➤ **write_to_jar.** Similar to **write_to_buffer_table**, but the Analyzer stores the raw events in a JAR file instead of a table in the database. The location of the JAR file is $TVISION_HOME/logs, and the file name will be **SCHEMA_raw_events.jar**. For each project schema a separate JAR file is created. You must stop the project in order to close the JAR file correctly.

➤ **read_from_jar.** Similar to **read_from_buffer_table**, but the Analyzer reads the raw events from a JAR file instead of a table in the database. For a project with database schema SCHEMA the Analyzer will look for a file **SCHEMA_raw_events.jar** in the <TVISION_HOME>/logs directory. Note that you can copy the jar file while the Analyzer is running, it will periodically scan the directory for the corresponding files. Once the Analyzer starts processing the events in the JAR file it is renamed to **SCHEMA_raw_events.jar.processed.TIMESTAMP**. Note that you should never set this setting to True if write_to_jar is also set to True.

➤ f**ail_safe_collection_shutdown.** If true, causes the Analyzer to stop sending configuration messages if a serious failure occurs in event collection and processing. The default value is true.

➤ **jdbc_batching.** The default jdbc_batching value (on) causes TransactionVision to execute database statements in batch mode.

➤ **jdbc_batch_count.** Specifies The number of SQL operations to batch in JDBC batching mode. Note that this number should be equal to or a multiple of the batch_commit_count property. The default value is 50.

Analysis Properties

The following properties are used for the event analysis operations of the Analyzer:

➤ **save_event_document.** Specifies whether to save event documents in the database. The default is on. If this property is set to off, event detail is not available. Furthermore, the Analyzer will not be able to recover asynchronously flushed transaction and static topology data. This means that in the event of a crash or other abnormal program termination, transaction data and statistics data for the static topology view may no longer be accurate.

➤ **eventmatching_interval.** Specifies the time in seconds to wait between every invocation of event matching. The default is value is 600.

➤ **partial_event_lifetime.** The maximum time in minutes a partial event entry may exist in the partial_event table. When this time limit is reached, the partial event will be flushed. The default value is 10 minutes.

➤ **latency_resolution.** The resolution used to calculate latency between events. Possible values are:

Value	Description
1	Seconds
10	1/10 seconds
100	1/100 seconds (default)
1000	Milliseconds

➤ **generate_api_only_txns.** By default the Analyzer will not perform any transaction analysis on events that have been collected with a DataCollection filter "API only". Set this property to true to change this behavior.

➤ **correlation_limit.** This setting limits the number of event relationships created by the Analyzer for a specific correlation key and can be used as a safeguard against flawed correlation rule design in custom correlation, leading to performance issues. The default value is -1 (no limit).

➤ **local_txn_limit.** This setting limits the number of local transactions assigned by the Analyzer to one business transaction, and can be used as a safeguard against flawed local transaction rule design in custom local transaction analysis, leading to performance issues. The default value is -1 (no limit).

➤ **separate_child_thread_txns.** By default, if a servlet spins off a thread to make some JMS calls, the servlet passes tracking information to the child thread. The result is that both the servlet and JMS events belong to the same business transaction. However, there may be some cases in which you want to separate these events into different transactions. For example, a servlet may spin off a long-running thread that you do not want to be part of the same transaction as the servlet. If you do not want threads spun off by a servlet to be included in the same business transaction as the servlet, list the servlet program name as the value for this property. Separate multiple program names with a comma, as in the following example:

separate_child_thread_txns=program1, program2

➤ **enable_dbcaching.** Enables standard mode in addition to or instead of failure mode. If all of your projects only use Failure Mode, the Standard Mode can be disabled to save some system resources.

➤ **enable_failure_mode.** Enables failure mode so that the Analyzer will only store event data of failed business transactions (or transactions violating SLA). For successful business transactions, only the corresponding business transaction rows will be saved. Since this mode relies heavily on a meaningful value of the 'result' attribute of the business transaction, it requires a transaction classification with suitable rules for the 'result' column to be in place. The failure mode can be enabled on a per-project basis, allowing it to run certain projects in failure mode while running others in the conventional Standard Mode. The default value is off. For more information about failure mode, see "Failure Mode" on page 74 of *Using TransactionVision.*

➤ **dbcache_thread_count.** Specifies the number of threads to use for writing the cached analysis data to the database. The best value is dependent on the hardware the Analyzer is running on and can only be determined by performance measuring with different values, but a good starting point would be to set this number to half the number of collection threads used for a particular schema. The default value is 2.

➤ **failure_mode_thread_count.** Defines the number of flushing threads to use for Failure Mode. This number should be set high enough so that the cache data can be flushed fast enough, but low enough to not waste too many system resources. The default value is 2.

➤ **failure_mode_process_delay.** Defines the amount of time, in milli-seconds, which gets added to the SLA value for determining the age out timeout and is a crucial value for the Failure Mode. If this value is set too low, a lot of successful transaction data will unnecessarily age out and be written to the database, which will result in poorer performance, and wasted database disk space. If this value is set too high, too much data will be held in the memory caches. If the cache sizes are not large enough for the system load, cache overflows will occur frequently and lead to very poor performance. The default value is 5 seconds.

➤ **failure_mode_discard_overflow.** In case of a cache overflow all the data in the caches will normally be written to the database. If the overflow was caused, for whatever reason, by a longer lasting system slowdown, the constant flushing of the cache could further worsen the situation and slow down the Analyzer even more. If the main concern is to avoid any chance of filling up the event queue in such situations, you can set the above parameter to true and force the discarding of all cache data in case of an overflow. The default value is false.

CacheSize.properties

The **<TVISION_HOME>/config/services/CacheSize.propertie**s file specifies the size of the caches used in the event analysis. A bigger cache size minimizes database access and increases performance, but also increases the amount of memory used by TransactionVision. It sets the following properties:

➤ **system_model_objects.** Specifies the number of system model object other than PII cached.

➤ **pii_objects.** Specifies the number of PII system model objects cached.

➤ **event_based.** Specifies the number of events cached.

➤ **transaction_based.** Specifies the number of transactions cached.

Database.properties

The **<TVISION_HOME>/config/datamgr/Database.properties** file specifies the database the Analyzer and the TransactionVision UI/Job Server read from and write to.

Required Entries

The following mandatory entries are required:

➤ **jdbc_driver=COM.ibm.db2.jdbc.app.DB2Driver.** This entry is the class name of the JDBC driver used. The default for the IBM DB2 JDBC driver is as above. For Oracle, the default is oracle.jdbc.driver.OracleDriver. For SQL Server, the default is com.microsoft.jdbc.sqlserver.SQLServerDriver.

➤ **database_connection_name.** For DB2, this entry is the name of the database connection to be used (typically the database alias). For Oracle, this entry is the database name for an oci client connection or the SID (system identifier) for a thin client connection. For SQL Server, this entry is not used.

➤ **database_name.** For DB2, this entry is the name of the database on the server to connect to. This name may be different from the "database_connection_name" if a client database connection is used. For Oracle, this entry is the SID. For SQL Server, this is the name of the database.

➤ **database_host.** This entry is the host the database server is running on.

➤ **database_port.** For Oracle, this entry specifies the port number for the Oracle listener on the target host. The default value is 1521. This attribute is only used if you are using the Oracle thin client. For more information on using Oracle with TransactionVision, see Chapter 6, "Configuring Databases." For SQL Server, the entry specifies the port to connect to on the server, the default value is 1433.21."

➤ **oracle_client_type.** Specifies whether to use the Oracle thin or client JDBC driver. Set it to thin (recommended) or oci for Oracle 9 or oci8 for Oracle 8.1.7. The default is thin. This attribute is only used if you are using an Oracle database. For more information on using Oracle with TransactionVision, see Chapter 6, "Configuring Databases."

➤ **oracle_user_password.** This entry specifies the user password that will be set for the Oracle user which gets created for a new project schema. The default value is ABC99DEF. You can change this value if you need to conform to specific password policies.

➤ **db2_instance_env.** This entry specifies the value of the DB2 environment variable DB2INSTANCE.

Optional Entries that Override Automatic Detection

The following optional entries may be setup in this file. These entries override automatic detection.

➤ **connection_type.** This entry defines how to obtain JDBC connections. Use one of the following values:

  ➤ JDBC: Uses connections obtained from the JDBC driver, with no driver connection pooling (default).

  ➤ DB2DataSource: Uses DB2 connection pooling.

  ➤ OracleDataSource: Uses Oracle connection pooling.

  ➤ JNDI: Uses the data source registered in JNDI. To use a JNDI connection, you must also set the jndi_url property.

➤ **database_url.** This entry defines a custom URL to use with the JDBC driver manager.

➤ **jndi_url.** This entry defines the JNDI name for the database (defined in WebSphere) when using JNDI connections.

➤ **unicode_db.** If this property is set, all character-based XDM columns with the attribute unicode=true will be generated with double the byte size to allow the specified number of characters to be stored in the database.

➤ **unicode_bytes_per_character.** If this property is set, all character-based XDM columns with the attribute unicode_true will be generated with a size using this value as a multiplier of the base value. The default value of 2 matches the behavior of setting unicode_db=true and not setting this value. Values larger than 3 may cause database creation problems. This property will not take effect unless unicode_db=true is also set.

➤ **jdbc_url.** This entry specifies a database driver to use other than the default DB2 Universal JDBC driver (type 4) or the Oracle thin client driver.

