HP Business Service Management

for the Windows and Linux operating systems

Software Version: 9.02

TransactionVision Deployment

Document Release Date: August 2011
Software Release Date: March 2011

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

Trademark Notices

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

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

AMD and the AMD Arrow symbol are trademarks of Advanced Micro Devices, Inc.

Google™ and Google Maps™ are trademarks of Google Inc.

Intel®, Itanium®, Pentium®, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and other countries.

iPod is a trademark of Apple Computer, Inc.

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

Microsoft®, Windows®, Windows NT®, Windows® XP, and Windows Vista® 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 Open Group.

Acknowledgements

This product includes software developed by the Apache Software Foundation (**http://www.apache.org**).

This product includes software developed by the JDOM Project (**http://www.jdom.org**).

This product includes software developed by the MX4J project (**http://mx4j.sourceforge.net**).

Documentation Updates

The title page of this document 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

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 online support 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 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 register for an HP Passport ID, go to:

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

To find more information about access levels, go to:

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

Table of Contents

Welcome to This Guide

This guide describes how to deploy and maintain TransactionVision for use with the Transaction Management application in HP Business Service Management (BSM).

This chapter includes:

How This Guide Is Organized

This guide contains the following parts:

Part I Introduction to TransactionVision

Introduces TransactionVision and provides an overview of the TransactionVision components and how they fit in the BSM deployment environment.

Part II Processing Server Installation

Describes how to install and configure the TransactionVision Processing Server.

Part III Agent Installation and Configuration

Describes how to install and configure the TransactionVision agents.

Part IV Security

Describes how to secure the TransactionVision components.

Part V Upgrading

Describes how to upgrade TransactionVision components to 9.02.

Who Should Read This Guide

This guide is intended 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.

How Do I Find the Information That I Need?

This guide is part of the HP Business Service Management Documentation Library. This Documentation Library provides a single-point of access for all HP Business Service Management documentation.

You can access the Documentation Library by doing the following:

- In Business Service Management, select **Help** > **Documentation Library**.
- From a Business Service Management Gateway Server machine, select **Start** > **Programs** > **HP Business Service Management** > **Documentation**.

TransactionVision and Transaction Management Documentation

TransactionVision documentation provides information on deploying and administering the TransactionVision-specific components in the Business Service Management deployment environment. Transaction Management documentation provides information on using the Transaction Management application.

The TransactionVision and Transaction Management documentation includes:

- The *TransactionVision Deployment Guide* describes the installation and configuration of the TransactionVision Processing Servers and agents in the Business Service Management deployment environment. This guide is available as a PDF in the Documentation Library.
- *Using Transaction Management* describes how to set up and configure TransactionVision to track transactions and how to view and customize reports and topologies of business transactions. It also describes how to define business transaction CIs. This guide is available as the Transaction Management Portal or as a PDF in the Business Service Management Online Documentation Library.
- The *TransactionVision Planning Guide* contains important information for sizing and planning new installations of .
- The *TransactionVision Advanced Customization Guide* contains information for how the TransactionVision platform can be extended and customized to achieve further control over its various functions. It presents an architecture overview of the TransactionVision system and documents the different methods available to use and extend the Analyzer, the query service and the TransactionVision user interface. This guide is available as a PDF in the Transaction Management Portal > Administration > Advanced Customization Using the APIs topic.

Note: Updates to these guides sometimes occur independently of the software. See "Documentation Updates" on page 18 for information on how to get the most current documentation.

Additional TransactionVision and Transaction Management documentation can be found in the following areas of the Business Service Management:

Readme. Provides a list of version limitations and last-minute updates. From the HP Business Service Management DVD root directory or download package root directory, double-click **readme<version>.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 Service Management, select **Help** > **What's New**.

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

To access the Documentation Library, in HP Business Service Management, select **Help** > **Documentation Library**. Context-sensitive help is available from specific 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 Service Management 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

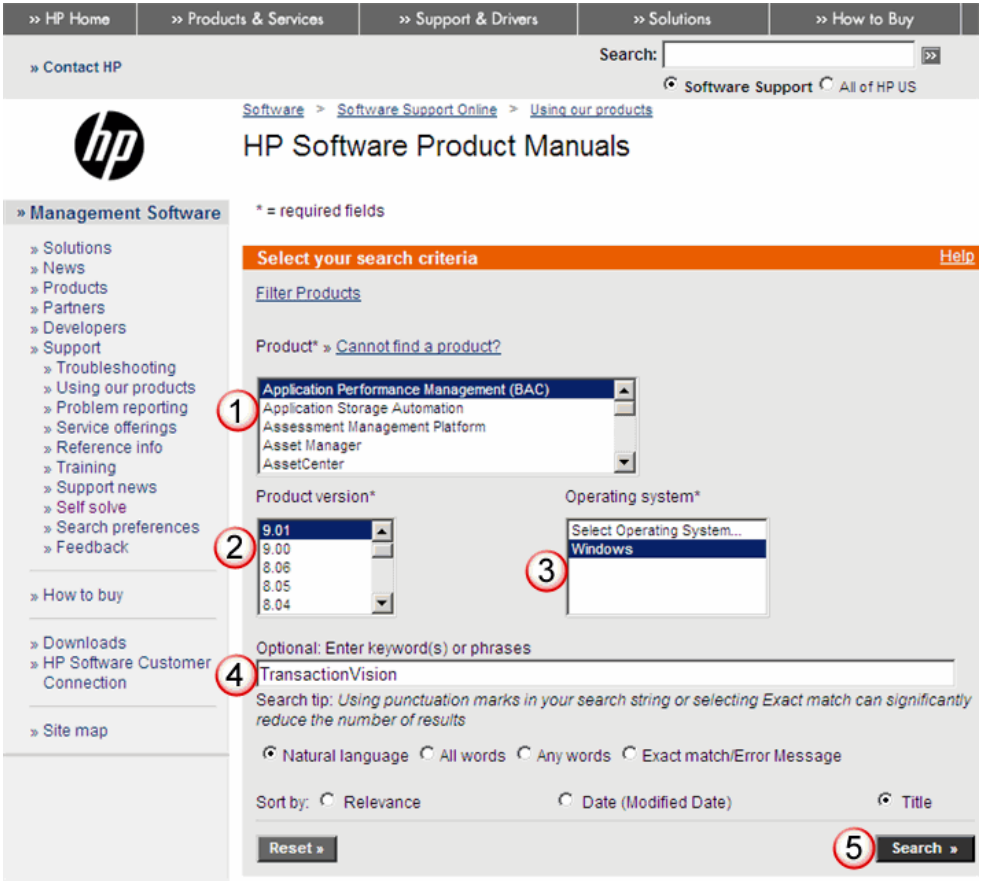
HP Software Web site accesses the HP Software Web site. This site provides you with the most up-to-date information on HP Software products. This includes new software releases, seminars and trade shows, customer support, and more. Choose **Help** > **HP Software Web site**. The URL for this Web site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

To search for TransactionVision documentation, choose **Application Performance Management (BAC)**, the desired product version and operating system, and specify the keyword as follows:



Part I

Introduction to TransactionVision

1

Introduction to TransactionVision

This chapter includes:

About TransactionVision

HP TransactionVision is the transaction tracking solution that 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.

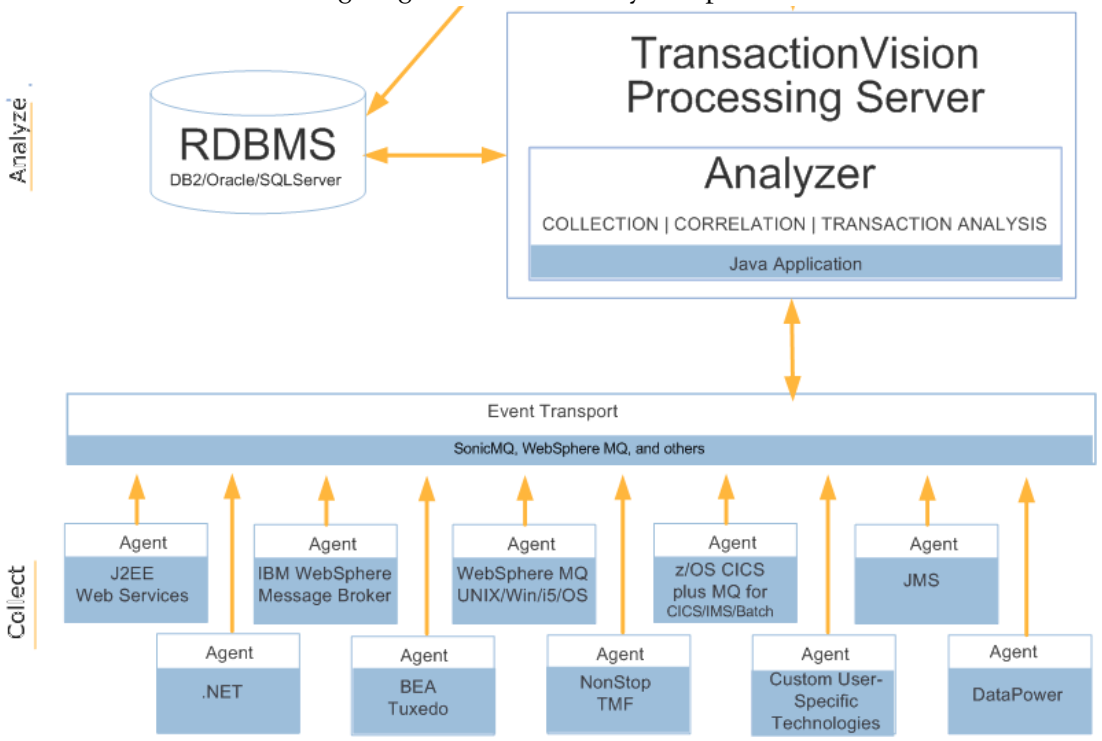
Key capabilities of TransactionVision include:

- Non-intrusively tracks each application event across each processing step.
- Automatically correlates application events into business transactions.
- Collects both technical and business data for each business transaction, identifying each transaction's business context (customer identity, financial value, etc).
- Provides end-to-end visibility of individual business transactions to the Transaction Management application's reports and topologies.

- Provides deep visibility to reduce mean time to problem isolation & resolution of business transaction issues.
- Tightly integrated with other key applications in HP Business Service Management including End User Monitoring, Diagnostics and Business Process Insight.

Architecture

The following diagram shows the key components of TransactionVision.



Each component is described in the sections that follow.

TransactionVision Processing Server

The TransactionVision Processing Server is a container for the TransactionVision components that deliver the core functionality of business transaction tracing to BSM. A Processing Server typically runs on its own host, separate from other BSM components.

The deployment environment can contain multiple Processing Servers. Each Processing Server can contain any of the following TransactionVision components:

- **Job Manager**. Manages the built-in and custom jobs that are used by TransactionVision.

  Job Managers are designated as either primary or backup. One and only one Processing Server in the deployment environment must contain the primary Job Manager.

- **Query Engine.** Manages the queries that are used to populate some of the Transaction Management reports and topologies.

  Like Job Managers, Query Engines are designated as either primary or backup. One and only one Processing Server in the deployment environment must contain the primary Query Engine.

- **Analyzers.** The Analyzers communicate with TransactionVision agents and process event data collected by the agents into transactions. At least one Processing Server in the deployment environment must contain an Analyzer.

See Part II, "Processing Server Installation," for information about the installation and configuration of the TransactionVision Processing Servers.

Analyzers

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

By default, TransactionVision uses SonicMQ as the messaging middleware provider. WebSphere MQ is also supported. TIBCO EMS and WebLogic JMS are supported for 8.0x agents and sensors only.

The Analyzer also retrieves events placed on an event queue by agents and processes them for analysis and display by the reports and topologies in the Transaction Management application of HP Business Service Management. It performs the unmarshalling, correlation, analysis, and data management functions.

See *Using Transaction Management* for information about configuration of the Analyzer.

Agents

TransactionVision agents collect transactional events from the various applications involved in your distributed transactions. Agents are lightweight libraries or exit programs that are installed on each computer in your environment.

Each agent monitors calls made by supporting technologies on that system and compares them against filter conditions. If the call matches the filter conditions, the agent collects entry information about the call, then passes the call on to the appropriate library for processing. When the call returns, the agent 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.

See Part III, "Agent Installation and Configuration," for information about the installation and configuration of the agents.

RDBMS (Database)

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

See "Configuring Databases" on page 65 for more information.

TransactionVision in the Business Service Management Deployment Environment

TransactionVision operates in the HP Business Service Management deployment environment. The HP Business Service Management has two types of deployment scenarios:

➤ "One-Machine Deployment" on page 27

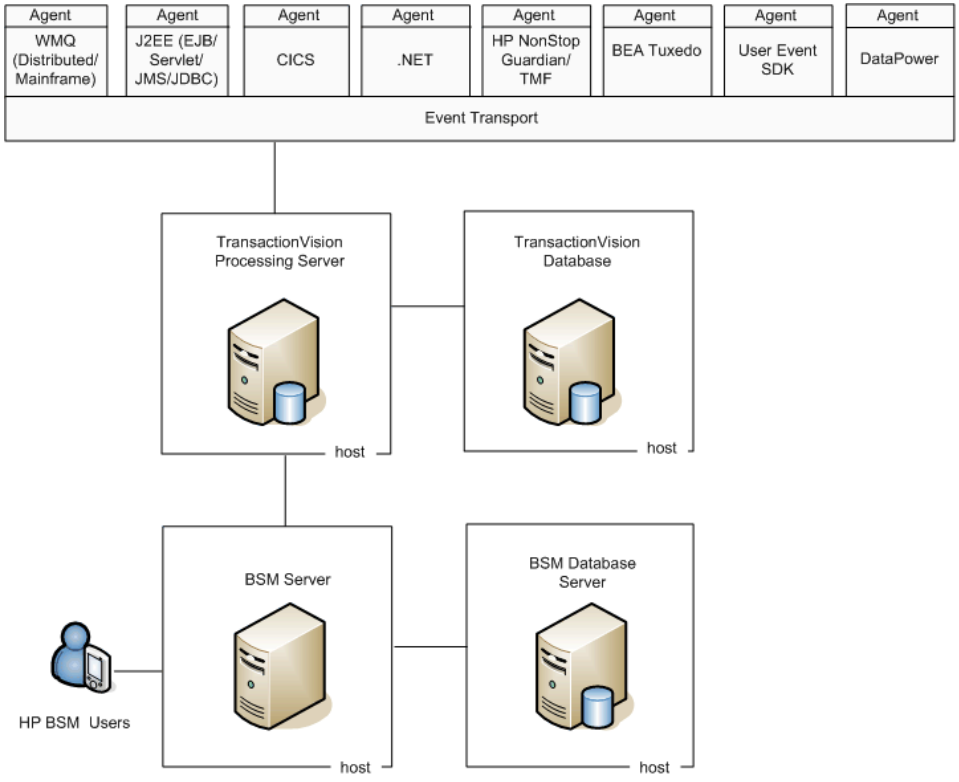
➤ "Two-Machine Deployment" on page 28

For information about setting up these deployment environments, see the *HP Business Service Management Deployment Guide* PDF.