For the DB2 Universal Driver (type 2), this entry has the following format:

jdbc:db2:<database-name>

where <database-name> is the database to which the Analyzer connects to.

For the Oracle oci driver, this entry has the following format:

jdbc:oracle:oci:<user>/<password>@<database-name>

Where <database-name> is the database to which the Analyzer connects to, and <user>, <password> are the user name and passwords required for the Oracle connection.

Optional Entries with Default Values

The following optional entries may be setup in this file. If they are not specified, default values are used.

➤ **user.** This optional entry is the user name to be used while making the database connection. If this field is empty, the currently logged in user is used to make the database connection. Make sure that the user specified or currently logged has privileges for database access.

➤ **passwd.** This optional entry is the password to be used while making the database connection. If this field is empty, the currently logged in user's password is used to make the database connection.

➤ **jce_provider.** This optional entry specifies the Java Cryptographic Extension (JCE) package name to encrypt the database password. The following table shows the valid package names. If this entry is blank, TransactionVision stores the password as plain text.

Provider	Package Name
Sun	com.sun.crypto.provider.SunJCE
IBM	com.ibm.crypto.provider.IBMJCE

The TVisionSetupInfo utility automatically searches for a an installed JCE provider that supports the DES encryption algorithm. If such a provider is found, TVisionSetupInfo sets the jce_provider value to the class name of the JCE provider. If no JCE provider is found, TVisionSetupInfo displays the following message:

Java Cryptography Extension (JCE) is not present in your current JDK. Password encryption feature will be disabled and stored in plain text.

➤ **reconnect_interval.** This entry specifies the frequency in seconds that TransactionVision should try to reconnect with the database. The default value is 10.

➤ **reconnect_timeout.** This entry specifies the amount of time in seconds that TransactionVision should try to reconnect to the database (at the reconnect_interval). The default value is 600.

➤ **query_timeout.** This field specifies the timeout in seconds, for queries run in the UI. By default this setting turned off.

JobManager.properties

The <**TVISION_HOME**>/**config/jobs/JobManager.properties** file specifies configuration information for TransactionVision jobs. This file specifies the following properties:

➤ **BeansXMLFile.** Specifies the location of the configuration file listing job templates available when you create a new project.

➤ **logSize.** Specifies the number of entries to keep in the log file.

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).

Attribute	Description
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.

StatisticsCache.properties

The **<TVISION_HOME>/config/services/StatisticsCache.properties** file defines setting used in the static mode of the component topology analysis. It contains the following entries:

➤ **flush_interval.** Specifies how often the statistics cache is written out to the database. Until the data is written to the database, it won't be visible in graph. Depending on the volume of incoming requests, it might be useful to either flush more often, or less often. A longer flush time means fewer database writes, but will result in a larger cache size.

➤ **timeslice_interval.** Specifies the duration in minutes of the timeslice for which event statistics should be calculated. The concept of a timeslice is used to partition data into different pieces so that depending on what timeframe one is interested in looking at this data can be quickly retrieved. A timeslice represents a period of time during which all events belonging to the same program and MQ object are condensed into a single statistic. For example, if you had a program putting messages to a particular queue, all the statistics data relating to the program and the queue (latency times, success counts, etc.) during a particular timeslice are stored in the same database row. By changing the values of the time slice interval you can control how big and how efficient access to the statistics table is. The larger the timeslice, the more efficient the storage of results will be; the trade-off you lose is that your time slice interval is the smallest increment of time that you can view your data in. For example, if you were to set a time slice of one day, you would not be able to view these statistics on an hourly basis.

UI.properties

The **<TVISION_HOME>/config/ui/UI.properties** file specifies properties used by the TransactionVision web user interface. It contains the following entries:

➤ **ui_config_dir.** Specifies the location of the user interface configuration files.

➤ **view_config_filename.** Specifies the file defining TransactionVision views.

➤ **reports_xml_filename.** Specifies the location of the file listing available reports.

➤ **category_xml_filename. S**pecifies the location of the query configuration file.

➤ **category_xsl_filename.** Specifies the location of the query style sheet.

➤ **filter_config_xml.** Specifies the location of the data collection filter configuration file.

➤ **filter_config_xsl.** Specifies the location of the data collection filter style sheet.

➤ **detail_modifier_xsl.** Specifies the location of the Event Detail view style sheet.

➤ **logging_xml.** Specifies the location of the user interface log file.

➤ **support_url.** The URL for product support.

➤ **autologin.** Enables the "Remember Login" check box on the TransactionVision login page. Setting the check box causes the user name and password to be saved as a cookie. so that users are not prompted for a user name and password the next time they try to access TransactionVision. The password is encrypted using the specified Java JCE provider.

➤ **jceProvider.** Specifies the name of the JCE package used to encrypt the password if autologin is true. If you don't have a JCE package, you can download one from java.sun.com.

➤ **trace.** Enables trace logging for the TransactionVision web user interface. For more information, see "Using Windows and UNIX System Logs" on page 100.

➤ **logout_Redirect.** This property specifies the URL that should be displayed when a user logs out from TransactionVision, such as an SSO logout page.

➤ **dashboard_xml_filename.** The dashboard report configuration file name.

➤ **hasProxySensor.** Set to True to enable the dynamic Component Topology Analysis view to show proxy related links if you are using the Proxy Sensor.

➤ **bac_lwsso_enabled.** Set to False if LW-SSO is disabled in Business Availability Center. This will allow you to access TransactionVision without requiring LW-SSO.

For more information about LW-SSO, see the *HP Business Availability Center Hardening Guide* PDF.

C

Database Migration

This chapter includes:

Time and Space Requirements

TransactionVision version 7.50 contains major changes to the database schema that have a large impact on the time and resources required by the migration process. Most of the table data of a pre-7.50 project will need to be copied during the migration. Due to the fail-safe implementation of the migration process this requires an additional amount of free space equal to the amount of space used for the largest existing project in the database. For example, if your largest TransactionVision project is using 2 GB of space, the migration will need an additional 2 GB to finish successfully. Also, depending on the amount of data in your database the migration process can take a considerable amount of time to complete.

Disabling Unused Integration Columns

The new database schema contains some new table columns to be used for the integration of TransactionVision with other products in the Business Availability Center suite. If you are operating TransactionVision in a performance critical environment and do not intend to use any of the integrations, you can avoid creating those new columns in the migration process. Run the 'MigrateDB' script (see Appendix A, "Utilities Reference") with the following command line option:

MigrateDB.sh|.bat -disableMigration,

Migration of Customized Database Schemas

If you have added custom XDM definition files in the previous TransactionVision installation, these files will be preserved in the new configuration. But you must change the 'proginst_id' key field from INTEGER to BIGINT for any custom event tables, and the "business_trans_id" key field from INTEGER to BIGINT for any custom transaction tables. If you added or modified columns in the standard XDM files, those changes will not get preserved in the new installation. Choose to save the old configuration files at setup time and make the necessary updates after installation.

Database Migration - Technical Details

The following list contains all changes to the database schema since TransactionVision version 5.00:

TVISION System Schema

➤ Table SCHEMA_VERSION: renamed column schema -> schema_name.

➤ Table ID_TABLE: renamed column key -> key_name.

➤ Renamed table SCHEMA -> SCHEMA_TABLE.

➤ Added table CLASSIFICATION.

➤ Added table CLASSIFICATION_REL.

➤ Added table TXN_CLASS_ATTRIBUTE.

➤ Added table TXN_CLASS_ATTR_VALUE.

➤ Added table PROPERTIES.

Project Schema

➤ Table ID_TABLE: renamed column key -> key_name.

➤ Table SCRATCH: renamed column key -> key_name.

➤ Changed type of column 'proginst_id' from INTEGER to BIGINT for the following tables: PARTIAL_EVENT, EVENT, EVENT_OVERFLOW, USER_DATA, USER_DATA_OVERFLOW, RELATION_LOOKUP.

➤ Changed type of column 'proginst_id' from INTEGER to BIGINT in all tables defined via xdm files in $TVISION_HOME/config/xdm with documentType="/Event".

➤ Changed type of column 'local_trans_id' and 'business_trans_id' from INTEGER to BIGINT in table LOCAL_TRANSACTION.

➤ Changed type of column 'local_trans_id' from INTEGER to BIGINT in table TRACKING_OVERFLOW.

➤ Changed type of column 'business_trans_id' from INTEGER to BIGINT in table BUSINESS_TRANSACTION.

➤ Changed type of column 'proginst_id' and 'proginst_id2' from INTEGER to BIGINT in table EVENT_RELATION.

➤ Changed type of column 'object_id' from INTEGER to BIGINT in table SYS_MDL_OBJECT.

➤ Changed type of column 'object_id' and 'object_id2' from INTEGER to BIGINT in table SYS_MDL_OBJECT_RELATION.

➤ Table LOCAL_TRANSACTION: renamed column key -> key_name.

➤ Table BUSINESS_TRANSACTION: renamed column sequential_id -> update_id and changed column type from INTEGER to BIGINT.

➤ Table EVENT_LOOKUP: renamed column sequential_id -> seq_id and changed column type from INTEGER to BIGINT.

➤ Table TRANSACTION_STATS: renamed column begin-> begin_time and end -> end_time.

➤ Table JMS_LOOKUP: added column exception_class (INTEGER).

➤ Table BUSINESS_TRANSACTION: added column exception (INTEGER).

➤ Added the following columns for integration purposes to
SERVLET_LOOKUP: probe_id, probe_group_id, uri, bpm_profile_id,
bpm_location_id, bpm_txn_id, client_ip, session_id, url (data types can be
found in $TVISION_HOME/config/xdm/Servlet.xdm).

➤ Added the following columns for integration purposes to
BUSINESS_TRANSACTION: trans_id, is_bpievent (data types can be found in
$TVISION_HOME/config/xdm/Transaction.xdm).

➤ Changed unique index (tracking_id, tracking_seq) on table
TRACKING_OVERFLOW to non-unique.

➤ Added table JDBC_LOOKUP.

➤ Added table JDBC_STATS.

➤ Added table SYS_MDL_OBJECT_ATTR.

Optimizing TransactionVision in Non-Integration Environments

TransactionVision 7.50 introduces several new project table columns for
integrating with other products in the Business Availability Center product
family. If you are not using any of the integrations, NULL values are written
into these columns during Analyzer processing. If your database product is
updating index files even for NULL values it might be possible that this has
a small impact on the Analyzer performance. So if you are running the
TransactionVision Analyzer in a high-performance environment and do not
intend to integrate with other products, we suggest to disable the
integration columns as follows:

➤ Run **TVisionSetupInfo.bat|.sh** from $TVISION_HOME/bin with the
command line switch -integrations.

➤ Answer "**n**" to the following questions:

```
---------------------------------------------------------
 Integration Settings
---------------------------------------------------------
Retrieving current integration settings...
Notice:
  If you disable integrations, projects created afterwards may not work properly
  as the database columns won't exist. It's safer to enable them now and just
  disable them in the UI later.
IntegrationEnableBAC (y/n) [n]:
IntegrationEnableDiag (y/n) [n]:
```

This disables all integration columns in the XDM files (currently
Servlet.xdm and **Transaction.xdm**). For new project schemas, the tables will
be created without those columns. For existing project schemas, the
columns will remain in the tables, but the Analyzer will no longer access
them. You can change this setting any time by running **TVisionSetupInfo**
with the corresponding command line switch again.

D

Additional z/OS Settings

This chapter includes:

➤ RACF Authorizations on page 377

➤ Firewall Settings on page 380

➤ MIPS Required on page 380

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
SLDPCMX	Yes	No	No
SLDPCCX	Yes	No	No
SLDPDSX	Yes	No	No
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

Exit Program	Needs to Run	Needs to Access WebSphere MQ	Needs to Run as a CICS Global User Exit
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 an may also need to be RACF authorized based on the local RACF authorization scheme.

Libraries	APF Authorized	In DFHRPL
SSLDLOAD	No	Yes
SSLDAUTH	Yes	Yes

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.

E

uCMDB Discovery Agents

This chapter describes how to install, configure, and operate the uCMDB Discovery Agent on IBM z/OS.

This chapter includes:

➤ Installing and Configuring the uCMDB Discovery Agent on page 381

➤ uCMDB Discovery Agent Components and Operation on page 396

➤ uCMDB Discovery Agent Security Requirements on page 398

➤ uCMDB Command Summary on page 402

➤ uCMDB Mainframe Services Agent Console Messages on page 403

➤ uCMDB z/OS Discovery Error Messages on page 411

Important: Read both the uCMDB Discovery Agent Components and Operation on page 396 and uCMDB Discovery Agent Security Requirements on page 398 before you install and configure the Agent. Doing so will provide useful background and context.

Installing and Configuring the uCMDB Discovery Agent

The procedure to install and configure the uCMDB Discovery Agent and co-requisite products is as follows:

Task 1: Verify System Requirements

Task 2: Install the uCMDB Discovery Agent

Task 3: Install the IBM HTTP Server

Task 4: Enable the IBM HTTP Server for uCMDB Discovery

Task 5: Customize the uCMDB Discovery Service Agent

Task 1: Verify System Requirements

Verify that the host on which you are installing meets these system requirements:

➤ Operating system IBM z/OS V1.7 or above.

➤ IBM HTTP Server for z/OS, Version 5.3 (5694-A01), installed and configured as described in tasks 3 and 4.

➤ IBM SDSF for z/OS (System Display and Search Facility). This product is required to discover characteristics of IMS subsystems.

The IBM HTTP Server for z/OS Version 5.3 is hereafter referred to as the IBM HTTP Server.

Task 2: Install the uCMDB Discovery Agent

The general procedure to install the uCMDB Discovery Agent is as follows:

1. Transfer files from the installation media to the zSeries platform. Two options are available:

    ➤ Use the **tvinstall_800_zos_zseries.bat** script found on the installation media to assist with the transfer (recommended).

    ➤ Manually FTP the install files.

2. TSO RECEIVE the transmitted files to create product datasets. Two options are available:

    ➤ Use the (TSORECV) Rexx script created by the **tvinstall_800_zos_zseries.bat** script in Step 1 above to execute the TSO RECEIVE commands (recommended).

    ➤ Manually execute the needed TSO RECEIVE commands.

3. Determine the best installation approach for your environment. Two options are available:

    ➤ SMP/E (recommended). You can either create a new SMP/E global zone (recommended) or use an existing SMP/E global zone.

    ➤ Non-SMP/E

4. Tailor and execute the installation jobs. The degree of customization needed will vary depending on the installation approach. Two options are available:

    ➤ SMP/E installation jobs

    ➤ Non-SMP/E installation jobs

5. Review installation libraries.

File Name Conventions

File names on the installation media adhere to the following pattern:

dsslr<**filequal**>**_<**ver**>_zos_zseries.xmit**

FTP output dataset file names should adhere to the following pattern:

<**&custhlq**>**.<**prodcode**>|<**ver**>.<**filequal**>

where:

<**filequal**> = f1 | f2 | f3 | f4 | mcs

<**prodcode**> = SLR

<**ver**> = 800

<&custhlq> = a customer-provided high level qualifier for output of the FTP PUT sub-command on the target z/OS system.

For example, a file name on the installation media is:

dsslrf1_800_zos_zseries.xmit

An FTP output dataset file name is:

DSCVRY.SLR800.F1

Step 1: Transfer the Files from the Installation Media to the zSeries Host

Two options are available:

➤ Run the **tvisntall_800_zos_zseries.bat** script.

➤ Manually FTP the install files.

Option 1: Run the tvisntall_800_zos_zseries.bat script

1 Logon to a workstation running Microsoft Windows that is capable of connecting to the target z/OS system using TCP/IP FTP.

2 Start an MS-Windows Command Prompt session (**Start** > **All Programs** > **Accessories** > **Command Prompt**). Adjust the window properties, increasing its size and buffering as necessary.

3 Navigate, using the change directory (cd) command, to the folder containing installation media.

4 Review the information that will be collected by the script by entering:

tvinstall_800_zos_zseries.bat /h | more

5 Invoke **tvinstall_800_zos_zseries.bat** with no parameters and respond to the prompts.

Note: If you are not satisfied with the values you enter, you can quit tvinstall_800_zos_zseries.bat without initiating a file transfer. In addition, you can terminate the script at any time by entering **CTRL+C** repeatedly.

---

6 Carefully review the results of the FTP command output issued by
tvinstall_800_zos_zseries.bat.

➤ There should be no errors.

➤ Check for the existence of installation RELFILES with the high-level
qualifiers specified on the target z/OS system.

➤ Check for the existence of the Rexx script (TSORECV) in the PDS library
specified.

➤ If any datasets are missing or FTP errors were detected, they should be
investigated, corrected, and the tvinstall_800_zos_zseries.bat process
repeated.

Option 2: Manually FTP the install files:

Don't perform this step if you installed the files by using the
tvisntall_800_zos_zseries.bat script.

1 Logon to a workstation running Microsoft Windows that is capable of
connecting to the target z/OS system using TCP/IP FTP.

2 Start an MS-Windows Command Prompt session (**Start** > **All Programs** >
Accessories > **Command Prompt**). Adjust the windows properties, increasing
its size and buffering, as necessary.

3 Navigate, using the change directory (cd) command, to the folder
containing installation media.

4 Using a z/OS account with permissions sufficient to create datasets with the
desired high level qualifier, initiate an FTP session with the target z/OS
system, for instance:

> ftp hostname

5 Issue the following FTP sub-commands:

a ftp> **quote site fixrecfm 80 lrecl=80 recfm=fb blksize=3120 vol=&custvol
u=&custunit pri=30 sec=5 tr**

where **&custvol** is a valid disk volser and **&custunit** is a valid disk device
type or esoteric

b ftp> **bin**

c ftp> **put dsslr<filequal>_<ver>_zos_zseries.xmit**
'**<&hlq>.slr<|ver>.<filequal>**'

where
<&hlq> = customer selected high level qualifier
<filequal> = f1 | f2 | f3 | f4 | mcs
<ver> = 800

For example:

PUT dscvry_agent_slr_800_zos_zseries.xmit '&hlq.slr|ver.filequal'

d Issue FTP **put** commands for any remaining product installation files
with a filename suffix of **fn** or **mcs**.

e It is important that FTP command output be carefully examined for
errors. If detected, the target dataset on the z/OS system may need to be
deleted and the ftp put command re-executed.

f ftp> **quit**

Step 2: TSO RECEIVE the Transmitted Files to Create Product Datasets

Two options are available:

➤ Run the (TSORECV) Rexx script created by the
tvinstall_800_zos_zseries.bat in step "Step 1: Transfer the Files from the
Installation Media to the zSeries Host" on page 384.