One-Machine Deployment

A one-machine deployment has the BSM Gateway Server and the BSM Data Processing Server on the same machine shown as BSM Server below. The BSM Database Server is on a separate machine. A one-machine deployment should be used primarily for development and testing purposes.

In this environment, a single TransactionVision Processing Server is typically used.

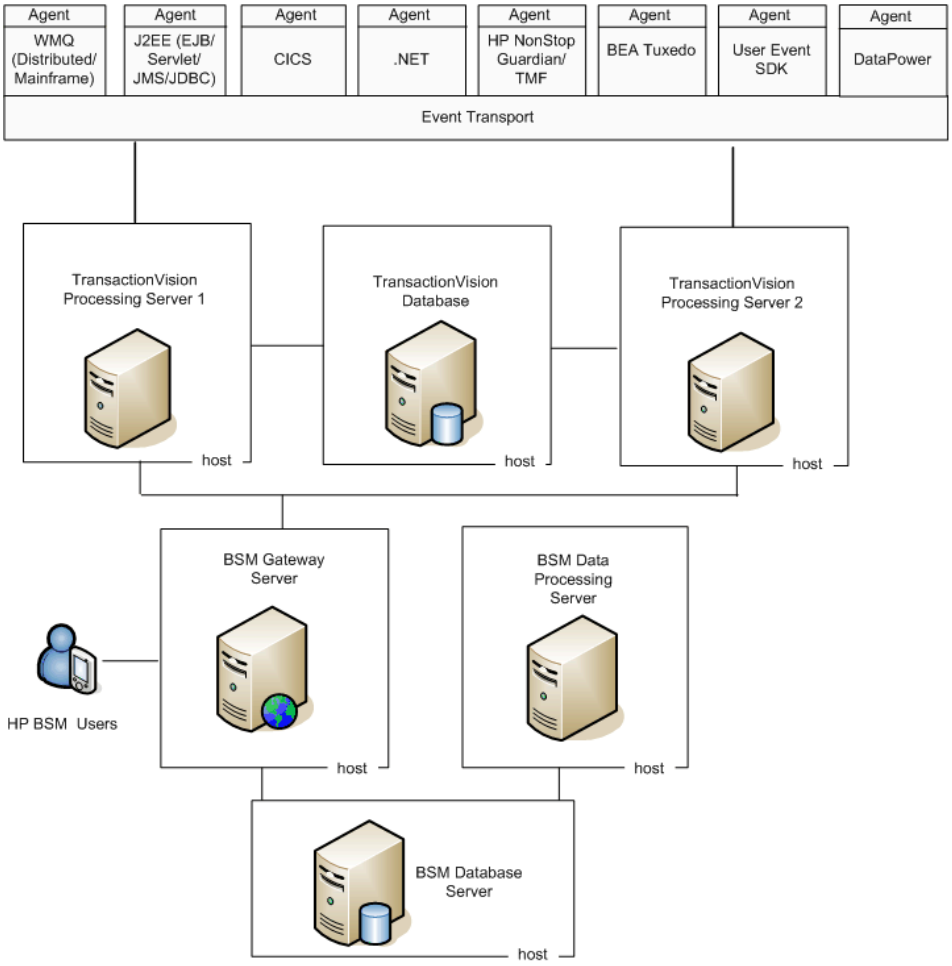


Note: Each TransactionVision Processing Server contains an embedded database that can be used in place of the TransactionVision database for POC or other testing purposes.

Two-Machine Deployment

A two-machine deployment has the BSM Gateway Server installed on one machine and the BSM Data Processing Server on a second machine.

In this environment, multiple TransactionVision Processing Servers are typically used.



The database used by TransactionVision is separate from the database used by BSM, and must be installed on a host that is accessible from every TransactionVision Processing Server.

Note: A two-machine Business Service Management deployment can have either a standard or enterprise configuration. These configurations differ in terms of the memory and CPU required. See the *HP Business Service Management Deployment Guide* PDF.

TransactionVision has its own memory and CPU requirements. See Chapter 3, "Reviewing System Requirements."

A Closer Look at the TransactionVision Processing Server

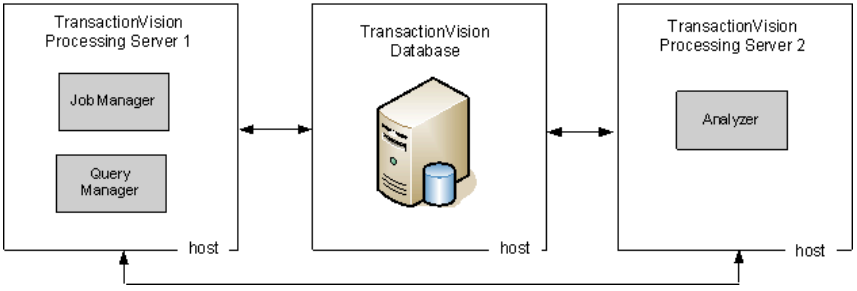
When the deployment environment includes multiple Processing Servers, you have several options in how the Analyzer instances, Job Manager, and Query Manger components are placed.

The guidelines for the deployment environment are:

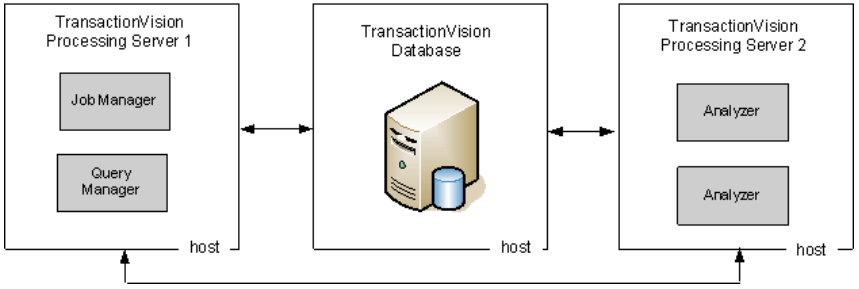
- The TransactionVision deployment environment must have at least one Processing Server.
- The TransactionVision deployment environment must have at least one Analyzer, one Job Manager, and one Query Engine. Each of these runs in the context of a Processing Server.
- The TransactionVision deployment environment must have at least one database configured to store transaction and event data collected by the agents.

Some possible configurations are shown below.

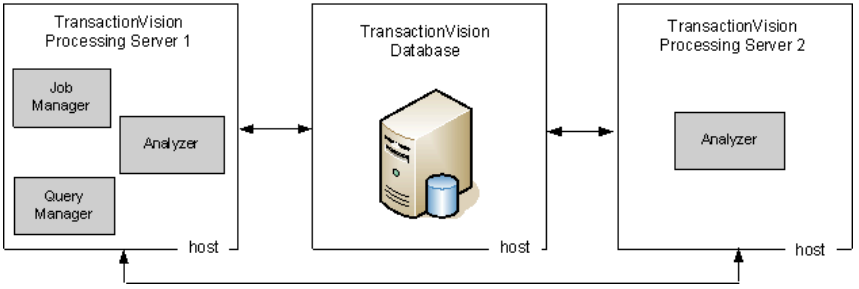
➤ One Processing Server dedicated to the Job and Query Managers and one Processing Server dedicated to the Analyzer:



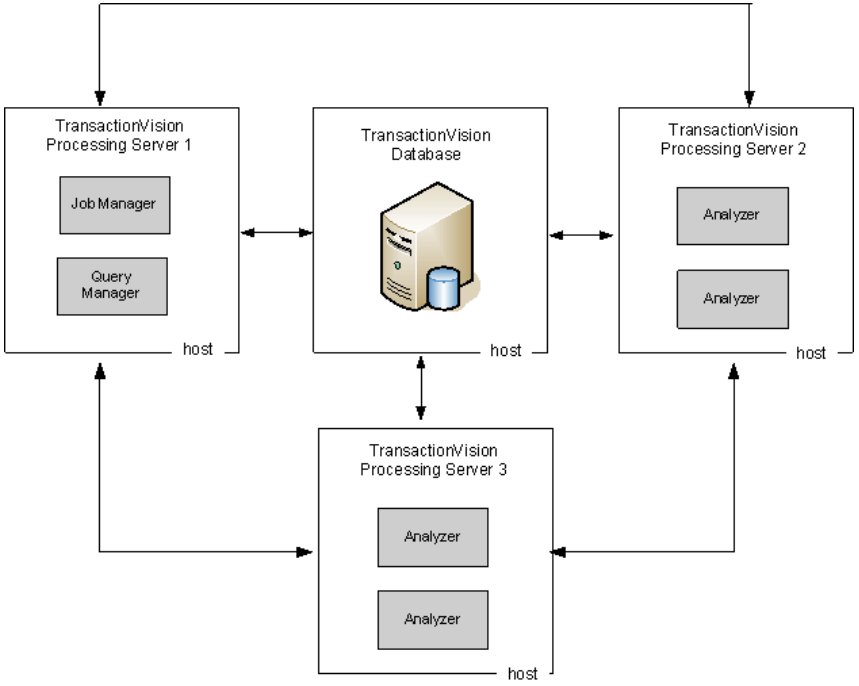
➤ One Processing Server dedicated to the Job and Query Managers, one Processing Server dedicated to two instances of the Analyzer



➤ One Processing Server for the Job Manager, the Query Manager, and one instance of the Analyzer, one Processing Server for another Analyzer:



➤ One Processing Server Dedicated to the Job and Query Managers, the remaining two Processing Servers each running two Analyzers.



2

TransactionVision Deployment Planning

This chapter includes:

Installation Packages

The TransactionVision Processing Servers and Agents are installed separately from the Business Service Management components.

The TransactionVision components are in packages that are specific to a platform. The installation instructions in this guide indicate 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. See Chapter 3, "Reviewing System Requirements."

Compatibility Matrixes

HP BSM and the TransactionVision Processing Server

The following table lists the compatible versions of HP BSM and the most recent TransactionVision Processing Server.

HP BSM	TransactionVision Processing Server
BSM 9.01 plus the Transaction Management 9.02 Update for BSM 9.01	9.02

TransactionVision Processing Server and Agents

The following table lists TransactionVision agent versions compatible with the 9.02 Processing Server, as well as the versions of the Processing Server that are compatible with the 9.02 version of that agent.

TransactionVision Agent	Versions of Agent Compatible with 9.02 Processing Server	Versions of Processing Server Compatible with 9.02 Agent
HP Diagnostics/ TransactionVision Java Agent	8.0x, 9.00, 9.01, 9.02	9.02[1]
HP Diagnostics/ TransactionVision .NET Agent	8.0x, 9.00, 9.01, 9.02	9.02[1]
WebSphere MQ Agent	8.0x, 9.00, 9.01, 9.02	9.02[1]
DataPower Agent	9.02	9.02
CICS, WMQ Batch, WMQ CICS, WMQ IMS, and IMS Bridge Agents on z/OS	8.0x, 9.00, 9.01, 9.02	9.02[1]
BEA Tuxedo Agent	8.0x, 9.00	N/A
NonStop TMF Agent	8.00	N/A

[1] If you require use of this 9.02 agent with an older Processing Server/ Analyzer, contact HP TransactionVision Support for potential product compatibility/incompatibility details.

Sizing and Tuning

The *TransactionVision Planning Guide* contains important information for sizing and tuning new installations of TransactionVision.

This guide is available by download from the HP Software Product Manuals site. See "Documentation Updates" on page 4.

3

Reviewing System Requirements

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

Note: The HP Business Service Management readme file contains additional system requirements. For information about how to access the readme file, see "TransactionVision and Transaction Management Documentation" on page 15.

This chapter includes:

➤ LDAP Support on page 48

➤ Java Support on page 49

➤ Flash Player Support on page 49

➤ Localization and I18N Support on page 49

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

Supported Processing Server Platforms

Operating Environment	WebSphere MQ	TIBCO EMS	SonicMQ
Highly recommended for enterprise deployment: ➤ Microsoft Windows Server 2008 Enterprise Edition SP2 ➤ Microsoft Windows Server 2008 Enterprise Edition SP2 (64bit) Also supported: ➤ Microsoft Windows Server 2008 Enterprise Edition R2 (64bit) ➤ Microsoft Windows Server 2003 Enterprise Edition (64bit) ➤ Microsoft Windows Server 2003 Enterprise Edition SP1 or later (32bit)	6.0, 7.0	5.1.2	7.6.2
Highly recommended for enterprise deployment: ➤ RedHat Enterprise Linux 4.x, 5.x x86 64-bit Also supported: ➤ RedHat Enterprise Linux 4.x, 5.x x86 32-bit	6.0, 7.0	5.1.2	7.6.2

Running the Processing Server on virtual platforms is supported. For details, see "Supported Virtual Platforms" on page 48.

Supported Database Management Systems

The following database management systems are supported by the TransactionVision Processing Server:

DBMS Servers
DB2 9.1, 9.5
Oracle 10g, RAC 10g, 11g
Microsoft SQL Server 2005, 2008

These database server configurations can be accessed remotely through the DataDirect Connect ODBC drivers included with TransactionVision. You do not need to install vendor-specific database client software on the Processing Server host.

It is recommended that the latest Fix Pack for your database product also be installed.

Supported Messaging Middleware Providers

Agent	WebSphere MQ	Transaction - Vision SonicMQ[1]	SonicMQ[1]	HTTP	WebLogic JMS and TIBCO EMS
Java Agent (EJB, Servlet, JMS and JDBCs)	✓	✓	✓		✓[2]
.NET Agent	✓	✓	✓		
WebSphere MQ Agent	✓				
z/OS CICS Agent	✓				
z/OS IMS Bridge Agent	✓				
z/OS WMQ Batch Agent	✓				
z/OS WMQ CICS Agent	✓				
z/OS WMQ IMS Agent	✓				
Tuxedo Agent				✓	
NonStop TMF Agent				✓	
DataPower Agent				✓	

[1] TransactionVision SonicMQ refers to the version of SonicMQ that is included with the TransactionVision Processing Server installation. SonicMQ refers to a version of the SonicMQ software that is acquired and installed separately from TransactionVision.

[2] WebLogic JMS and TIBCO EMS are supported for 8.0x Java Agents only.

Supported WebSphere MQ Agent Platforms

Platform	Operating Environment	WebSphere MQ	Supports WMQ API Exit Agent
Microsoft Windows	Windows Server 2003 x86 and x64 Editions (32bit and 64bit)	6.0, 7.0 [1, 6]	Yes
	Windows Server 2008 x86 and x64 Editions (32bit and 64bit) Windows Server 2008 x86 and x64 Editions R2 (32bit and 64bit)	7.0 [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)	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 4.x, 5.x x86 32 and 64-bit	6.0, 7.0 [1, 6] (32-bit), 6.0, 7.0 [1, 6] (64-bit)	Yes
IBM i5/OS [2]	i5/OS V5R4 iSeries	6.0, 7.0 [1, 6]	Yes
IBM z/OS [3, 5]	z/OS 1.8, 1.9, 1.10, 1.11, 1.12 CICS TS 2.x, 3.1, 3.2, 4.1 IMS 7.x, 8.x, 9.x	6.0, 7.0 [1]	N/A

[1] TransactionVision does support WMQ applications running against WMQ 7. However, events will not be generated from new WMQ 7 APIs added in that release. Applications that only use WMQ 6 APIs are fully supported whether they are run against WMQ 6 or 7.

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

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

[4] **Caution**: 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 Agent and the Analyzer.

Supported Java Agent Platforms

Platforms	Operating Environments
X86/x64	➤ Windows Server 2003 x86 and x64 Editions (32bit and 64bit) ➤ Windows Server 2008 SP1 ➤ Windows Server 2008 SP2, (32bit and 64bit)
X86/x64	RedHat Enterprise Linux 4.x, 5.x x86 32 and 64-bit
SPARC	Solaris 9, 10
POWER	AIX 5L 5.3, 6.1
PA-RISC/Itanium	HP-UX 11i v2, v3
zSeries	z/OS 1.8, 1.9, 1.10, 1.11, 1.12

Running Java Agent on virtual platforms is supported. For details, see "Supported Virtual Platforms" on page 48.

Supported Java Agent Technologies

Application Servers
IBM WebSphere Application Server 6.0[1], 6.1[2,3,4], 7
BEA WebLogic Application Server 9.2.3, 10, 11g
JBoss Application Server 5.1[5]
Tomcat 5.5, 6.0

[1]TransactionVision 9.0x requires RefreshPack and FixPack version 6.0.2.29 or higher when using WebSphere Application Server 6.0 to eliminate a JMX-related problem with prior versions.

[2]TransactionVision 9.0x does not support IBM WebSphere Application Server Community Edition.

[3]TransactionVision 9.0x requires FixPack 9 or higher when using WebSphere Application Server 6.1 to eliminate performance problems seen with prior versions.

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

[5]Java Agent support for JBoss is limited to collection of Servlet, JDBC and EJB 3 events. Support for collection of JMS events will be added in a future product version.

JMS Providers
WebSphere MQ 6.0[1], 7
TIBCO EMS 4.2.0, 4.4.2, 5.1
SonicMQ 7.6.2
WebLogic JMS 8.1.6, 9.2.3, 10

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

JDBC Providers
Oracle 9.2, 10g, 10g RAC, 11g, 11g RAC DB2 9.1, 9.5 SQL Server 2005, 2008

Supported CICS Agent Platforms

Platform	Operating Environment	WebSphere MQ
zSeries	z/OS 1.8, 1.9, 1.10, 1.11, 1.12 CICS TS 2.x, 3.1, 3.2,, 4.1[1, 2]	6.0, 7.0

[1] To use the z/OS CICS WMQ Agent with CICS TS 3.2 you must apply IBM PTF UK60039.

[2] To run TransactionVision with CICS TS 4.1, you must apply PTF UK59997.

Supported BEA Tuxedo Agent 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 Agent 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 Server 2003 x86 and x64 Editions (32bit and 64bit) ➤ Windows Server 2008 SP2 (32bit and 64bit) ➤ Windows Server 2008 R2 (32bit and 64bit)	1.1, 2.0, 3.0, 3.5, 4.0

Supported IBM WebSphere DataPower Platforms

IBM WebSphere DataPower Appliance	Firmware Version
WebSphere DataPower Integration Appliance XI50	3.7.1.4 or later

Supported Virtual Platforms

Running the Processing Server and Java Agent on virtual platforms is supported.

If you are deploying TransactionVision on a virtual platform, the sizing guidelines for a regular installation are not applicable. The following general limitations and recommendations are applicable to an installation on a virtual machine:

- VMware ESX 4.x virtualization platform is supported.
- Performance of TransactionVision on a virtual machine can be expected to be slower than in a regular installation.
- TransactionVision capacities and performance vary according to the various server resources, such as CPU, memory, and network bandwidth, allocated to TransactionVision components.
- It is highly recommended that you use dedicated hardware for the Processing Server and database server in production environments where performance is a concern.
- It is strongly recommended that you do not run a database server containing TransactionVision databases on a virtual machine if the database files reside on a virtual disk.
- Use a Gigabit network card.

Supported Browser Configurations

The browser configurations supported by the TransactionVision application are the same as those supported by BSM. See the *HP Business Service Management Deployment Guide* PDF.

LDAP Support

TransactionVision LDAP support is managed by HP Business Service Management. See the *HP Business Service Management Hardening Guide* PDF.

Java Support

TransactionVision includes the Java 1.6 Runtime Environment which the TransactionVision Analyzer uses.

Supported JVMs for the JMS, JDBC, Servlet and EJB Agents match those versions distributed with the supported WebSphere Application Server and WebLogic Application Server.

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 Service Management Deployment Guide* PDF for version information.

Localization and I18N Support

TransactionVision 9.0x is not localized.

Part II

Processing Server Installation

4

Preparing to Install the TransactionVision Processing Server

This chapter includes:

About the TransactionVision Processing Server

After you install and configure a Processing Server on a host, it can contain any of the following processes:

- Up to five Analyzers
- A primary or backup Job Manager
- A primary or backup Query Engine
- The SonicMQ Broker

The process runs as a Windows service on Windows and as a daemon on Linux.

Each process runs on a specified port. See the default port assignments in the "Troubleshooting and Limitations" section of "Processing Servers" in *Using Transaction Management*.

Though normally managed through the Administration user interface, you can also initiate service shutdown or get status information by using a command-line utility. See **AnalyzerManager** in the "Administration Utilities" chapter in *Using Transaction Management*.

The host on which the TransactionVision Processing Server is installed must have a static IP address. When the Processing Server is registered to a BSM Gateway Server, its hostname is resolved to its underlying IP address which is then used as a unique identifier for the Processing Server.

Overview of the Processing Server Installation and Configuration

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

1 Locate the host on which the Processing Server is to be installed. This host needs access to the TransactionVision database, which can optionally be on the same host.

In most deployment environments, the TransactionVision Processing Server is installed on a separate host from the Business Service Management Gateway Server.

2 Review the system requirements for the Processing Server. See "Supported Processing Server Platforms" on page 39.

3 Install the Processing Server.

For installing on a host running a Windows operating system, see "Installing the Processing Server" on page 57.

For installing on a host running a Linux operating system, see "Installing the Processing Server on UNIX Platforms" on page 57.

Make a note of the installation folder that you specify during installation as you need to access this location for log files and SonicMQ tools among other things. The default installation directories, referred to as <TVISION_HOME>, are as follows:

Windows
C:\Program Files\HP\TransactionVision

Linux:
/opt/HP/TransactionVision

4 (Optional) If you are not using the SonicMQ product that is bundled with TransactionVision, install a supported Messaging Middleware product.

Note: Be sure that the Messaging Middleware product that you install is supported by the agent types in your deployment environment. See "Supported Messaging Middleware Providers" on page 41.

---

5 Set up the database.

See the "Configuring Databases" chapter.

6 Configure the Processing Server through the Transaction Management Administration pages. See "How to Create a Processing Server" in *Using Transaction Management*.

5

Installing the Processing Server

This chapter provides detailed instructions for a first time installation of the Processing Server on either a Windows or Linux machine. For upgrading prior versions of the Processing Server, see Chapter 24, "Upgrading TransactionVision."

This chapter includes:

Installing the Processing Server on Windows

You can get the Processing Server installation file for Windows (**HPTVProcServer_<version>_win.exe**) from the product disks or for patch releases, from the HP Software Support Online (SSO) portal (you must have an HP Portal account).

The following steps provide detailed instructions for a first time installation of the Processing Server on a Windows machine.

To install a new Processing Server on a Windows host:

1. Ensure that you are logged into the target system either as Administrator or as a user with Administrator privileges.
2. Close all Windows programs currently running on your computer, including automatic backup programs. Antivirus, antispyware, and threat protection programs.

3. Download the Processing Server installation file **HPTVProcServer_<version>_win.exe** to the machine on which you want to install the Processing Server in one of the following ways:

   - For a major release, copy the Processing Server installation file from the **Data_Collectors_and_Components/Components/TV** directory on the product install disks.
   - For a patch or minor release, download the Processing Server installation file from the SSO portal using your HP Passport login at:

     http://support.openview.hp.com/selfsolve/patches

     Select **TransactionVision** in the Product field, <***patch_number***> for Product Version, **Windows** for Operating system, and click **Search**.

4. Double-click **HPTVProcServer_<version>_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\HP\TransactionVision**), or click **Change** to select the desired folder and click **Next** to continue.
7. On the Setup Welcome screen, click **Next** to display the User Information screen.
8. Enter your name and company name, then click **Next**. The Destination Location screen appears.
9. 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.
10. Click **Finish** to complete the installation.
11. Run **<TVISION_HOME>\bin\SupervisorStart.bat**.

Installing the Processing Server on Linux

You can get the Processing Server installation file for Linux (**HPTVProcServer_<version>_linux.exe**) from the product disks or for patch releases, from the HP Software Support Online (SSO) portal (you must have an HP Portal account).

The following steps provide detailed instructions for a first time installation of the Processing Server on a Linux machine.

To install a new Processing Server on Linux platforms:

1 Download the Processing Server installation file **HPTVProcServer_<version>_linux.tgz** to a temporary directory on the machine on which you want to install the Processing Server, in one of the following ways:

- For a major release, copy the Processing Server installation file from the **Data_Collectors_and_Components/Components/TV** directory on the product install disks.
- For a patch or minor release, download the Processing Server installation file from the SSO portal at http://support.openview.hp.com/selfsolve/patches, using your HP Passport login. Select **TransactionVision** in the Product field, <***patch_number***> for Product Version, **Linux** for Operating system, and click **Search**.

Note: If you cannot download this file directly to the Linux machine on which you are installing the Processing Server, make sure that you download the file to a machine from which you can later FTP (in binary mode) the file to the Linux machine.

---

2 Log in as superuser:

```
su
```

3 Change to the directory of the downloaded TransactionVision installation files.

4 Unzip and untar the Linux package for your platform. For example:

```
gunzip HPTVProcServer_<version>_linux.tgz
tar xvf HPTVProcServer_<version>_linux.tar
```

5 Enter the following command to begin the installation procedure:

./tvinstall_<version>_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 Processing Server

99. All of above
q. Quit install

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

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

---

6 Type **1** and press **Enter**.

The following prompt is displayed:

Please enter the user name to use for running TransactionVision processes [root]:

7 To run the Processing Server as a different user than **root**, enter that user name.

This sets appropriate permissions on the installed files for that user.

The installation script installs the package at **opt/HP/TransactionVision**, and displays the following messages:

```
...
Package tvision-sonicmq was successfully installed
Package tvision-analyzer was successfully installed
```

The TransactionVision component menu is displayed.

8 Type **q** and press **Enter** to quit the installation process.

9 To start the Processing Server, run:

<TVISION_HOME>/bin/run_topaz 'start'

Uninstalling the Processing Server From a Windows Host

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

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

3 Select the HP TransactionVision Processing Server program and click **Change/Remove**.

4 When prompted, press **OK** to automatically shut down all TransactionVision processes and continue with the uninstall, or press **Cancel** to postpone the uninstall to a later time.

5 When the maintenance menu screen appears, select **Remove** and click **Next** to remove TransactionVision components.

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

Uninstalling the Processing Server From a Linux Host

To uninstall TransactionVision components, perform the following steps:

1 Log in as superuser:

```
su
```

2 Change to the directory where you unzipped and untarred the TransactionVision installation files.

If you do not have these files, download the installation file again and unzip/untar it. For details on downloading the installation file, see "Installing the Processing Server on Linux" on page 59.

3 Enter the following command:

./tvinstall_<version>_unix.sh -u

The following menu is displayed (actual options and numbers depend on the TransactionVision components that are installed):

```
This script will uninstall TransactionVision components.

The following TransactionVision packages are installed on the system:

1. TransactionVision Processing Server

99. All of above
q. Quit install

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

4 During the uninstall of the Processing Server, all TransactionVision processes are automatically stopped. If you want to postpone the uninstall to a later time, type **q** and then press **Enter**. Otherwise, type **1** and press **Enter** to uninstall only the Processing Server, or **99** and press **Enter** to uninstall all TransactionVision components.

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

6 Enter **1** and press **Enter**.

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

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

7 Type **q** and press **Enter** to quit the installation.

6

Configuring Databases

This chapter includes:

About Configuring Databases

After you install the TransactionVision Processing Server, you access the Transaction Management application to configure one or more instances of the Processing Server.

That configuration assumes that the TransactionVision database has been created and configured as described in this chapter. This configuration differs for DB2, Oracle, and SQL Server databases.

For POC environments, you can instead use the built-in database. You specify this database when you configure a Processing Server. No other configuration is needed.

Note: The built-in database does not perform well with large amounts of data. For a reasonable performance, the number of events should not exceed more than 100,000 - 200,000 events.

For more details, see "How to Create a Processing Server" in *Using Transaction Management*.

Supported Databases

TransactionVision Processing Server supports IBM DB2, Oracle, and Microsoft SQL Server database management systems. For the supported versions, see Supported Database Management Systems on page 40.

Controlling Database Access

In the deployment environment, TransactionVision connects to the database via JDBC. During Processing Server configuration, you are prompted for the database type (Oracle, DB2, SQL Server), host name, database name, database port, user name, and password.

These values are saved as part of the Processing Server configuration and are used for establishing each JDBC connection to the database. The user password is stored in encrypted form.

Setting DB2 Variables

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

Variable	Minimum Value	Description
APP CTL HEAP SZ	1024	Maximum application control heap size. This value indicates the number of 4KB blocks.
APPLHEAPSZ	1024	Default application heap size. this value indicates the number of 4KB blocks.

Following is an example of using DB2 commands to set these values for a database named **tvision**. Note that the last three commands 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 APPLHEAPSZ 1024
db2 force application all
db2stop
db2start
```

Setting Oracle Variables

If TransactionVision is 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 reports and topologies, several Processing Servers will be in use, or where the Processing Server 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 appear in the Processing Server logs if this limit is exceeded. To increase the number of Open Cursors, run the following command:

alter system set open_cursors = 600

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

Configuring TransactionVision to Access the Oracle RAC Database

This section describes how to configure TransactionVision to access the Oracle 10g RAC and 11g RAC databases.