➤ Manually execute the needed TSO RECEIVE commands

Option 1: Execute the (TSORECV) Rexx script

From a TSO command line or TSO/ISPF command line, prefixing with 'TSO'
as necessary, execute the (TSORECV) Rexx script contained in the PDS
identified in step 1 above.

exec 'PDS.identified.instep1(TSORECV)'

This executes the TSO RECEIVE commands for all of the z/OS target datasets
specified in step 1, 6 above. The RECEIVEd datasets are allocated using the
same high level qualifier, however the character 'A' is added as a prefix to
the prodcode.

For example, if input pattern=DSCVRY.SLR800.filequal, then the output pattern would be DSCVRY.ASLR800.filequal.

Option 2: Manually execute the needed TSO RECEIVE commands

1 From a TSO command line or TSO/ISPF command line, prefixing with 'TSO' as necessary, execute RECEIVE commands as shown in the example table below. In response to prompt 'INMR906A Enter restore parameters or 'DELETE' or 'END'" enter 'DSN(&custhlq.ASLR800.filequal)'.

Command	Filename
RECEIVE INDSNAME('&hlq.SLR8000.F1')	DSN('&hlq.ASLR800.F1')
RECEIVE INDSNAME('&hlq.SLR8000.F2')	DSN('&hlq.ASLR800.F2')
RECEIVE INDSNAME('&hlq.SLR800.F3')	DSN('&hlq.ASLR800.F3')
RECEIVE INDSNAME('&hlq.SLR800.F4')	DSN('&hlq.ASLR800.F4')
RECEIVE INDSNAME('&hlq.SLR800.MCS')	DSN('&hlq.ASLR800.SMPMCS')

Step 3: Determine the Best Installation Approach

Two options are available:

> ➤ SMP/E (recommended). You can either create new SMP/E global zone (recommended) or use an existing SMP/E global zone.

> ➤ Non-SMP/E

Step 4: Tailor and Execute the Installation Jobs

Two options are available: SMP/E or non-SMP/E.

Option 1: SMP/E installation jobs

Tailor and execute the jobs in the order presented. Jobs are in **dataset &custhlq.ASLR800.F3**. Read the comments at the beginning of each job and customize accordingly.

Following execution, carefully review all output making sure all steps completed successfully before moving on to the next job. If installing into an existing SMP/E global zone, carefully review SMP/E steps, parameters, and inputs.

SMP/E installation jobs:

Order	Job	Description
1.	**SLRALLOC**	Allocate target and distribution libraries
2.	**SLRGZON**	Defines SMP/E global zone (not needed if using existing global zone)
3.	**SLRDZON**	Defines SMP/E distribution zone
4.	**SLRTZON**	Defines SMP/E target zone
5.	**SLRDDDEF**	Defines SMP/E DDDEFs
6.	**SLRRECV**	SMP/E Receive
7.	**SLRAPPLY**	SMP/E Apply
8.	**SLRIVP**	Installation Verification
9.	**SLRACCPT**	SMP/E Accept **Important!** Run this job only after the installation has been configured and fully tested, since it is not possible to remove the product from target or distribution libraries once the SMP/E ACCEPT function has been executed.

Option 2: Non-SMP/E installation jobs

Jobs are found in **dataset &custhlq.ASLR800.F3**. Read the comments at the beginning of each job and customize accordingly. Following execution, carefully review all output making sure all steps completed successfully.

Non-SMP/E installation jobs:

Order	Job	Description
1.	**SLRINSTL**	
2.	**SLRIVP**	Installation verification

Step 5: Review the Installation Libraries

DS#	Dataset Name	Description
1	DSCVRY.SSLRCARD	Control Cards, Config data
2	DSCVRY.SSLREXEC	Rexx CGI Program Source
3	DSCVRY.SSLRINST	Installation JCL
4	DSCVRY.SSLRLOAD	Service Agent Loadlib (APF Auth)
5	DSCVRY.SSLRPROC	Service Agent Proclib
6	DSCVRY.SSLRSAMP	Sample JCL, Commands, Etc.
7	DSCVRY.SLR800.F1	Inst RELFILE - JCLIN
8	DSCVRY.SLR800.F2	Inst RELFILE - Load Modules
9	DSCVRY.SLR800.F3	Inst RELFILE - Installation JCL
10	DSCVRY.SLR800.F4	Inst RELFILE - Rexx Programs
11	DSCVRY.SLR800.SMPMCS	Inst RELFILE - SMP/E MCS stmts

➤ The table does not include the installation SMP/E distribution libraries (DLIBS). These are not populated until execution of SLRACCPT.

➤ 'DSCVRY' is the default high level qualifier; yours may differ.

Task 3: Install the IBM HTTP Server

The IBM HTTP Server for z/OS Version 5.3 is a z/OS base install component and available without charge to IBM z/OS customers. Refer to the z/OS Program Directory for this product, PSP documentation, and IBMLink for needed product service and fix information.

The most current information for this product is available through the IBM Publications Center at the following URL:

http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi

The notes in this section reference the following IBM publication: *HTTP Server Planning, Installing, and Using* manual (SC34-4826-09).

Chapter 1 of this manual should be reviewed before installing the product.

Follow these steps when installing the IBM HTTP Server.

1 Note the following items related to the installation:

➤ uCMDB Discovery components do not utilize custom DLLs.

➤ The Fast Response Cache Accelerator is not being utilized at this time.

➤ Regarding performance and tuning, uCMDB Discovery activities involve few if any concurrent user connections. Therefore the MaxActive Thread settings, MAXSOCKETS, MAXFILEPROC, and related performance settings, can be set relatively low assuming uCMDB Discovery activities are the primary IBM HTTP Server's primary application workload.

➤ The External Security Manager (ESM) customizations, commands, and examples provided in this manual pertain to the IBM's Security Server (RACF). Implementation of uCMDB Discovery does not require RACF, however use of other ESM products such as ACF2, TopSecret, etc. will require the corresponding commands and functions be identified and executed in the customer's non-RACF ESM environment.

➤ The ESM account under which the IBM HTTP Server executes must have the following capabilities:

  ➤ Authority sufficient to read the Discovery Service Agent loadlib (DS #4).

  ➤ Authority sufficient to execute Rexx scripts located in the IBM HTTPS Servers's ../cgi-gin directory.

2 Follow the instructions in Chapter 3 of the *HTTP Server Planning, Installing, and Using* manual to install the IBM HTTP Server.

Following basic installation, carefully review and execute customization steps 1-10 of Chapter 3. Several of these steps include security customizations such as SSL, private key ring certificates, creation of Web server accounts with non-zero user IDs, and so forth. Although these customizations are not strictly necessary for Web server operation and uCMDB Discovery processing, Hewlett-Packard Software strongly encourages uCMDB Discovery customers take full advantage of features strengthening their environment's overall security profile given the cross platform characteristics of Discovery processing.

The following sample jobs are provided to assist you in configuring security aspects of your server. Some of the samples will not be applicable for every environment. They should be tailored and executed in conjunction with instructions found in Chapter 3 of the *HTTP Server Planning, Installing, and Using* manual.

Corresponding instructions can be found in the Steps 1-5 of Chapter 3, they are made available here for your convenience:

➤ **SLRSEC01**. IMWEB group and WEBADM user definitions

➤ **SLRSEC02**. WEBSRV user definition

➤ **SLRSEC03**. Defining a user with a non-zero USS user-id

➤ **SLRSEC04**. Surrogate class profile definitions

➤ **SLRSEC05**. Turning on program control for MVS datasets

➤ **SLRSEC06**. Turning on program control for z/OS SSL

➤ **SLRSEC07**. Permitting WEBSRV access to a key ring

➤ **SLRSEC08**. Permitting WEBSRV access to cryptographic hardware services

➤ **SLRSEC09**. Giving WEBSRV access to profiles in cryptographic services

➤ **SLRSEC10**. Permitting WEBSRV access to z/OS WorkLoad Management

➤ **SLRSEC11**. Permitting WEBSRV access to SMF Facilities

➤ **SLRSEC12**. Adding the IBM HTTP Server PROC to the started task table

3 Test the basic functionality of the IBM HTTP Server by following the procedures in Chapters 4, 5, and 6 of the *HTTP Server Planning, Installing, and Using* manual.

There appears to be no IVP for the IBM HTTP Server itself, however the material in these chapters will guide you through start-up, shut-down, and access of the Web Server's default front page which provides an alternative configuration interface. It is important to check basic server functionality before proceeding with additional customizations.

Task 4: Enable the IBM HTTP Server for uCMDB Discovery

Once the basic functionality of your IBM HTTP Server has been verified, perform these customizations specific to the operation of uCMDB Discovery components.

1 It is strongly recommended that you enable the SSL communication and basic authentication features of your IBM HTTP Server. See chapter 8 of the *HTTP Server Planing, Installing, and Using* manual.

The following member, found in DS#6 - Sample JCL, may be of value as an example of the RACF commands and related tasks required to enable basic SSL communication.

> SLRSSLEX   SSL Setup - Sample Steps

Use this material with that found in Chapter 8 of the above named manual.

2 Update the STEPLIB concatenation found in the started task JCL for the IBM HTTP Server. The default started task PROCedure name for the IBM HTTP Server is IMWEBSRV. Two libraries need to be added to the STEPLIB concatenation:

a DS #4, the Service Agent Load Library, should be added to the end of the concatenation.

b If DB2 is installed, add the DB2 SDSNLOAD library followed by DB2 RUNLIB.LOAD library as well.

3 Update the primary configuration file, typically **httpd.conf**, associated with this instance of the IBM HTTP Server, to include Exec mapping rules for Rexx CGI scripts. The default location of the file is:

> /etc/httpd.conf

However, this may have changed. Verify the location and name of the primary configuration file by locating the following string within the output messages of the IBM HTTP Server, then initiate an edit session on the file.

> "Using configuration file"

Add the following Exec mapping rules for CGI program and script execution. Create the target directory in the paths specified as necessary.

```
Exec    /uCMDB/*.cgi    /etc/HP/Discovery/cgi-bin/*.cgi
Exec    /uCMDB/*.sh     /etc/HP/Discovery/cgi-bin/*.sh
```

Note: You may change the actual location of the CGI programs and scripts. This should not pose a problem as long as security attributes are also changed.

---

4 The user-id identified in Step 1 of Chapter 3 in the *HTTP Server Planning, Installing, and Using* manual, needs to have the ability to read and execute the Rexx CGI scripts located in the above directory. Adjust security attributes accordingly.

Task 5: Customize the uCMDB Discovery Services Agent

Once the basic functionality of your IBM HTTP Web Server for z/OS has been verified, it is necessary to perform customizations specific to the operation of uCMDB Discovery.

1 After making the appropriate substitutions, execute the following command from the TSO/ISPF Command Shell - Option 6. This copies the Rexx CGI scripts to the USS filesystem (…/dscvry/cgi-bin) identified in Step 3 on page 392.

> OPUTX 'sourcePDS' 'targetUSSfilesys' TEXT CONVERT(YES)

For example:

> OPUTX 'DSCVRY.SSLREXEC' '/etc/HP/Discovery/cgi-bin' TEXT CONVERT(YES)

2 The uCMDB Discovery Service Agent performs SAF basic authentication on the userid and password accompanying discovery requests. This userid and password must be known to the ESM in order for basic authentication to complete successfully.

3 Customize PDS member SLRSRVIN located in DS#1- Control Cards and Configuration Data.

4 Review and tailor as necessary, the Authorized Command List (ACL) file or member referred to in Task 5, Step 3, above. The default ACL member, SLRACL,is located in DS#1 - Control Cards and Configuration Data.

5 APF authorize uCMDB Discovery library &hlq.SSLRLOAD - DS#4. Consult MVS Initialization and Tuning Reference (SA22-7592) for assistance.

6 Tailor the following JCL started task procedures per instructions found at the top of each member, and then copy them to an appropriate JCL PROCcedure library for your site.

Sample JCL for these started task PROCedures can be found in DS#5 – Service Agent Proclib.

Note: If the PROCedure names below are inconsistent with site standards or preferences, you may change their names, however in the case of SLRTSREX, it will also be necessary to reflect the change in the SLRSERV configuration file.

➤ SLRSERV

➤ SLRTSREX

7 SLRSERV and SLRTSREX are executed as MVS started tasks. The default user-id associated with these tasks needs to have the following capabilities:

➤ Create files in the USS file system /tmp/Discovery.

➤ Read authority for all libraries is identified in the STEPLIB DD statement

➤ Read authority for libraries identified by the SYSEXEC DD statement

➤ Read authority for the configuration file and authorized command list

➤ Authority to create and use an MVS MCSE console

➤ Authority to execute all commands identified by the Authorized Command List (ACL)

8 Consult "uCMDB Discovery Agent Components and Operation" on page 396 for a better understanding of the overall architecture, components, and basic operations.

9 After gaining experience with the product, return to Step 3 of "Task 2: Install the uCMDB Discovery Agent" on page 383 and perform the **SLRACCPT (**SMP/E Accept) job.

uCMDB Discovery Agent Components and Operation

This section provides a brief overview of the uCMDB Discovery Agent architecture and instructions on starting and stopping its z/OS based components.

The following table lists the major components of the uCMDB Discovery Agent:

PROCedure	Description
IMWEBSRV	IBM HTTP Server V5.3 for z/OS (default PROCedure name used)
SLRSERV	MVS Console Service Agent (started task)
SLRTSREX	TSO/E Rexx Command Processor (started task)
DDM Client	HP uCMDB DDM Client (remote Discovery and Dependency Mapping Client running on a Linux, Unix, or Windows platform)

The uCMDB architecture is as follows:
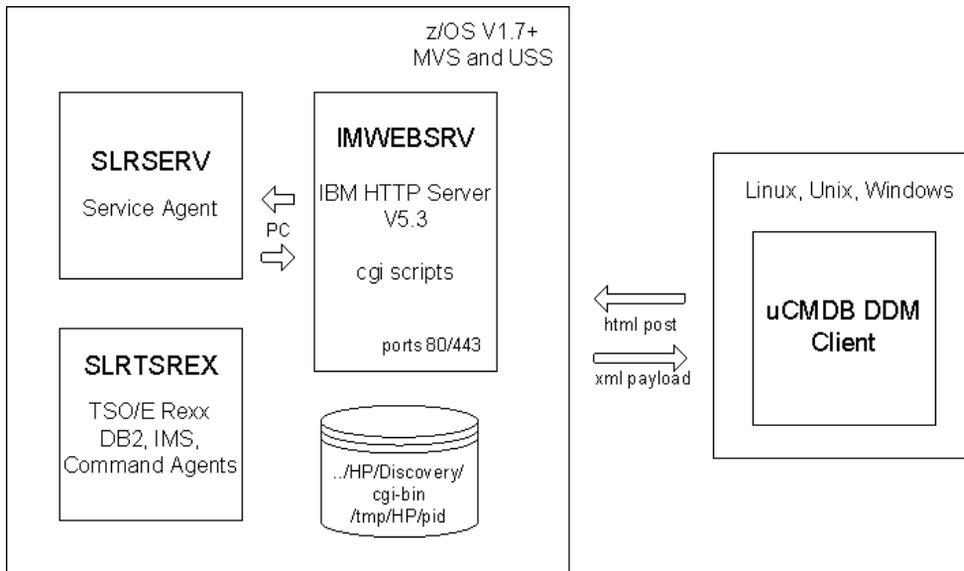
The IBM HTTP Server needs to be up and running in order for SLRSERV to work correctly. See Chapters 4 and 6 of the *HTTP Server Planning, Installing, and Using* manual.

Starting SLRSERV:

To start SLRSERV, use an MVS console "start" command:

```
S SLRSERV
```

Note: The SLRSERV procedure automatically starts SLRTSREX by issuing an internal "start" command to the MVS operating system. Do not attempt to start SLRTSREX manually.

Once started, SLRSERV and its companion task SLRTSREX, can remain running until the next IPL.

Stopping SLRSERV:

If desired, you can bring SLRSERV down by using an MVS "stop" command:

```
P  SLRSERV
```

Note: When SLRSERV is brought down, SLRTSREX comes down automatically. Do not attempt to stop SLRTSREX manually.

Refreshing the ACL:

Dynamic configuration data, including an authorized command list (ACL) is passed to SLRSERV via the PARM= parameter of the EXEC statement. Use the MVS modify 'REFRESH' command to instruct SLRSERV to re-read the ACL, picking up any recent changes. For example:

```
F SLRSERV,REFRESH
```

uCMDB Discovery Agent Security Requirements

The following list is intended to be used in conjunction with any z/OS ESM (External Security Manager) for the purpose of building the ESM specific command strings necessary to grant permissions and authorize uCMDB Discovery Agent components.

All items within the list can be found within the various installation and configuration sections of this document. They are consolidated here for easy reference. The list is broken down by Discovery Agent components. A user-id associated with each Discovery Agent component, whether primary, secondary, or group, needs to be granted permissions sufficient to perform the task described.