Accessing the Oracle 10g RAC Database

To access an Oracle 10g RAC database, the TransactionVision database configuration must be manually configured.

To configure the TransactionVision database:

1. Copy the **tnsnames.ora** file (contains the RAC configuration from the Oracle server), to the machine or machines running the TransactionVision Analyzer and Transaction Management user interface.
2. On the **Database** tab for the Processing Server configuration, select **Use custom JDBC driver** and specify the following:
   - **JDBC connection URL:**
     jdbc:mercury:oracle:TNSNamesFile=C:\\db\\tnsnames.ora;TNSServerName=TVDB

     TNSNamesFile should point to the tnsnames.ora file that has been copied from the Oracle server.

     TNSServerName should contain the RAC instance name.
   - **JDBC driver class:**
     com.mercury.jdbc.oracle.OracleDriver

Example: tnsnames.ora file for an RAC configuration with two Oracle instances:

```
TVDB =
 (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = labm1db15-vip)(PORT = 1521))
   (ADDRESS = (PROTOCOL = TCP)(HOST = labm1db16-vip)(PORT = 1521))
   (LOAD_BALANCE = yes)
   (CONNECT_DATA =
     (SERVER = DEDICATED)
     (SERVICE_NAME = TVDB)
        (failover_mode=(type=select)(method=basic))
   )
 )
```

Accessing the Oracle 11g RAC Database

To connect to an Oracle 11g RAC database, you can simply use the SCAN (Single Client Access Name) as the database name. You do not need to use the custom JDBC driver configuration with the **tnsnames.ora** file.

DBMS Performance Tuning

Since 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.

This section includes the following:

- "Optimizing I/O Throughput" on page 67
- "Testing DBMS and Diagnosing Performance Bottlenecks " on page 68
- "Updating Database Statistics" on page 70

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 Analyzers - if the data can be partitioned then separate DBMS instances on different hosts can be employed to achieve parallelism.
- 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 64 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 Processing Server will be installed. For more information about these tools, see "Command-Line Utilities" in *Using Transaction Management*.

The tools simulate a specific 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 Processing Server 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 Processing Server host and the DBMS host are different, the above tests should be run on the Processing Server 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 Processing Server.

It could result in large performance gains in queries made by TransactionVision views and reports, as well as queries made internally by the TransactionVision Processing Server 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" in "Command-Line Utilities" in *Using Transaction Management*.

DBMS Disk Space Requirements

Another factor to be determined 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 SQL Server to Support Queries with User Data Buffer Criterion

By default the User Data Buffer query criterion is not supported with SQL Server. If the deployment environment requires queries that include this criterion, configure the database as described below.

Note: It is recommended that you configure the SQL Server database before the Analyzer schema is created. If the Analyzer schema already exists, it needs to be recreated and all existing data will be lost.

Changing the SQL Server Database if Analyzers Have Not Been Created

If you have not yet created any Analyzers in the TransactionVision user interface, follow these steps.

To change the SQL Server database before creating Analyzers:

1. Open the **<TVISION_HOME>/config/datamgr/DatabaseDef.xml** file on the Processing Server in a text editor.
2. Search for the following line in the table definition for table **EVENT**:

   <Column name="event_data" type="CLOB" size="1M"/>
3. Replace the line with:

   <Column name="event_data" type="VARCHAR" size="MAX"/>
4. Search for the following line in the table definition for table **USER_DATA**:

   <Column name="user_data" type="BLOB" size="10M"/>
5. Replace the line with:

   <Column name="user_data" type="VARBINARY" size="MAX"/>

Now when you create new Analyzers in the user interface, the Analyzer schema is automatically created with the correct data type and User Data Buffer search is supported.

Changing the SQL Server Database if Analyzers Have Already Been Created

If you have already created Analyzers in the TransactionVision user interface, follow these steps.

To change the SQL Server database after creating Analyzers:

1. Before changing the SQL Server database, from the HP Business Service Management user interface, stop the event collection if it is currently processing for all Analyzers:
   a. Select **Admin** > **Transaction Management**.
   b. (left pane) Click the Configuration tab and expand the Processing Server to display the Analyzer or Analyzers.

c For each Analyzer, select the Status tab > General tab, and click the **Force Stop Analyzer**  icon on the Analyzer Status page.

The analyzer status should state: **The Analyzer status is currently stopped.**

2 Perform steps 1 through 5 in Changing the SQL Server Database if Analyzers Have Not Been Created on page 74.

3 On the command line, execute the following two commands for each Analyzer schema:

CreateSqlScript -d -e -s *schemaname*

CreateSqlScript -c -e -s *schemaname*

These steps drop and recreate the Analyzer schema with the correct data types to support User Data Buffer Search. Note that all existing data will be lost.

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.

This section includes the following:

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 **Unicode Database** property in the Processing Server Database configuration page 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. This property is also set on the Processing Server Database configuration page.

Using Table Partitioning

In production systems that have large data volumes and need peak performance, table partitioning can help to improve database access performance and simplify data management.

Table partitions can be managed individually and allow you to purge large amounts of data much more efficiently than the standard data cleanup based on row deletion.

Each of the TransactionVision tables has a TIMESTAMP column that can be used to range-partition the data stored in the database. You can either create and manage the partitions with third-party tools available for the database product, or use utilities included in TransactionVision which can assist with creating, adding, and dropping partitions.

This section includes:

Creating Tables With Range Partitioning

TransactionVision tables intended for use with table partitioning must be created manually by a DBA by using the **CreateSqlScript** utility as described below. Tables that are not intended for partitioning are created automatically when an Analyzer is added to the deployment environment through the Analyzer wizard.

The **CreateSqlScript** utility creates the required SQL for the table creation. For example:

```
CreateSqlScript.bat -c -s TRADE -partCount 5 -partStartDate "03/27/2010 4:00 pm"
-partLength 1 -partInterval days -ts TS1
```

This creates an SQL script for the TransactionVision tables with 5 initial partitions in tablespace TS1, the first partition starting at the given start date, and each partition containing data for one day. The initial partition will be named 'PART1', and each consecutive partition 'PARTn'. The corresponding Oracle SQL for the EVENT table will be:

```
CREATE TABLE TRADE.EVENT ( proginst_id NUMBER(19) NOT NULL, sequence_no
INTEGER NOT NULL, event_data CLOB, event_time TIMESTAMP NOT NULL,
CONSTRAINT PK_EVENT PRIMARY KEY (proginst_id, sequence_no) )
PARTITION BY RANGE (event_time) (
PARTITION PART1 VALUES LESS THAN (TIMESTAMP '2010-03-28 16:00:00.00'
TABLESPACE TS1),
PARTITION PART2 VALUES LESS THAN (TIMESTAMP '2010-03-29 16:00:00.00'
TABLESPACE TS1),
PARTITION PART3 VALUES LESS THAN (TIMESTAMP '2010-03-30 16:00:00.00'
TABLESPACE TS1),
PARTITION PART4 VALUES LESS THAN (TIMESTAMP '2010-03-31 16:00:00.00'
TABLESPACE TS1),
PARTITION PART5 VALUES LESS THAN (TIMESTAMP '2010-04-01 16:00:00.00'
TABLESPACE TS1),
);
```

Note that the timestamps in the database are based on GMT, so the start date either has to be specified in GMT time, or the option '-partLocalTime' has to be used.

For information about the CreateSQLScript utility, see "Command-Line Utilities" in *Using Transaction Management*.

Adding and Dropping Partitions

TransactionVision also includes a utility that eases the management of adding and dropping partitions after the initial table creation. Use PartitionUtil to create SQL scripts for adding/dropping partitions after the initial partitions have been created with CreateSqlScript. Example for dropping the first two partitions:

```
PartitionUtil.bat -db oracle -drop -s TRADE -startNo 1 -count 2
```

This will create an SQL script 'drop_partitions.sql' in the current directory. The generated SQL for the EVENT table will be:

```
ALTER TABLE TRADE.EVENT DROP PARTITION PART1 UPDATE INDEXES;
ALTER TABLE TRADE.EVENT DROP PARTITION PART2 UPDATE INDEXES;
```

Example to add two new partitions:

```
PartitionUtil.bat -db oracle -add -s TRADE -startNo 6 -count 2 -length 1 -interval days
-startDate "04/01/2010 4:00 pm" -ts TS1
```

This will create an SQL script 'add_partitions.sql' in the current directory. The generated SQL for the EVENT table will be:

```
ALTER TABLE TRADE.EVENT ADD PARTITION PART6 VALUES LESS THAN
(TIMESTAMP '2010-04-02 16:00:00.00') TABLESPACE TS1;
ALTER TABLE TRADE.EVENT ADD PARTITION PART7 VALUES LESS THAN
(TIMESTAMP '2010-04-03 16:00:00.00') TABLESPACE TS1;
```

For information about the PartitionUtil utility, see "Command-Line Utilities" in *Using Transaction Management*.

Custom XDM Tables

Custom user-defined event or transaction tables can also be managed by the TransactionVision partitioning tools. In order to include a custom XDM table in the table partitioning, it is necessary to define a timestamp column in the table and set the 'partitionCol' attribute of this column to true. Example:

```
    <Column name="event_time" type="TIMESTAMP" description="EventTime"
partitionCol="true">
        <Path>/Event/EventTimeTS</Path>
    </Column>
```

The partition utilities (**CreateSqlScript** and **PartitionUtil**) include all tables for which this attribute is set.

System Model Tables

Note that system model object entries do not have a timestamp associated with them, so the system model tables do not support partitioning.

Although the system model is usually mostly static in size, in monitoring environments that generate a large number of different Program Instance objects the system model tables can potentially grow over time, and a cleanup might be necessary. This can be accomplished by using the system model data purging options for the Analyzer.

For more information about the purging options, see Key Configuration Options for Analyzers in *Using Transaction Management*.

7

Post-Installation Tasks for the TransactionVision Processing Server

This chapter includes:

Deploy the Required Applications to the BSM Server

Before users can use the TransactionVision Processing Server, the Transaction Management administration and application must be enabled in the BSM deployment environment.

Transaction Management is a special application made up of three separate applications:

- TransactionVision
- HP Diagnostics
- End User Management

At least one of these applications must be deployed to the BSM deployment environment to enable the Transaction Management application pages. To enable the Transaction Management administration pages, TransactionVision or HP Diagnostics must be deployed.

To deploy an application, use the following BSM Platform Administration page: **Admin** > **Platform** > **Setup and Maintenance** > **Server Deployment**. You will need BSM Super User or Admin level privileges.

Note: Deploying an application requires a valid license for that application. To add a license or view information about existing licenses, use the following BSM Platform Administration page: **Admin** > **Platform** > **Setup and Maintenance** > **License Management**.

About the TransactionVision Licenses

Licensing for TransactionVision is enabled and managed from the BSM user interface. For information on updating the license key, see "Licenses" in *Platform Administration*.

Permanent TransactionVision license keys are capacity based, based on application instances. An application instance is one of the following TransactionVision-specific object types:

- J2EE Application Servers - J2EE (Servlet, EJB, JMS, JDBC)
- JMS Servers - JMS
- Database Servers - JDBC
- Queue Managers - IBM WebSphere MQ
- CICS Regions - CICS

Capacity is based on the total number of application instances used over the last 24-hour period. For example, one JMS Server and one queue manager would equal a capacity of two application instances.

Capacity is calculated once per hour.

The BSM License Management page (**Admin** > **Platform** > **Setup and Maintenance** > **License Management**) reports:

- Whether the TransactionVision license is valid.
- The TransactionVision consumed license count (application instance count).
- The purchased TransactionVision license capacity (maximum number of application instances).

You install the TransactionVision license from either the BSM License Management page at **Admin** > **Platform** > **Setup** and **Maintenance** > **License Management** or on the License page of the BSM Setup and Database Configuration Utility tool. For details, see "Licenses" in *Platform Administration*.

Note: Initially you are assigned a 60 day time-based evaluation license. This license covers the TransactionVision application and an unlimited number of application instances. Within this evaluation period, you must obtain either a evaluation license extension or a permanent license key.

Set Up Security

After the TransactionVision Processing Server is installed and the Transaction Management administration and application pages are enabled the default user roles are in effect.

You should set up specific users and roles before accessing the Processing Server configuration. See "Managing User Permissions" on page 306.

Add the Processing Server to the TransactionVision Deployment Environment

Each installation of a Processing Server must be mapped to an instance of itself in the TransactionVision Configuration page of the Transaction Management administration pages. Then the user can configure the Processing Server as only minimal configuration was done during installation.

For more information, see "How to Create a Processing Server" in "*Using Transaction Management*.