The following diagram shows the various resources involved and the Discovery Agent component(s) needing access.
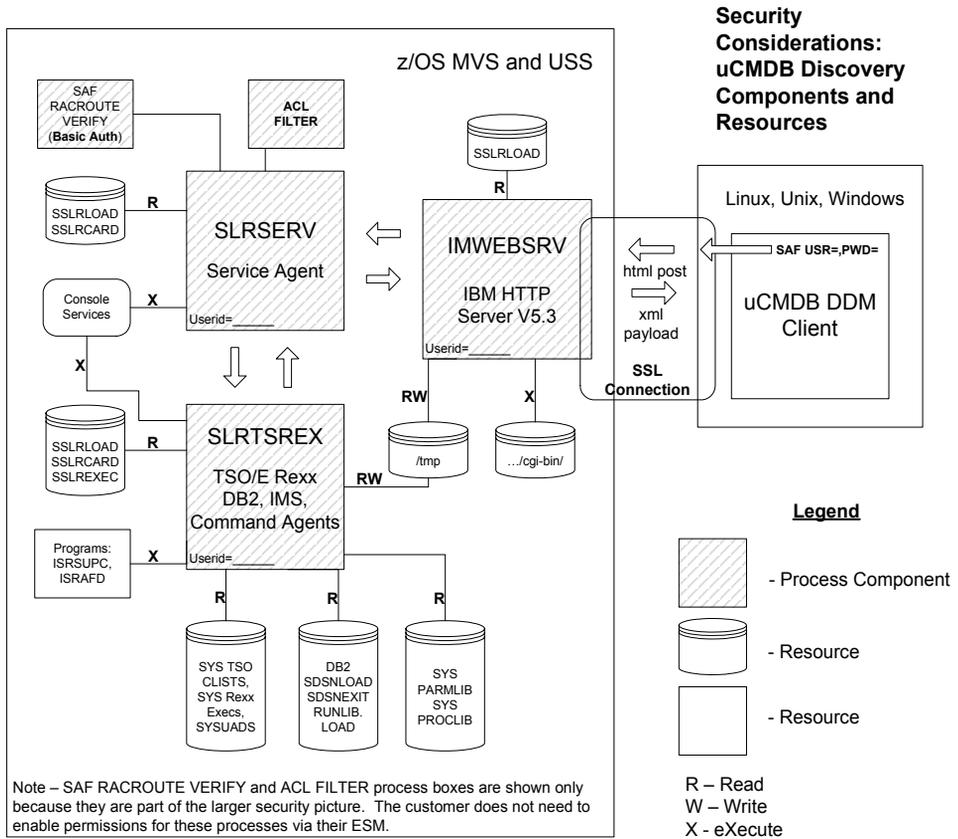
IBM HTTP Server

➤ Execute permission for cgi-bin scripts and programs within the folder identified by the HTTP server's httpd.conf file for Rexx mapping rules. The location of this folder is customizable. The suggested path is /etc/HP/Discovery/cgi-bin/

➤ Read and write access to files and subdirectories within the /tmp filesystem

➤ Read access to DS#4 – DSCVRY.SSLRLOAD (see table in Task2,Step5) (please note – this library must also be defined to MVS as APF authorized)

➤ Although not specifically required by the uCMDB Discovery Service Agent, Hewlett-Packard Software recommends the customer take all appropriate precautions to secure their IBM HTTP Server environment, including SSL and administrative customizations described in the IBM HTTP Server Planning, Installing, and Using manual (SC34-4826). Sample jobstreams, specific to the RACF security environment, have been included which may be of value in enabling SSL and securing the HTTP server (see SLRSSLEX, SLRSEC01-SLRSEC12 in DS#6 – DSCVRY.SSLRSAMP, Sample JCL, Commands, Etc.)

SLRSERV Started Task

➤ MVS Authorized Service calls associated with the creation and use of MCSE console services: MCSOPER (ACTIVATE,DEACTIVATE), MGCRE, MCOPMSG

➤ Execution of various MVS commands (typically display commands), TSO commands, and several Rexx Execs and TSO CLISTS. Some commands are directed to specific subsystems via a 'recognition character', so specific permissions would need to include use of such characters. A full list of all commands potentially issued can be found in the Command Reference section of this document. The ability to execute any command can be controlled by the customer using the ACL (Authorized Command List) discussed below.

➤ Read access to DS#1 – DSCVRY.SSLRCARD (see table in Task2, Step5) or corresponding customer QSAM file or PDS containing the SLRSERV configuration file and ACL (Authorized Command List).

➤ Read access to DS#4 – DSCVRY.SSLRLOAD (see table in Task2,Step5) (please note – this library must also be defined to MVS as APF authorized)

➤ In addition to any granular operator command permissions the customer may have defined using their ESM, the Authorized Command List (ACL) can also be used to control which commands, execs, and clists are executed by the Discovery Service Agent. The location of the ACL is defined within the SLRSERV configuration file, read from the PARM= statement in the SLRSERV procedure. If an ACL is not specified, SLRSERV uses an internally defined default equivalent in content to the default ACL (SLRACL) found within

DS#1 – DSCVRY.SSLRCARD Control Cards, Config data. Please review the content of the ACL file in use and make adjustments as needed. In the absence of, or in conjunction with ESM operator command level security, the ACL provides the customer with the ability to control which commands or class of commands (if wildcarding is used), can be executed by the Discovery Service agent.

SLRTSREX Started Task

➤ Read access to DS#4 – DSCVRY.SSLRLOAD (see table in Task2,Step5) (please note – this library must also be defined to MVS as APF authorized)

➤ Read access to all DB2 SDSNLOAD, SDSNEXIT, and RUNLIB.LOAD libraries if DB2 components are to be discovered.

➤ Read access to local SYSUADS dataset

➤ Read access to local system TSO CLIST and Rexx Exec libraries

➤ Read access to DS#2 – DSCVRY.SSLREXEC (see table in Task2,Step5)

➤ Read access to DS#1 – DSCVRY.SSLRCARD (see table in Task2,Step5)

➤ Read and write access to files and subdirectories within the /tmp filesystem

➤ Read access to system level parameter and procedure libraries (e.g. SYS1.PARMLIB, USER.PROCLIB, etc.)

➤ Ability to execute IBM program ISRSUPC for string searches

➤ Ability to execute IBM SDSF program ISRAFD (for discovery of IMS resources)

➤ Execute authority for DB2 plans DSNREXX and DSNTEP2

➤ MVS Authorized Service calls associated with the creation and use of MCSE console services: MCSOPER (ACTIVATE,DEACTIVATE), MGCRE, MCOPMSG

➤ Execution of various MVS commands (typically display commands), TSO commands, and several Rexx Execs and TSO CLISTS. Some commands are directed to specific subsystems via a 'recognition character', so specific permissions would need to include use of such characters.

Other

➤ The uCMDB DDM Client has its own security requirements.

➤ Associated with all uCMDB DDM Client requests is a user-id and password used by the SLRSERV started task to perform basic SAF authentication. This user-id must be defined to the z/OS ESM.

uCMDB Command Summary

The following commands are executed by the uCMDB Discovery Service Agent. They are broken down by command sub-type within command type. The command types recognized by the Service Agent and meaningful within the ACL are: MVS, TSO, and SAF.   It is understood that some of the commands classified under MVS are actually DB2, CICS, and MQ Series commands. Further command type delineation can be expected in future releases.

MVS

D ASM
D M=CPU
D OPDATA
D NET,MAJNODES
D PROD,REGISTERED
D PROD,STATE
D R
D SSI
D SYMBOLS
D TCPIP,,NETSTAT,CONN
D TCPIP,,NETSTAT,DEV
D TCPIP,,NETSTAT,HOME
D TCPIP,,NETSTAT,ROUTE
D VIRTSTOR,HVSHARE
D XCF,GRP
D XCF,GRP,groupname,ALL

DISPLAY CHANNEL	MQ Series
DISPLAY CHINIT	MQ Series
DISPLAY SYSTEM	MQ Series
CEMT I TAS	CICS

DISPLAY DATABASE   DB2
DISPLAY GROUP   DB2
DISPLAY LOCATION   DB2

TSO

TSO ALTLIB DISPLAY
TSO LISTA SYSNAMES
TSO LISTALC SYSNAMES
TSO NETSTAT
TSO STATUS
TSO TIME

SLRRDFLE   Rexx exec to read a file
SLRREXSC   Rexx exec for dynamic SQL execution
SLRSRPDS   Rexx exec to search a PDS mbr
SLRDTEP2   Rexx exec to invoke DSNTEP2 for dynamic SQL execution
SLRGETSS
SLRIMSDS   Rexx exec used to execute an IMS DISPLAY ACTIVE command
SLREXPRD   TSO CLIST which produces system software inventory report

SAF

SAF USR=,PWD=Internal SAF basic authentication request

uCMDB Mainframe Services Agent Console Messages

The following messages are displayed to the operator console. These messages, along with various other messages, are also written to the MSGOUT dataset.

SLR-0001-E: Not APF authorized

The uCMDB Services Agent is not APF authorized. This message might indicate that the uCMDB Services Agent load library is not APF authorized, or the SLRAGENT load module has a zero authorization code, or the STEPLIB of the execution JCL contains one or more non-authorized libraries, or SLRAGENT was invoked by non-authorized software.

SLR-0002-I: TRANSWAP completed successfully

This message indicates that the uCMDB Services Agent address space has successfully made itself non-swappable. This message does not appear if the uCMDB Services Agent address space was already non-swappable when SLRAGENT received control.

SLR-0003-I: uCMDB services agent (SLRvvvvv) main task starting

This message is displayed when the uCMDB services agent begins its start-up and initialization sequence. The customer may rely on this message for automation purposes. The message includes (as denoted by "**vvvvv**") a version/release code, up to five character in length.

SLR-0011-E: Operating system is too old, no longer supported

The host operating system does not provide all of the functions and capabilities that the uCMDB services agent requires.

SLR-0012-E: Error in authorized command list

The CMDLIST= clause, in the configuration file, specifies the name of an authorized command list (ACL) file. However, the specified ACL file is either inaccessible or contains one or more errors. See MSGOUT for further details.

SLR-0013-E: Error in configuration file

The EXEC statement of the execution JCL contains a non-empty PARM= clause, which specifies the name of a configuration file. However, the specified configuration file is either inaccessible or contains one or more errors. See MSGOUT for further details.

SLR-0014-E: Invalid PARM= clause in execution JCL

The EXEC statement of the execution JCL contains a non-empty PARM= clause. The text specified by the PARM= clause is erroneous in some way. For example, the length of the PARM text might exceed 100 bytes.

SLR-0015-E: ALESERV rc=X'hhhh'

An ALESERV EXTRACTH macro statement returned a non-zero return code. The message contains the return code that was returned by the ALESERV macro. ALESERV return codes are document in *MVS Programming: Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC)*.

SLR-0016-E: Services agent already active

Another instance of the uCMDB services agent is already active in the current z/OS image.

SLR-0017-E: Unable to load xxxxxxxx, retc=X'hhhhhhhh', reas=X'hhhh'

The specified module (denoted by xxxxxxxx) could not be loaded. This message may indicate insufficient memory or that the specified module is not found in STEPLIB.

SLR-0025-I: Attach of subtask xxxxxxxx failed, rc=X'hhhh'

An ATTACH macro statement returned a non-zero return-code. This usually indicates insufficient memory

SLR-0026-I: Subtask xxxxxxxx terminated, ecb=X'40sssuuu'

The specified subtask had terminated. The message shows (a) the terminating subtask's main program name, and (b) the 32-bit contents of the subtask's termination ECB. In general, "**sss**" denotes a system abend code and "**uuu**" denotes a user abend code.

SLR-0028-I: uCMDB Services Agent main task is now active

This message is displayed when the uCMDB services agent has completed its start-up and initialization sequence. The customer may rely on this message for automation purposes.

SLR-0029-I: RQE successfully reclaimed

This message indicates that a client program had issued a request but failed to retrieve the results of that request in a timely manner. The request queue element (RQE) associated with the request has been released and made available for reuse.

SLR-0030-E: Severe error in time management routine

If this message appears, contact technical support!

SLR-0040-W: Unrecognized console command ignored

The console operator issued an improper console command to the uCMDB Services Agent address space. After start-up, the uCMDB Services Agent only recognizes MODIFY and STOP commands from the console operator. All other types of console commands issued by the console operator are ignored.

SLR-0041-I: Modify command data: xxxxxxxx

A MODIFY command was received from the console operator. The data which accompanied the MODIFY command is shown in the message.

SLR-0042-I: STOP command accepted

A STOP command was received from the console operator.

SLR-0043-E: Unable to free a CIB

A QEDIT macro statement returned a non-zero return-code during an attempt to free a command input buffer (CIB).

SLR-0052-E: Unable to obtain memory for an ECB table, rc=X'hhhh'

A STORAGE OBTAIN macro statement returned a non-zero return-code during an attempt to get memory for an ECB table.

SLR-0065-E: IEANTRT failed, rc=X'hhhh'

The z/OS name/token retrieval service routine (IEANTRT) returned a non-zero return-code other than not-found. This message shows, in hex notation, the return-code (**hhhh**) that was returned by IEANTRT. Name/token return codes are documented in *MVS Programming: Assembler Services Reference, Volume 2 (IARR2V-XCTLX)*.

SLR-0069-E: IEANTCR failed, rc=X'hhhh'

The z/OS name/token creation service routine (IEANTCR) returned a non-zero return-code. This message shows, in hex notation, the return-code (**hhhh**) that was returned by IEANTRT. Name/token return codes are documented in *MVS Programming: Assembler Services Reference, Volume 2 (IARR2V-XCTLX)*.

SLR-0072-E: IEANTDL failed, rc=X'hhhh', name= tokenname

The z/OS name/token deletion service routine (IEANTDL) returned a non-zero return-code during an attempt to delete a named token. This message provides the return-code (**hhhh**) that was returned by IEANTDL and the name of the token (**tokenname**) that IEANTDL was trying to delete. Name/token return codes are documented in *MVS Programming: Assembler Services Reference, Volume 2 (IARR2V-XCTLX)*.

SLR-0212-E: Not APF authorized

The TSO/E-Rexx address space is not APF authorized. This message might indicate one or more of the following problems in the JCL procedure specified by the TSOPROC= parameter in the configuration file: (1) The TSO/E-Rexx load library is not APF authorized; (2) The SLRTSREX load module has a zero authorization code; (3) The STEPLIB of the execution JCL contains one or more non-authorized libraries; (4) Program SLRTSREX was invoked by non-authorized software.

SLR-0213-E: Unable to retrieve named token, name= tokenname

The TSO/E-Rexx address space was not able to retrieve the token whose name is denoted by **tokenname**. See MSGOUT for further information, such as the return code that was returned by IEANTRT.

SLR-0214-E: Terminating due to ALESERV error

A non-zero return-code was returned by either an ALESERV EXTRACTH or ALESERV ADD macro statement. See MSGOUT for further information, such as the ALESERV return code value.

SLR-0215-E: Terminating due to ASEXT error

A non-zero return-code was returned by either an ASEXT macro statement. See MSGOUT for further information, such as the ASEXT return-code value.

SLR-0216-E: Unable to open TASKLIB

The JCL procedure, specified by the TSOPROC= parameter in the configuration file, does not contain a valid TASKLIB DD statement.

SLR-0331-E: Unable to allocate ACL file

The CMDLIST= parameter, in the configuration file, specifies the name of an authorized command list (ACL) file. However, the specified ACL file cannot be allocated. The ACL file may not exist, or the uCMDB services agent may not have sufficient authority to read the ACL file. See MSGOUT for further information, including the dynamic allocation return code, error reason code and error information code.

SLR-0332-E: Unable to de-allocate ACL file

Dynamic de-allocation of the ACL file failed. See MSGOUT for further information, including the dynamic allocation return code, error reason code and error information code.

SLR-0333-E: ddname not allocated

This message indicates that a previously-allocated DD-name is no longer present in the TIOT. If this message appears, contact technical support!

SLR-0334-E: Unable to open ACL file

An OPEN macro statement returned a non-zero return-code during an attempt to open a previously-allocated ACL file. The likely reason is that the ACL file's format is not supported by the uCMDB services agent. The ACL file must be a "flat" MVS file, a member of a partitioned dataset (PDS), or an EBCDIC text file in a Unix Systems Services hierarchical file system. Other file formats, such as VSAM, are not supported. See MSGOUT for further information.

SLR-0335-E: Unable to close ACL file

A CLOSE macro statement returned a non-zero return-code during an attempt to close a previously-opened ACL file. See MSGOUT for further information.

SLR-0356-E: Stmt nnnnn invalid key xxxxxxxx

This message is displayed when a statement, in the configuration file, specifies an unrecognized keyword. The message shows the line number (**nnnnn**) of the statement in error, and the erroneous keyword (**xxxxxxxx**) specified in that statement.

SLR-0358-E: Stmt nnnnn has no equal sign

This message is displayed when a statement, in the configuration file, does not contain an equal sign. Each non-comment statement in the configuration file must consist of a keyword, followed by an equal sign, followed by a string value to be assigned to the specified keyword. A statement is invalid if it does not contain an equal sign. The message shows the line number (**nnnnn**) of the statement in error.

SLR-0359-E: Configuration file exceeds max record count

The total number of records (including comments) in the configuration file exceeds 999.

SLR-0362-E: Unable to allocate configuration file

A configuration file name was specified in the PARM= clause of the EXEC statement of the execution JCL. However, the named file could not be allocated. The file may not exist, or the uCMDB services agent may not be authorized to access the file for read-only. MSGOUT provides additional information, such as dynamic allocation return code, error reason code and error information code.

SLR-0363-E: Unable to de-allocate configuration file

An error occurred during an attempt to de-allocate a previously allocated configuration file. MSGOUT provides additional information, such as dynamic allocation return code, error reason code and error information code.

SLR-0364-E: Unable to find ddname in TIOT

This message indicates that a previously-allocated DD-name is no longer present in the TIOT. If this message appears, contact technical support!

SLR-0365-E: Unable to open configuration file

An OPEN macro statement returned a non-zero return code during an attempt to open a previously allocated configuration file. The likely reason is that the configuration file's format is not supported by the uCMDB services agent. The configuration file must be a "flat" MVS file, a member of a partitioned dataset (PDS), or an EBCDIC text file in a Unix Systems Services hierarchical file system. Other file formats, such as VSAM, are not supported.

SLR-0366-E: Unable to close configuration file

A CLOSE macro statement returned a non-zero return-code during an attempt to close a previously-opened configuration file.

SLR-0368-W: No configuration file name specified

The PARM= clause of the EXEC statement of the execution JCL does not specify a configuration file name. The PARM= clause is empty or omitted. Default configuration parameters will be used.

uCMDB z/OS Discovery Error Messages

\<return code>	\<message>	Probable Cause	Plan of Action
00010	get_process_vars\getcwd {working_directory}\{errno}\{errnojr}	IBM HTTPD Server not setup correctly	- Check {errno} and {errnojr} if available for the reason. - Check IMWEBSRV SYSOUT for error messages.
00010	Get_process_vars\getpid\ {errno}\{errnojr}	IBM HTTPD Server not setup correctly	- Check {errno} and {errnojr} if available for the reason. - Check IMWEBSRV SYSOUT for error messages.
00020	Invalid HTTP request method\<method>{meth}	uCMDB software setup error	- Check IMWEBSRV SYSOUT for error messages. - Check SSL setup.
00100	UserID Required	uCMDB client not setup to pass UserID/ Password but uCMDB agent requires UserID/Password	Setup uCMDB client with UserID/Password.
00110	Password Required	uCMDB client did not pass the Password	Setup uCMDB client Password.
00120	Authentication Failed - service {service error message}	Security	- Check {service error message} - Check IMWEBSRV SYSOUT for error messages. - Contact your Access / Security Administrator.
00130	SAF {service error message}	Security	- Check {service error message} - Check IMWEBSRV SYSOUT for error messages. - Contact your Access / Security Administrator.
00140 *	Authentication Failed - bpxwunix retval={retval}	Security	Contact your Access / Security Administrator.