8

Configuring Analyzer Logging

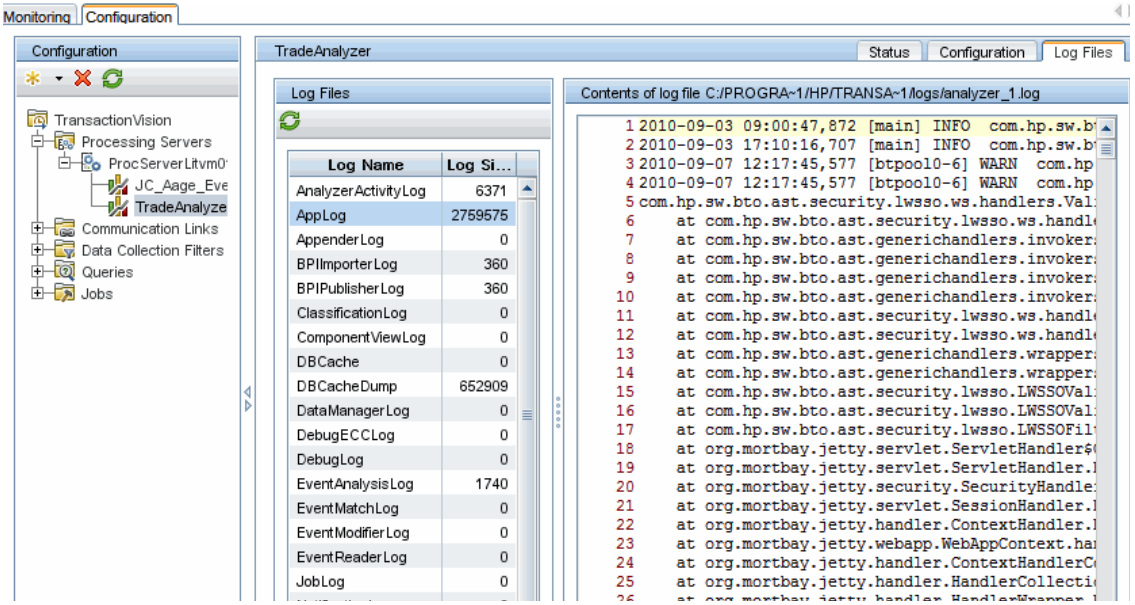
This chapter includes:

Accessing the Main Analyzer Log File

By default, the Analyzer logs error and warning messages to a corresponding log file named **AppLog**. You can view the content of **AppLog** through the Transaction Management Administration pages in BSM as follows:

1. Log into BSM as a user with Transaction Management Administration privileges.
2. Select **Admin** > **Transaction Management** > **Administration** > **Configuration.**
3. Select <my_Analyzer> on the left and then click the **Log Files** tab on the right.
4. Select the **AppLog** entry.

5 View the content of the log file in the Log Files tab:



6 To search for text in the contents of the log file, enter **CTRL-F**.

Note: The Analyzer log file is stored in the **<TVISION_HOME>/logs** directory. Multiple Analyzer log files exist in this directory. The log files are accessed by the names **analyzer_<id>.log** file where **<id>** is a digit from 1-5 corresponding to each possible Analyzer on a Processing Server. For example, in the screen shot above the file **analyzer_2.log** corresponds to the **TradeAnalyze** Analyzer.

Accessing the Backup Logs

By default, an Analyzer uses one or more backup logs. When the main **analyzer_<id>.log** file reaches its maximum size, it is renamed to **analyzer_<id>.log.1** and a new empty **analyzer_<id>.log** file is created.

You can have up to 5 backup log files. The oldest log file content gets removed first. For example the following events take place when **analyzer_1.log** reaches its maximum size and there are two backup logs:

- **analyzer_1.log.2** is removed if it exists.
- **analyzer_1.log.1** is renamed **analyzer_1.log.2**.
- **analyzer_1.log** is renamed **analyzer_1.log.1**.
- A new **analyzer_1.log** is created.

This type of logging is called circular logging. If you do not want to use circular logging, you can change the configuration to use linear logging, in which a single log file is generated. See "Specifying Linear Logging Instead of Circular" on page 89.

Changing the Maximum Size of the Analyzer Log File

By default the maximum size of the AppLog file is 10 MB.

To change the maximum size:

1. Log into BSM as a user with Transaction Management Administration privileges.
2. Select **Transaction Management** > **Administration** > **Configuration.**
3. Select <my_Analyzer> on the left and then click the **Configuration** tab on the right.
4. Click the **XML** tab.
5. Select the **Analyzer.Logging** entry.

6 In the following area of the XML, edit the **MaxFileSize** value as desired:

```
<!-- Appenders -->

 <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
      <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d [%t] - %m%n" />
      </layout>
 </appender>

 <appender name="ANALYZER_LOGFILE"
class="org.apache.log4j.RollingFileAppender">
      <param name="File" value="analyzer.log" />
      <param name="Append" value="true" />
      <param name="MaxBackupIndex" value="2" />
      <param name="MaxFileSize" value="10MB" />
      <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d [%t] %-5p %c %x - %m%n" /
>
      </layout>
 </appender>
```

Changing the Number of Backup Log Files

If the Analyzer is using circular logging, you can specify how many backup files are used. Set the **MaxBackupIndex** value to 2,3,4,or 5 as follows:

```
<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="3"/>
  <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>
```

Specifying Linear Logging Instead of Circular

By default, the Analyzer uses circular logging. You can specify instead to use a single log file.

To specify linear logging instead of circular, follow these steps:

1. Log into BSM as a user with Transaction Management Administration privileges.
2. Select **Transaction Management** > **Administration** > **Configuration.**
3. Select <**my_Analyzer**> on the left and then click the **Configuration** tab on the right.
4. Click the **XML** tab.
5. Select the **Analyzer.Logging** entry.
6. In the following area of the XML, change the **RollingFileAppender** string to **FileAppender**:

```
<!-- Appenders -->

 <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
      <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d [%t] - %m%n" />
      </layout>
 </appender>

 <appender name="ANALYZER_LOGFILE" class="org.apache.log4j.FileAppender">
      <param name="File" value="analyzer.log" />
      <param name="Append" value="true" />
      <param name="MaxBackupIndex" value="2" />
      <param name="MaxFileSize" value="10MB" />
      <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d [%t] %-5p %c %x - %m%n" /
>
      </layout>
 </appender>
```

7 Remove the entries for the **MaxBackupIndex** and **MaxFileSize** parameters:

```
<!-- Appenders -->

 <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
       <layout class="org.apache.log4j.PatternLayout">
             <param name="ConversionPattern" value="%d [%t] - %m%n" />
       </layout>
 </appender>

     <appender name="ANALYZER_LOGFILE" class="org.apache.log4j.FileAppender">
       <param name="File" value="analyzer.log" />
       <param name="Append" value="true" />
       <layout class="org.apache.log4j.PatternLayout">
             <param name="ConversionPattern" value="%d [%t] %-5p %c %x - %m%n" /
>
       </layout>
 </appender>
```

Using Windows and UNIX System Logs

On UNIX and Windows platforms, you can configure TransactionVision Processing Server 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/*/Analyzer.Logging.xml**.

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

This section includes the following:

➤ "Windows Event Appender" on page 91

➤ "UNIX Event Appender" on page 91

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

Agent Installation and Configuration

9

Preparing to Install TransactionVision Agents

This chapter includes:

About Agent Coverage

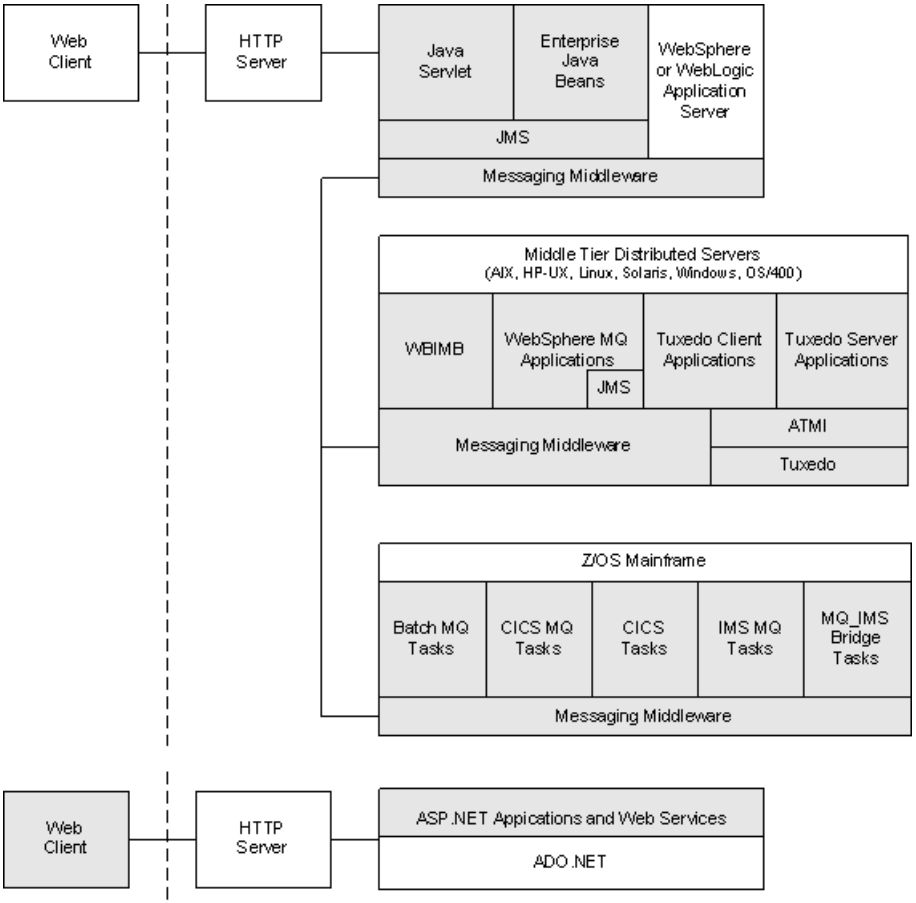
Some TransactionVision agents do not collect event data that supports the full capabilities of Transaction Management topologies and data filtering. All agents support the Transaction Management reports and queries.

The following table summarizes the limitations per agent.

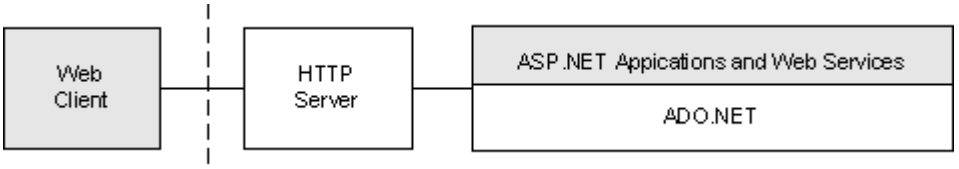
TransactionVision Agent	Data can be Filtered?	Data Used in Transaction Topology?	Data Used in Component Topology?
HP Diagnostics/ TransactionVision Java Agent	✓	✓	✓
HP Diagnostics/ TransactionVision .NET Agent	✓		
WebSphere MQ Agent	✓	✓	✓
DataPower Agent		✓	✓
CICS Agent on z/OS	✓		✓
WMQ Batch, WMQ CICS, WMQ IMS Agents on z/OS	✓	✓	✓
IMS Bridge Agents on z/OS	✓		✓
BEA Tuxedo Agent	✓		
NonStop TMF Agent	✓		

Applications That Can Be Monitored

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

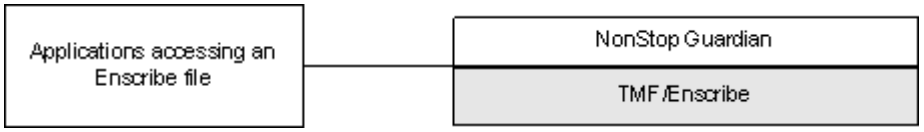


ASP.NET applications can also be monitored:



.NET Remoting and WCF client and server applications can also be monitored.

NonStop TMF applications can also be monitored:



About WebSphere MQ (WMQ) Agents

The WebSphere MQ Agent tracks MQ API calls. These API calls include the entire MQ API set, the major APIs being MQPUT, MQGET, MQCONN, MQDISC, MQOPEN, MQCLOSE, etc.

Distributed (Non-Mainframe) Platforms

There are two types of WebSphere MQ Agents provided by TransactionVision on distributed platforms:

- The **WebSphere MQ Library Agent** intercepts a WebSphere MQ API call by the shared library (or DLL) interception method on distributed platforms. This involves placing the TransactionVision Agent 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 Agent** uses the WebSphere MQ API exit support. This agent 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.

Both of these agents 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.

z/OS Based Agents

There are five TransactionVision Agents which execute in various z/OS environments:

- The CICS Agent collects detail level data about CICS API calls (Program Control, Task Control, File Control, and so forth).
- The WMQ-CICS Agent collects detailed data on WMQ API calls performed by CICS application programs.
- The WMQ Batch Agent collects detail level data on WMQ API calls performed by z/OS batch applications.
- The WMQ IMS Agent collects detail level data on WMQ API calls performed by IMS applications.
- The WMQ IMS Bridge Agent collects detail level data about WMQ API calls executed from within the WMQ IMS Bridge.

About Java Agents

- The **Servlet Agent** tracks servlet methods in a J2EE application server. This agent tracks HTTP calls such as HTTP_POST, HTTP_GET, HTTP_PUT, etc., which result in method calls into the J2EE container. The Servlet agent tracks these method invocations by instrumenting the servlet to collect events at the entry and exit of each call.
- The **JMS Agent** tracks WebSphere MQ Java Message Service or TIBCO EMS events from standalone Java applications as well as from J2EE application servers. This agent 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 Agent** tracks transactions through business logic within a J2EE application server. This agent tracks all public business methods in an entity bean, session bean or message driven bean. In addition to the business methods, this agent tracks the ejbCreate, ejbPostCreate, ejbRemove, ejbLoad, ejbStore and onMessage methods. These methods are instrumented by the agent to collect events at the entry and exit of each call.
- The **JDBC Agent** 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 Agents (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 Agent in a TransactionVision environment. For combined environments, the agent can simultaneously serve as both the Probe and the agent. See Chapter 13, "Installing and Configuring the Java Agent" for details.

About .NET Agent

When used with TransactionVision the .NET Agent traces the following types of .NET events:

- **Web Services**
  - **ASP.NET** (*.asmx) - Client and Server

    To generate events, use the **ASP.NET.points** file.
  - **WCF** (*.svc) - Client and Server

    To generate events, use the **wcf.points** file.
- Database calls executed using **ADO.NET**

  To generate events, use the **ADO.points** file.

- **.NET Remoting** - Client and Server

  To generate .NET remoting events, use the **Remoting.points** file and setup the application for instrumentation as described in "How to Configure Instrumentation for .NET Remoting" in the *HP Diagnostics Installation and Configuration Guide*.

- **MSMQ** - Send and Receive (asynchronous)

  To generate events, use the **Msmq.points** file.

- **HTTP**

  - Client outbound - includes calls to REST services

    To generate events, use the **ASP.NET.points** file.

  - ASP.NET inbound/server (POST, GET, PUT) (*.aspx)

    To generate events for HTTP, use **ASP.NET.points** file.

To turn off event generation remove the relevant points file from scope. For more details, see "About Configuration of the .NET Agent for TransactionVision" in the *HP Diagnostics Installation and Configuration Guide*.

About BEA Tuxedo Agent

The BEA Tuxedo Agent 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 agent 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.

About NonStop TMF Agent

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

About the IBM WebSphere DataPower Agent

TransactionVision monitors DataPower services through the use of a combination of built in DataPower WS-Management event reporting and a specialized DataPower Multi-Protocol Gateway for TransactionVision.

DataPower service activity is monitored and reported by DataPower through WS-Management. The TransactionVision Multi-Protocol Gateway (hptv-wsman-subscriber) lives in its own dedicated DataPower domain (HPTVMonitoring) and is responsible for optionally transforming the WS-Management event to a TransactionVision User Event and delivering the event to the Processing Server via TransactionVision's SonicMQ HTTP listener.

10

Installing the WebSphere MQ Agent on Windows

This chapter includes:

Installing the WebSphere Agent on Windows

The TransactionVision WebSphere MQ Agent is installed as a single package on Windows. This chapter provides instructions for a first time installation of the WebSphere MQ Agent on a Windows machine.

Note: If there is a pre-existing installation of the WebSphere Agent on the host machine, you must follow the instructions for upgrading the agent system instead of these install instructions. For details, see "Upgrading the WebSphere MQ Agent on Windows" on page 348.

To install the WebSphere MQ Agent on a Windows host, 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. Close all Windows programs currently running on your computer, including automatic backup programs. Antivirus, antispyware, and threat protection programs.

3 Download the WebSphere MQ Agent installation file **HPTVWMQAgent_<version>_win.exe** to the machine on which you want to install the WebSphere MQ Agent in one of the following ways:

- Copy the WebSphere MQ installation file from the **Data_Collectors_and_Components/Components/TV** directory on the product install disks.
- For a patch release, download the WebSphere MQ installation file from the SSO portal using your HP Passport login at:

  http://support.openview.hp.com/selfsolve/patches

  Select **TransactionVision**, **<patch_number>** for Product Version, **Windows** for Operating system, and click **Search**.

  Click the appropriate link to download the WebSphere MQ Agent installer for Windows.

4 In the Windows Explorer, double-click **HPTVWMQAgent_<version>_win.exe**. The InstallShield Welcome screen appears.

5 On the Setup Welcome screen, click **Next**. The User Information screen appears.

6 Enter your name and company name, then click **Next**. The Destination Location screen appears.

7 To use the default folder for extracting the installation file, click **Next**. To choose a different folder, click **Change**, select the desired folder, then click **Next**. InstallShield extracts the installation file. The Setup Complete page appears.

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

Uninstalling the WebSphere MQ Agent

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

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

3 Select the HP TransactionVision WebSphere MQ agent 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.

11

Installing the WebSphere MQ Agent on UNIX Platforms

This chapter provides instructions on installing WebSphere MQ Agent on UNIX platforms. To install the WebSphere MQ Agent for Java applications and application servers, see Chapter 13, "Installing and Configuring the Java Agent."

This chapter includes:

Installing the WebSphere MQ Agent on UNIX

This section includes the following:

Installation Files for UNIX Platforms

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

Platform	Files
AIX	HPTVWMQAgent_<version>_aix.tgz
HP-UX	HPTVWMQAgent_<version>_hppa.tgz HPTVWMQAgent_<version>_hpia.tgz
Linux	HPTVWMQAgent_<version>_linux.tgz
Solaris	HPTVWMQAgent_<version>_sol.tgz
IBM i5/OS	HPTVWMQAgent_<version>_i5os.savf

Installation Steps for UNIX Platforms

The WebSphere MQ Agent is installed to the following default directories:

Platform	Intallation Directory
IBM AIX	**/usr/lpp/HP/TransactionVision**
Other UNIX and Linux	**/opt/HP/TransactionVision**

1 Download the WebSphere MQ Agent installation file for the appropriate platform to a temporary directory on the machine on which you want to install the WebSphere MQ Agent, in one of the following ways:

- Copy the WebSphere MQ installation file from the **Data_Collectors_and_Components/Components/TV** directory on the product install disks.
- For a patch release, download the WebSphere MQ installation file from the SSO portal using your HP Passport login at:

  http://support.openview.hp.com/selfsolve/patches

  Select **TransactionVision**, **<patch_number>** for Product Version, **Windows** for Operating system, and click **Search**.

Click the appropriate link to download the WebSphere MQ Agent installer for the appropriate UNIX platform.

2 Log in as superuser:

```
su
```

3 Unzip and untar the packages for your platform; see "Installation Files for UNIX Platforms" on page 112 for the installation file names. For example:

```
gunzip tvue_<version>_aix_power.tar.gz
tar -xvf tvue_<version>_aix_power.tar
```

4 Enter the following command to begin the installation procedure:

./tvinstall_<version>_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 Processing Server
2. TransactionVision WebSphere MQ 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.

---

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**.

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

The installation script installs the specified package(s) to the following appropriate directory, then displays the menu again:

Platform	Intallation Directory
IBM AIX	**/usr/lpp/HP/TransactionVision**
Other UNIX and Linux	**/opt/HP/TransactionVision**

6 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 Agent on AIX

For the WebSphere MQ Agent on the AIX platform, the installation calls the **rebind_sensor** script to relink the Agent 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 for monitored applications to run correctly. For more information about this script, see "Command-Line Utilities" in *Using Transaction Management*.

Uninstalling the WebSphere MQ Agent on UNIX

1 Log in as superuser:

```
su
```

2 Enter the following command:

./tvinstall_<version>_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 Processing Server
2. TransactionVision WebSphere MQ Agent

99. All of above
q. Quit uninstall

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

3 Enter the number associated with the TransactionVision package you want to uninstall.

To uninstall all TransactionVision components, enter **99**.

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

4 To quit the uninstall, enter **q**. If the common package is the only TransactionVision package still installed, it will be uninstalled automatically.

12

Installing Agents on i5/OS

This chapter includes:

Starting the Installation Program on i5/OS

1. If an earlier version of the TransactionVision agent 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 agent installation file **HPTVWMQAgent_<version>_i5os.savf** from the CD_ROM and rename it to **agent<version>.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 HPTVWMQSensor_<version>_i5os.savf sensor<version>.savf

4. On the i5/OS machine, run the following command to install the Agent. Note that you may need to replace **TVTMP** in the command with the name of the library in which the **sensor<version>.savf** package resides:

   RSTLICPGM LICPGM(3RBB9ES) DEV(*SAVF) SAVF(TVTMP/SENSOR<version>)

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 be safely deleted.

13

Installing and Configuring the Java Agent

This chapter provides instructions on installing and configuring the HP Diagnostics/TransactionVision Java Agent on Windows machines, UNIX machines, and z/OS mainframe machines. It also describes how to use a generic installer to install the Java Agent on other platforms.

This chapter includes:

About the Java Agent Installer

The Java Agent combines the capabilities of the Java Agent for TransactionVision and Diagnostics into a single component. The HP Diagnostics/TransactionVision Java Agent installer installs a Java Agent to collect data for either Diagnostics or TransactionVision or both. For more information about installing the Java Agent for Diagnostics, see the *HP Diagnostics Installation and Configuration Guide*.

The Java Agent is the agent for these technologies: JMS, Servlet, JDBC and EJB. It can also be used to collect custom events.

Java Agent Install and Configuration Workflow

The sections that follow present the required tasks for installing and configuring the Java Agent for use in the TransactionVision deployment environment. The tasks are as follows:

- "Task 1: Determine if the Predefined Java Agent Will Collect the Desired Events" on page 121
- "Task 2: Install the Java Agent for Your Platform" on page 121
- "Task 3: Run the JRE Instrumenter for Each Target JRE" on page 141
- "Task 4: Configure the Applications to Be Monitored" on page 149
- "Task 5: Set Up Messaging Queues and Communication Links" on page 150
- "Optional Task: Configuring Custom User Events" on page 151

Task 1: Determine if the Predefined Java Agent Will Collect the Desired Events

Based on its default configuration, the Java Agent can track the following:

- Servlet methods in a J2EE application server—for instance, HTTP requests such as HTTP GET, POST, and PUT, which result in method calls into the J2EE container. These method invocations are tracked by instrumenting the servlet to collect events at the entry and exit of each call.
- JMS from standalone Java applications as well as from J2EE application servers. JMS interface methods such as send, receive, publish, and onMessage. These method names (as well the JMS method names in the previous paragraph) need special formatting (as code). These methods are tracked by instrumenting the JMS library to collect events at the entry and exit of each call.
- EJBs. Transactions through business logic within a J2EE application server. All public business methods in an entity bean, session bean or message driven bean. In addition to the business methods, the Java Agent tracks the ejbCreate, ejbPostCreate, ejbRemove, ejbLoad, ejbStore and onMessage methods. These methods are instrumented by the Agent to collect events at the entry and exit of each call.
- JDBC. SQL calls and transactions made to a relational database through the JDBC API.

Other calls can be tracked by using custom code to create user events. See "Optional Task: Configuring Custom User Events" on page 151.

Task 2: Install the Java Agent for Your Platform

For instructions on installing the Java Agent for your platform, see the appropriate section:

- "Installing and Configuring the Java Agent on Windows" on page 122
- "Installing and Configuring the Java Agent on UNIX" on page 131
- "Installing the Java Agent on a z/OS Mainframe" on page 137
- "Installing the Java Agent Using the Generic Installer" on page 140

Installation Files

The TransactionVision Java Agent runs on several platforms. Each platform has its own installation file:

Platform	Files
Windows	HPDiagTVJavaAgt_<version>_win.exe
AIX	HPDiagTVJavaAgt_<version>_aix.bin
Linux	HPDiagTVJavaAgt_<version>_linux.bin
Solaris	HPDiagTVJavaAgt_<version>_sol.bin
HP-UX	HPDiagTVJavaAgt_<version>_hppa.bin HPDiagTVJavaAgt_<version>_hpia.bin
Other UNIX	HPDiagTVJavaAgt_<version>_unix.tgz
z/OS	HPDiagTVJavaAgt_<version>_zos.tgz

Installing and Configuring the Java Agent on Windows

The following steps provide detailed instructions for a first time installation of 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.

Note: If there is a pre-existing installation of the Java Agent on the host machine, you must follow the instructions for upgrading the agent system instead of these install instructions. For details, see "Upgrading the Java Agent" on page 350.

This section includes:

Launching the Java Agent Installer on Windows

You can install the Java Agent for Windows in one of the following ways:

- For a major release, launch the installer from the product installation disks.
- For a patch or minor release, download the Java Agent installer from the Software Support Online (SSO) portal. You must have an HP Passport account.

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

To launch the installer from the product installation disks:

1. Run the **HPDiagTVJavaAgt_<version>_win.exe** installer in the **\Data_Collectors_and_Components\Components\Diag** directory of the product installation disks.

   You may also copy the Java Agent installer to another location and run it from there.

2. Continue with "Running the Installation on Windows" on page 124.

To launch the Installer from the SSO Portal:

1. Access the SSO portal at http://support.openview.hp.com/selfsolve/patches, using your HP Passport login.
2. Select **TransactionVision** in the Product field, <***patch_number***> for Product Version, **Windows** for Operating system, and click **Search**.
3. Click the appropriate link to download the Java Agent installer for Windows.
4. Continue with "Running the Installation on Windows" on page 124.

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.

Note: If you encounter an error, check to see if the JavaAgent directory already exists and remove it. This could happen if you installed and uninstalled the Java Agent previously without also removing the JavaAgent sub-directory.

Click **Next** to proceed.

3 Review the summary information.

The installation directory and size requirement are listed.

Click **Next** to proceed.

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

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

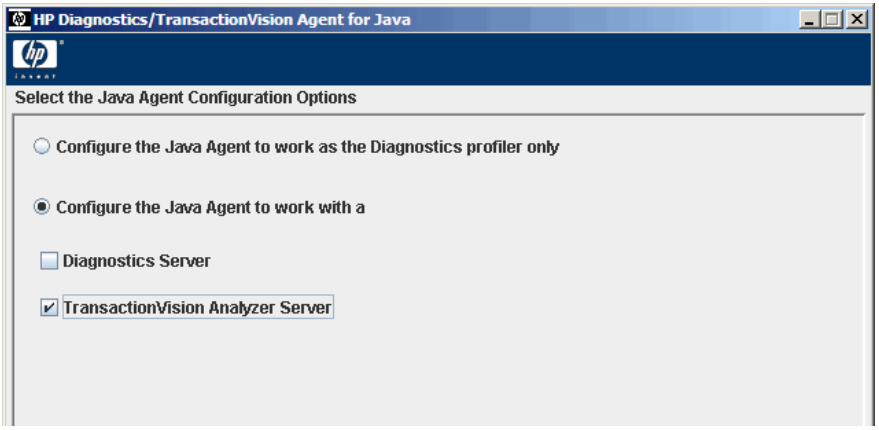
Configuring the Java Agent on Windows

You can configure the Java Agent to work as a TransactionVision Java Agent, a Diagnostics Java Probe, or both. To perform this configuration, you use the Java Agent Setup Module (JASM). JASM allows you to specify the configuration information that enables communication between the agent and TransactionVision Processing Server.

JASM 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**.

To configure the Java Agent to work as a TransactionVision Java Agent:

1 Specify the TransactionVision Analyzer Server option:

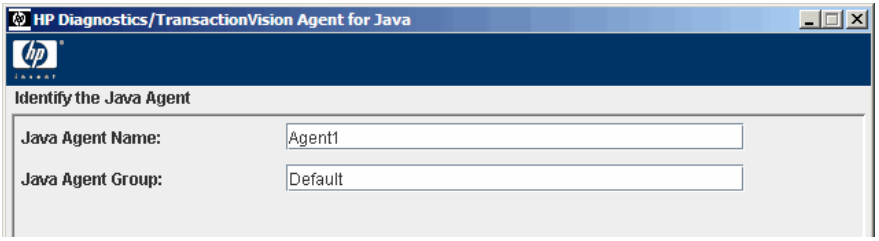


If you are configuring the Java Agent as a Diagnostics Profiler only, see "Configuring the Java Agent as a Profiler Only" in the *HP Diagnostics Installation and Configuration Guide*.

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*.

Click **Next** to proceed.