00150 *	Authentication Failed - no response	Security	Contact your Access / Security Administrator.
00160 *	{error message}	Security	- Check {error message}
			- Contact your Access / Security Administrator.
00500	Invalid Command = {cmd}	uCMDB software setup error. SSL setup error.	- Check IMWEBSRV SYSOUT for error messages.
11000	{error message}	Either Security or Resource problem.	- Check {error message}
			- Check IMWEBSRV SYSOUT for error messages.
11010	Empty message returned for CONNLIST		Check IMWEBSRV SYSOUT for error messages.
13000	{error message}	Either Security Related or DB2 Access Related	- Check {error message}
			- Check IMWEBSRV SYSOUT for error messages.
			- Check TSO information for security for userid of uCMDB TSO Service SLRTSREX.
			- Check DB2 access for userid of uCMDB TSO Service SLRTSREX.
13010	Empty message returned for DSNTEP2	Either Security or DB2 Access	- Check IMWEBSRV SYSOUT for error messages.
			- Check TSO information for security for userid of uCMDB TSO Service SLRTSREX.
			- Check DB2 access for userid of uCMDB TSO Service SLRTSREX.
14000	{error message}		- Check {error message}
			- Check IMWEBSRV SYSOUT for error messages.

14010	Empty message returned for GETSUBSYS		- Check IMWEBSRV SYSOUT for error messages.
15000	{error message}		- Check {error message}
15010	Empty message returned for GETMSTJCL		
16000	{error message}		- Check {error message}
16010	Empty message returned for HOMELIST		
17000	{error message}		- Check {error message}
17010	Empty message returned for IMSCMD		
18000	{error message}		
18010	Empty message returned for LINKLIST		
20000	{error message}	- Security - Resource Timeout	- Check {error message} - Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of uCMDB TSO Service SLRTSREX.
20010	Empty message returned for LISTPRD2	- Security - Resource Timeout	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of uCMDB TSO Service SLRTSREX.
21000	{error message}	Security	- Check {error message} - Check authorization for userid of uCMDB service SLRSERV.
21010	Empty message returned for LISTPROD	Security	Check authorization for userid of uCMDB service SLRSERV.

22000	{error message}	Security	- Check {error message}
			- Check authorization for userid of uCMDB service SLRSERV.
22010	Empty message returned for LISTPROD	Security	- Check authorization for userid of uCMDB service SLRSERV.
23000	{error message}	Security	- Check {error message}
			- Check authorization for userid of uCMDB service SLRSERV.
23010	Empty message returned for MAJNODES	Security	- Check authorization for userid of uCMDB service SLRSERV.
24000	{error message}	Security	- Check {error message}
			- Check authorization for userid of uCMDB service SLRSERV.
24010	Empty message returned for MVSCMD	Security	- Check authorization for userid of uCMDB service SLRSERV.
25000	{error message}	Security	- Check {error message}
			- Check authorization for userid of uCMDB service SLRSERV.
25010	Empty message returned for PAGELIST	Security	- Check authorization for userid of uCMDB service SLRSERV.
26000	{error message}	Security	- Check {error message}
			- Check authorization for userid of uCMDB service SLRSERV.
26010	Empty message returned for ROUTE	Security	- Check authorization for userid of uCMDB service SLRSERV.

27000	{error message}	Either Security or DB2 Access	- Check {error message} - Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of uCMDB TSO Service SLRTSREX. - Check DB2 access for userid of uCMDB TSO Service SLRTSREX.
27010	Empty message returned for DSNREXX	Either Security or DB2 Access	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of uCMDB TSO Service SLRTSREX. - Check DB2 access for userid of uCMDB TSO Service SLRTSREX.
28000	{error message}	Security	- Check {error message} - Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of uCMDB TSO Service SLRTSREX.
28010	Empty message returned for SEARCHPDS	Security	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of uCMDB TSO Service SLRTSREX.
29000	{error message}	Security	- Check authorization for userid of uCMDB service SLRSERV.
29010	Empty message returned for SSILIST	Security	- Check authorization for userid of uCMDB service SLRSERV.

30000	{error message}	Security	- Check {error message} - Check authorization for userid of uCMDB service SLRSERV.
30010	Empty message returned for SYMLIST	Security	- Check authorization for userid of uCMDB service SLRSERV.
31000	{error message}	Security	- Check {error message} - Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of uCMDB TSO Service SLRTSREX.
31010	Empty message returned for TSOCMD	Security	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of uCMDB TSO Service SLRTSREX.
32000	{error message}	Security Resource Timeout	- Check {error message} - Check IMWEBSRV SYSOUT for error messages. - Check Unix System Services authorization for IMWEBSRV userid.
32010	Empty message returned for UNIXCMD	Security Resource Timeout	- Check IMWEBSRV SYSOUT for error messages. - Check Unix System Services authorization for IMWEBSRV userid.
33000	{error message}	Security Resource Timeout	- Check {error message} - Check IMWEBSRV SYSOUT for error messages. - Check Unix System Services authorization for IMWEBSRV userid.

33010	Empty message returned for UPLOAD	Security	- Check IMWEBSRV SYSOUT for error messages. - Check Unix System Services authorization for IMWEBSRV userid.
34000	{error message}	Security Resource Timeout	- Check {error message} - Check authorization for userid of uCMDB service SLRSERV.
34010	Empty message returned for XCFLIST	Security Resource Timeout	- Check authorization for userid of uCMDB service SLRSERV.
35000	{error message}	Security Resource Timeout	- Check {error message} - Check authorization for userid of uCMDB service SLRSERV.
35010	Empty message returned for XCFLIST	Security Resource Timeout	- Check authorization for userid of uCMDB service SLRSERV.
36000	{error message}	Security Resource Timeout	- Check {error message} - Check authorization for userid of uCMDB service SLRSERV.
36010	Empty message returned for XCFLIST2	Security Resource Timeout	- Check authorization for userid of uCMDB service SLRSERV.

See additional information about {error message} below:

{error message}	Probable Cause	Plan of Action
do_console_cmd\address tso {cmd}\{errno}\{errnojr}	Security	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of IMWEBSRV.

do_dsnrexx\address tso {cmd}\{errno}\{errnojr}	- Security - DB2 Access	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of IMWEBSRV. - Check DB2 Authorization for userid of IMWEBSRV.
do_dsntep2\address tso {cmd}\{errno}\{errnojr}	- Security - DB2 Access	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of IMWEBSRV. - Check DB2 Authorization for userid of IMWEBSRV.
do_get_parmmembs\address tso {cmd}\{errno}\{errnojr}	Security	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of IMWEBSRV.
do_listprod_clist\{address tso {cmd}\{errno}\{errnojr}	- Security - Resource Timeout	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of IMWEBSRV.
do_search_pds\address tso {cmd}\{errno}\{errnojr}	- Security - Resource Timeout	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of IMWEBSRV.
do_tso_cmd\address tso {cmd}\{errno}\{errnojr}	- Security - Resource Timeout	- Check IMWEBSRV SYSOUT for error messages. - Check TSO information for security for userid of IMWEBSRV.
do_unix_cmd\bpxwunix {cmd}\{errno}\{errnojr}	- Security - Resource Timeout	- Check IMWEBSRV SYSOUT for error messages. - Check Unix System Services authorization for userid of IMWEBSRV.
{filename} file not opened, error codes {errno} - {errnojr}	Security	- Check IMWEBSRV SYSOUT for error messages. - Check Unix System Services authorization for userid of IMWEBSRV.
INVALID uCMDB_mode - {uCMDB_mode} lib - {uCMDB_lib}	uCMDB setup error.	- Check IMWEBSRV SYSOUT for error messages. - Check uCMDB setup including SSL.

SLRCMD {command} rc={rc}, *** Buffer Overflow : Buffer Size={buffer size} Returned={length}	uCMDB critical error.	
SLRCMD {command} rc={rc}, Console server is not active	Started task SLRSERV not running	- Check whether started task SLRSERV is running
SLRCMD {command} rc={rc}, Named token not found	Resource Timeout	- Check SLRSERV SYSOUT for error messages
SLRCMD {command} rc={rc}, Named token services not available	Resource Timeout	- Check SLRSERV SYSOUT for error messages
SLRCMD {command} rc={rc}, Unable to pass command to console server	Resource Timeout	- Check SLRSERV SYSOUT for error messages
SLRCMD {command} rc={rc}, No reply from server within timeout interval	Resource Timeout	- Check SLRSERV SYSOUT for error messages
SLRCMD {command} rc={rc}, Non-zero completion code posted in ECB	Resource Timeout	- Check SLRSERV SYSOUT for error messages
SLRCMD {command} rc={rc}, Cannot retrieve results from console server	Resource Timeout	- Check SLRSERV SYSOUT for error messages
SLRCMD {command} rc={rc}, Unrecognized return code {return code}	Resource Timeout	- Check SLRSERV SYSOUT for error messages

Index