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

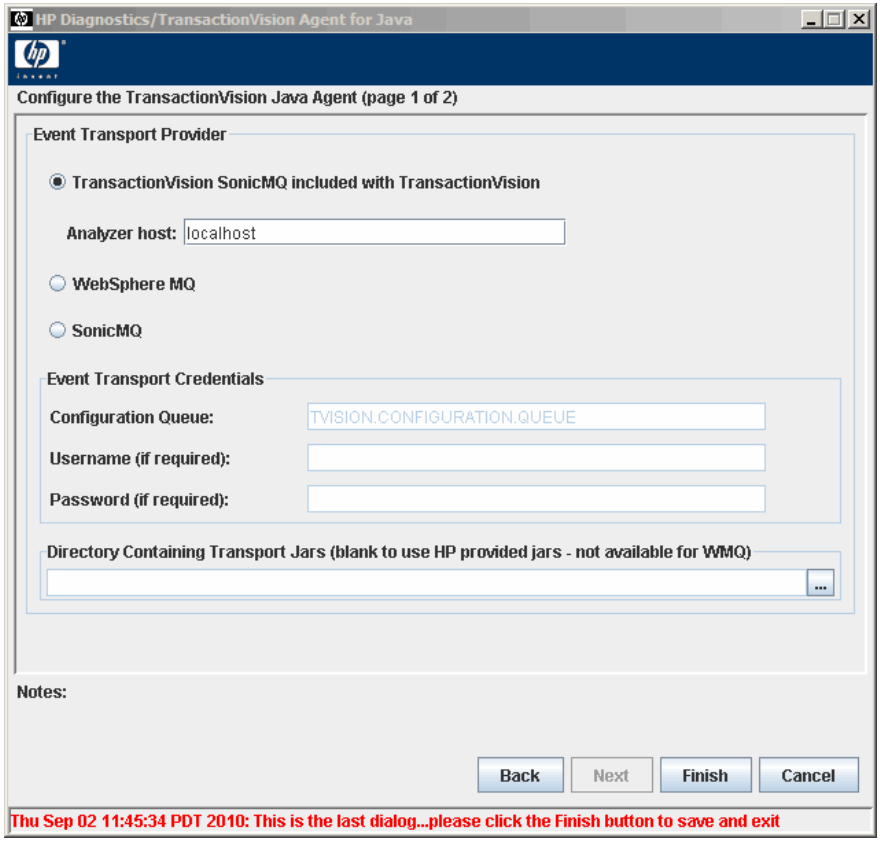


The Java Agent Name is used to uniquely identify each Java Agent. The Java Agent Group is a logical collection of agents reporting to a Diagnostics Server. The Java Agent Group name is case sensitive. If this is a TransactionVision Agent installation only, simply accept the default settings by pressing the Next button to continue with your configuration.

Click **Next** to proceed.

3 Choose the event transport provider and specify the credentials.

The event transport specifies where the Java Agent reads configuration messages sent by the Analyzers.



- **TransactionVision SonicMQ included with TransactionVision**

  To use the SonicMQ event transport provider that is included with TransactionVision, specify the **Analyzer host** name of the Processing Server that contains the Analyzer to which this agent will connect. When using the included SonicMQ, if there is a firewall between the Processing Server and Java Agent, you must open port 21104 and 21111 (or 21112 if SSL is desired) so that the Java Agent can connect to these two ports on the Processing Server host.

- **WebSphere MQ**

  If you choose to use WebSphere MQ as the event transport provider, the transport jars path varies depending on whether Java Agent is used on a WebSphere Application server. For the exact directory paths, see the **Directory containing Transport Jars** bullet below.

- **SonicMQ**

  If you choose to use your own SonicMQ as the event transport provider, you need to fill in the fields that follow. In the **Directory Containing Transport Jars** field, enter the directory path of your SonicMQ JAR files, for example **C:/Sonic/MQ7.5/lib**.

- **Configuration Queue**

  If you are not using the included SonicMQ, enter the configuration queue name for your event transport provider. The default configuration queue name is **TVISION.CONFIGURATION.QUEUE**. However this may have been set to a different value but must match the queue name specified in the Analyzer communication link configuration.

- **User name and Password (if required)**

  Enter the user name and password for the event transport provider if they are required. If you are using the built-in SonicMQ event transport provider, they are not required.

➤ **Directory Containing Transport Jars**

Enter or browse to select any Jar files required by your event transport provider. If you are using the built-in SonicMQ event transport provider, you do not need to specify a directory path.

➤ If you choose to use WebSphere MQ as the event transport provider, and the Java Agent is to be used on a WebSphere Application server V6.1 or V7.0 only, enter the directory path as follows:

For V6.1: **$was.install.root$/lib/WMQ/java/lib**

For V7.0: **$was.install.root$/installedConnectors/wmq.jmsra.rar**

The Java Agent automatically detects the correct **$was.install.root$** at runtime.

➤ If you choose to use the WebSphere MQ as the event transport provider, but the Java Agent will not be used on the WebSphere Application server V6.1 or V7.0, enter the JAR files directory path of your WebSphere MQ Java Client installation:

For AIX: **/usr/mqm/java/lib**

For other UNIX/Linux: **/opt/mqm/java/lib**

For 32-bit Windows: **C:/Program Files/IBM/WebSphere MQ/java/lib**

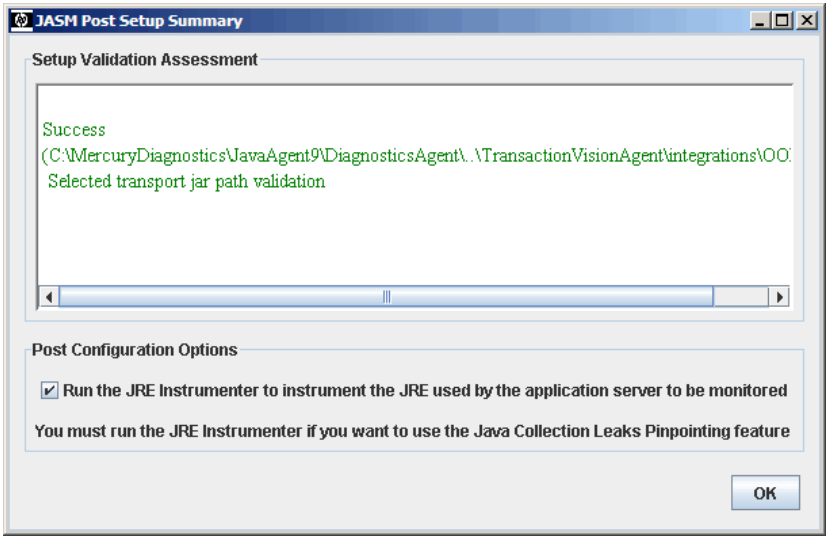
For 64-bit Windows: **C:/Program Files (x86)/IBM/WebSphere MQ/java/lib**

➤ Click **Finish** to continue.

Post Configuration Options

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

The Java Agent Setup Module executes a series of tests to validate and test the configuration settings and presents a Summary dialog:



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 the application to be monitored is using a JRE version 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 "Task 3: Run the JRE Instrumenter for Each Target JRE" on page 141.

Note: If you are running JASM after an initial configuration, your application must be restarted to read the updated configuration.

Installing and Configuring the Java Agent on UNIX

Java Agent installers are 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.

These instructions describe the steps for a **first time** installation of the Java Agent in most UNIX environments using a console mode installation.

You might not be able to use the regular UNIX installers. In this case, use the Generic installer as described in "Installing the Java Agent Using the Generic Installer" on page 140.

Note: If there is a pre-existing installation of the Java Agent on the host machine, you must follow the instructions for upgrading the agent system instead of these install instructions. For details, see "Upgrading the Java Agent" on page 350.

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.

Launching the Java Agent Installer on UNIX

You can download the Java Agent installer from the product disk, or for patch or minor releases, from the HP Software Support Online (SSO) portal (you must have an HP Portal account).

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

To launch the installer from the product installation disks:

1. From the product installation disks **/Data_Collectors_and_Components/Components/Diag** directory, copy the installer **HPDiagTVJavaAgt_<version>_<platform>.bin** to the machine where the TransactionVision Java Agent is to be installed.
2. Continue with "Running the Installation on UNIX" on page 132.

To download the Installer from the SSO Portal:

1 Access the SSO portal at http://support.openview.hp.com/selfsolve/patches, using your HP Passport login.

2 Select **TransactionVision** in the Product field, <***patch_number***> for Product Version, <***UNIX_OS***> for Operating system, and click **Search**.

3 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.

4 Continue with "Running the Installation on UNIX" on page 132.

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:

  ./HPDiagTVJavaAgt_<version>_<platform>.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 +
  export DISPLAY=<hostname>:0.0
  ./HPDiagTVJavaAgt_<version>_<platform>.bin -console

  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 122.

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.

Note: If you encounter an error, check to see if the JavaAgent directory already exists and remove it. This could happen if you installed and uninstalled the Java Agent previously without also removing the Java Agent sub-directory.

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 on UNIX

The following steps configure the Java Agent in UNIX environments. There are two sets of steps: one for graphical mode and one for 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
```

where <**java_agent_install_dir**> refers to the path of your Java Agent installation directory. The default path is **/opt/MercuryDiagnostics/JavaAgent** on UNIX and **/usr/lpp/MercuryDiagnostics/JavaAgent** on AIX.

Configuring the Java Agent on UNIX in Graphical Mode

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

1 Set up the graphical interface.

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

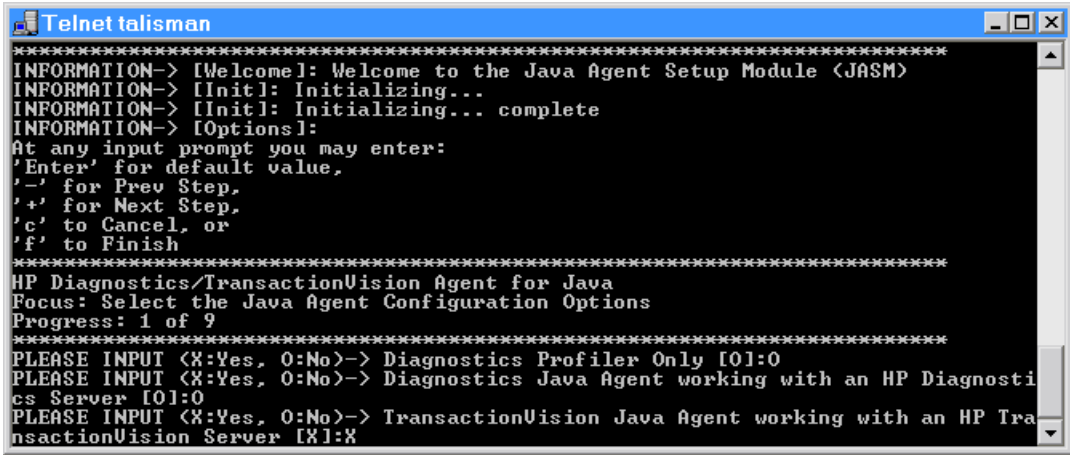
2 Run the Java Agent Setup Module.

```
<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 on Windows" on page 125.

Configuring the Java Agent on UNIX in Console Mode

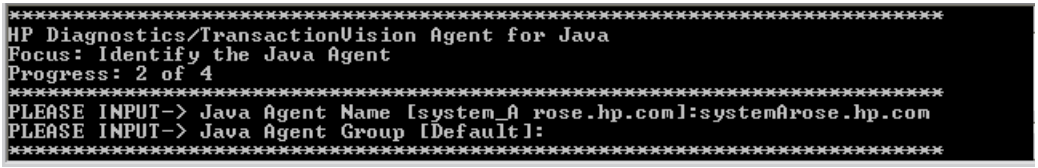
1 Select the TransactionVision Java Agent working with an HP TransactionVision Server option by entering X for this option.



- 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 Agent, enter X for both options and continue to step 2.

Press **Enter** to continue.

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



The Java Agent Name is used to uniquely identify each Java Agent. The Java Agent Group is a logical collection of agents reporting to a Diagnostics Server. The Java Agent Group name is case sensitive.

If this is a TransactionVision Agent installation only, simply accept the default settings by pressing the **Next** button to continue with your configuration.

Press **Enter** to continue.

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

4 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 146.

5 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.

6 Modify the startup script for the application server so that the probe is started together with the monitored application. See "Task 4: Configure the Applications to Be Monitored" on page 149.

7 Verify the Java Agent installation.

Installing the Java Agent on a z/OS Mainframe

This section provides instructions for installing the Java Agent from the .tar file that is included on the product installation disk.

Important Considerations When Installing the Java Agent on z/OS

Consider the following before you install a Java Agent and configure it to be a Java Agent in a z/OS environment:

- Editing property files on a z/OS mainframe

  When installed in a z/OS environment, the agent expects the TransactionVision property files to be in EBCDIC format. Use an EBCDIC editor to update the property files and store the updates in the same format.

- Viewing the System logs in z/OS

  You can view the system log by accessing the primary operator's console in SDSF.

- Capturing metrics on the z/OS mainframe

  Load test metrics are not captured for z/OS. The agent can be configured to capture a limited number of system level metrics for a Diagnostics server. For details on capturing system metrics in z/OS, see "Enabling z/OS System Metrics Capture" in the *HP Diagnostics Installation and Configuration Guide*.

Installing the Java Agent on z/OS From the Installation Disk

A .tar file containing the Java Agent files is included on the product installation disk and can be used to install the Java Agent on a z/OS mainframe.

To install the Java Agent on a z/OS mainframe:

1. Upload **HPDiagTVJavaAgt_<version>_zos.tgz** from the **Diagnostics_Installers** folder on the HP Diagnostics installation disk or from the **Data_Collectors_and_Components/Components/Diag** folder on the product installation disk to the directory on the z/OS machine where you want to unzip the installer.

2 Unzip **HPDiagTVJavaAgt_<version>_zos.tgz** using **gzip** as shown in the following example:

```
gzip -d HPDiagTVJavaAgt_9.00_zos.tgz
```

This command creates the unzipped file, **HPDiagTVJavaAgt_<version>_zos.tar**.

3 To unpack the .tar file, run the .tar command as shown in the following example:

```
tar -xvf HPDiagTV JavaAgt_9.00_zos.tar
```

This command creates the unpacked directory **javaAgentinstall**.

4 Prepare to run the Java Agent Setup Module (**setupModule.sh**) and JRE Instrumentor (**jreinstrumenter.sh**) by setting the permissions on the scripts so that they can be run (o+x).

5 Run the Java Agent Setup Module to configure the Java Agent to work with a Diagnostics Server and/or a TransactionVision Processing Server. For configuration details, see "Configuring the Java Agent on UNIX" on page 134.

For z/OS, use the **setupModule.sh** script or the following command:

```
<path_to_java> -Dcom.hp.javaagent.transaction.home=<tv_agent_dir> –
jar <path_to_setupModule.jar> -launchClass
com.mercury.opal.javaprobe.setupModule.SetupModule -importJarList
tvision.com.ibm.toad.jar,probe.jar,jreinstrumenter.jar,bootCL.jar,appCL.ja
r,probeCL.jar,appOrProbeCL.jar -importJarsFrom "*TV*/java/
lib,<agent_install_dir>/lib" -launchMethod launchSetupModule
-createBackups -console
```

In this instance:

- **<path_to_java>** is the path to the Java executable.
- **<path_to_setupModule.jar>** is **<agent_install_dir>/lib/setupModule.jar**.
- **<tv_agent_dir>** is **<agent_install_dir>/../TransactionVisionAgent**.

6 Run the JRE Instrumenter. For z/OS, use the following command:

```
<path_to_java> -jar <path_to_jre_instrumenter> -i <jvm_directory>
```

In this instance:

- <**path_to_java**> is the path to the Java executable.
- <**path_to_jre_instrumenter**> is <**probe_install_dir**>**/lib/jreinstrumenter.jar**.
- <**probe_install_dir**> is <**java_agent_install_dir**>**/DiagnosticsAgent**.
- <**java_agent_install_dir**> is the path to the Java Agent installation.
- <**jvm_directory**> is the path to the JRE for the Java installation your application is using.

7 If desired, configure the agent properties to enable the probe to work with the other Diagnostics components as described in "About Advanced java Agent Configuraiton" in the *HP Diagnostics Installation and Configuration Guide*.

8 After you install and configure the agent and instrument the JRE, you must manually modify the startup script for the application server so that an agent is started together with the monitored application. For detailed instructions, see Chater 7, "Configuring Application Servers for the Java Agent" in the *HP Diagnostics Installation and Configuration Guide*.

9 If the agent is configured to work with the other Diagnostics components, verify the agent installation as described in “Verifying the Java Agent Installation” in the *HP Diagnostics Installation and Configuration Guide*.

10 To collect z/OS system metrics for a Diagnostics server, see "Enabling z/OS System Metrics Capture" in the *HP Diagnostics Installation and Configuration Guide*.

Installing Java Agents on Multiple z/OS Machines

If you plan to install Java Agents on more than one z/OS machine, you might want to create a pax archive of the agent implementation on the first machine and then use the pax archive to install the agent onto the other machines. Contact your system administrato for more information.

Installing the Java Agent Using the Generic Installer

The installers for the Java Agent were built to support installing the agent on all of the platforms for which the component was certified. However, the agent might work on other platforms that are not yet certified. A generic installer is provided on the product installation disk to allow you to install Java Agent on these uncertified platforms.

For Java Agent to work on the platforms that are not supported by the regular installer, run the generic installer and manually configure the agent as a Java Probe and/or a TransactionVision Java Agent so that it can communicate with the other Diagnostics and/or TransactionVision components and monitor the processing of your application.

To install and configure the Java Agent on an uncertified platform:

1. Locate **HPDiagTVJavaAgt_<version>_unix.tgz** from the **Diagnostics_Installers** folder on the HP Diagnostics installation disk or from the **Data_Collectors_and_Components/Components/Diag** folder on the product installation disk.
2. Unzip **HPDiagTVJavaAgt_<version>_unix.tgz** using gzip as shown in the following example:

```
gzip -d HPDiagTVJavaAgt_9.00_unix.tgz
```

   When this command completes, the unzipped file is called **HPDiagTVJavaAgt_<version>_unix.tar**.

3. To unpack the tar file, run the following tar command:

```
tar -xf HPDiagTVJavaAgt_9.00_unix.tar
```

   This command creates the unpacked JavaAgent directory.

4 Run the Java Agent Setup Module to configure the Java Agent to work with a Diagnostics Server and/or a TransactionVision Processing Server. For configuration details, see "Configuring the Java Agent on UNIX" on page 134.

<probe_install_dir>/bin/setupModule.sh -console

- **<probe_install_dir>** is **<java_agent_install_dir>/DiagnosticsAgent**.
- **<java_agent_install_dir>** is the path to the Java Agent installation.

5 If desired, configure the agent to enable the agent to work with the other Diagnostics components as described in "About Advanced java Agent Configuraiton" in the *HP Diagnostics Installation and Configuration Guide*.

6 Configure the Application Server to allow the agent to monitor the application:

a Run the JRE Instrumenter as described in "Task 3: Run the JRE Instrumenter for Each Target JRE" on page 141.

b After you install and configure the agent and instrument the JRE, you must manually modify the startup script for the application server so that an agent is started together with the monitored application. For detailed instructions, see Chater 7, "Configuring Application Servers for the Java Agent" in the HP Diagnostics Installation and Configuration Guide.

7 If the agent is configured to work with the other Diagnostics components, verify the agent installation as described in “Verifying the Java Agent Installation” in the *HP Diagnostics Installation and Configuration Guide*.

Task 3: Run the JRE Instrumenter for Each Target JRE

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

The JRE Instrumenter can be run at the end of the Java Agent installation. Or if you skip running the JRE Instrumenter during the agent installation, you can run it manually before you can monitor the application.

Regardless of the JRE version, to enable all Diagnostics capabilities, to maximize the instrumentation coverage and to optimize monitored application performance, you should run the JRE Instrumenter. After you instrument the JRE version used by your application, copy the command line options presented by the JRE Instrumenter to the Java command in your application startup script.

The JRE Instrumenter instruments some standard Java classes for the JVM the application is using. It also provides you with the JVM parameters that must be used when the application server is started so that the application server uses the instrumented classes.

You must run the JRE Instrumenter again if the JDK (**java.exe** executable) used by the application server changes so that the Java Agent can continue to monitor the processing.

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 Agent for Multiple Application Server JVM Instances" in the *HP Diagnostics Installation and Configuration Guide*.

Notes:

- If you want to instrument IBM's 1.4.2 J9 JRE, you must instrument the correct classes and add the **-Xj9** option on the application's command line. The correct classes are located in the **<java dir>\jre\lib\jclSC14** directory (for example, jreinstrumenter.sh -i \usr\java14_64\jre\lib\jclSC14).
- The IBM 1.5.0 JVM supports the **-javaagent** option. However, using the option may sometimes cause increased CPU utilization by the JVM, which can mislead you to see it as Java agent overhead. Moreover, this option makes the JVM ignore the **-agentpath** option which is used by Diagnostics agents to load native libraries needed for the Diagnostics Java Profiler Heap Walker functionality. To avoid performance issues or to use Heap Walker, you need to use the JRE instrumenter and pre-instrument this JVM as documented in this section.

This section includes:

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

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.

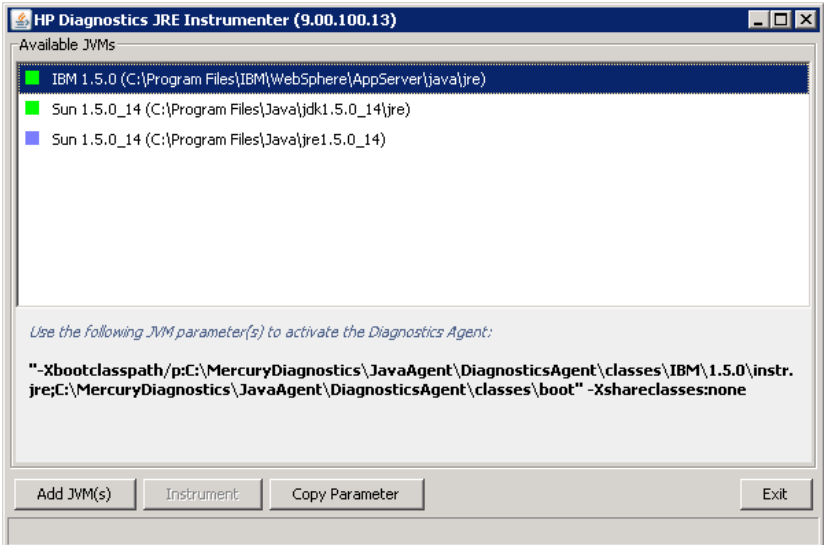
To start the JRE Instrumenter on a Windows machine:

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 dialog.



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

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

➤ **Add a New JVM**

➤ Click **Add JVM(s)** to search for other JVMs. The Instrumenter displays the Choose the Directory dialog box.

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

➤ Click **Update** to list all the folders in this directory in the **Folders** list.

➤ Select the folder where you would like to begin the search so that its name appears in the **File name** box.

➤ 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.

➤ **Instrument a 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.

➤ **Include the JVM Parameter in the Application Server's Startup Script**

If the JVM that you want to instrument is listed and has already been instrumented, you can copy the JVM parameter that must be inserted into the start script for the JVM to activate the Probe's monitoring.

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 144.

1 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 147 for details.
jreinstrumenter -a DIR	Look for JVMs below the DIR directory. See "Adding JVMs to the Available JVMs List" on page 148 for details.
jreinstrumenter -i JVM_DIR	Instrument the JVM in JVM_DIR. See "Instrumenting a Listed JVM" on page 148 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 148 for details.

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

./jreinstrumenter.sh -x

2 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.

3 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.

4 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.

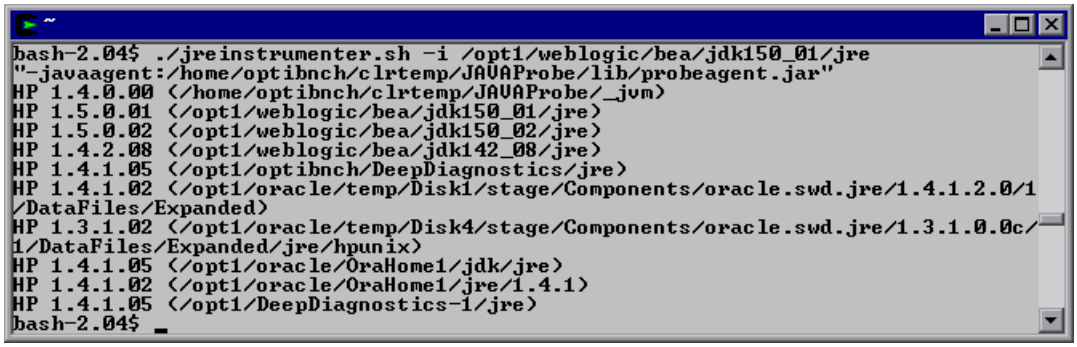
- Generic path to ClassLoader

  ./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:



5 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.

Task 4: Configure the Applications to Be Monitored

For the Java Agent to collect events from your application, it must be running the same Java instance as your application. To enable this instrumentation, you add options to the Java command that starts your application.

Regardless of the JRE version, you should run the JRE Instrumenter to instrument the JRE version used by your application. Copy the command line options presented by the JRE Instrumenter to the Java command in your application startup script.

To enable Java Agent for application servers, update the application server startup scripts manually. Refer to your application server documentation. Several examples can also be found in "Chapter 7: Configuring Application Servers for the Java Agent" of the *HP Diagnostics Installation and Configuration Guide*.

If the Java Agent is being used to monitor multiple JVMs running on the same machine, you must assign a unique probe identifier to the Java Agent for each JVM. You may do that using the Java command-line option **-Dprobe.id=<Unique_Name>**. For details, see "Configuring the Agent for Multiple Application Server JVM Instances" in the *HP Diagnostics Installation and Configuration Guide*.

Task 5: Set Up Messaging Queues and Communication Links

Like all agents in the TransactionVision deployment environment, the Java Agent requires a messaging middleware provider to manage the message queues. SonicMQ is the default messaging middleware provider and is included with the Java Agent.

If you want to use another message middleware provider, install and configure these as described by that product's documentation.

TransactionVision uses three message 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.

This section includes guidelines for the queues of each messaging middleware provider:

➤ "IBM WebSphere MQ" on page 151

➤ "Progress SonicMQ" on page 151

IBM WebSphere MQ

You 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. You also need to define a server connection channel and a listener on your queue manager. See the vendor's documentation for details.

Progress SonicMQ

If you are using the builtin SonicMQ, the queues are created automatically for you. If you are using a different installation of SonicMQ, you create the queues on your SonicMQ brokers using the SonicMQ Management Console. See the vendor's documentation for details.

Optional Task: 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, and so forth.

Contact your HP Consultant or Support Engineer for more information.

Optional Task: 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>\DiagnosticsAgent\bin\setupModule.cmd -createBackups
-console -recordFile <JASMResponseFileName>
```

➤ On UNIX:

```
<java_agent_install_dir>/DiagnosticsAgent/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>
```

On UNIX systems you need quotes around "**-DoNotRun**".

Uninstalling the Java Agent

- On a Windows machine, choose **Start** > **All Programs** > **HP Java Agent** > **Uninstaller**.

  Or run **uninstaller.exe**, which is located in the **<java_agent_install_dir>_uninst** directory.

- On a Linux or Solaris UNIX machine, run **uninstall***, which is located in the **<java_agent_install_dir>/_uninst** directory.

  On other UNIX machines, choose a 1.5 or later JVM and run **java -jar <java_agent_install_dir>/_uninst/uninstall.jar** to uninstall the Java Agent.

14

Installing and Configuring Agents on z/OS

This chapter includes:

About Agents in the z/OS Environment

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

- SLD Agent components are designed to monitor: CICS, WebSphere MQ (WMQ) Batch, WMQ IMS, and WMQ CICS events.
- SLM Agent components are designed to monitor WMQ IMS Bridge events.

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

Base Component Installation Summary

The general procedure for installing the base component is as follows:

1 Transfer files from the installation media to the zSeries platform.

Two options are available:

- Use the tvinstall_<ver>_zos_zseries.bat script found on the installation media to assist with the transfer, where <ver> is replaced by a three digit version-release identifier (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_<ver>_zos_zseries.bat script in Step 1 above to execute the TSO RECEIVE commands (recommended).
- Manually execute the needed TSO RECEIVE commands.

3 Tailor and execute the SMP/E installation job.

4 Review installation libraries.

5 Repeat steps 3 and 4 as needed.

The detailed procedure follows.

Base Component Installation Procedure

Perform the following steps to install the base component:

- "Step 1: Transfer the Files from the Installation Media to the zSeries Host" on page 157
- "Step 2: TSO RECEIVE the Transmitted Files to Create Product Datasets" on page 160
- "Step 3: Tailor and Execute the SMP/E Installation Jobs" on page 161

➤ "Step 4: Review the Installation Libraries" on page 163

➤ "Step 5: Repeat Base Component Installation Procedure (iF required)" on page 164

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

Two options are available:

➤ "Option 1: Run the tvisntall_<ver>_zos_zseries.bat script" on page 158

➤ "Option 2: Manually FTP the install files:" on page 159

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> = current version-release, for example 900

<&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:

tvsldf1_900_zos_zseries.xmit

An ftp output dataset name might be:

TVISION.SLD900.F1

Option 1: Run the tvisntall_<ver>_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. Review the information that is collected by the script by entering:

   tvinstall_<ver>_zos_zseries.bat /h | more

5. Invoke tvinstall_<ver>_zos_zseries.bat with no parameters and respond to the prompts.

Note: Note: If you are not satisfied with the values you enter, you can quit tvinstall_<ver>_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_<ver>_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_<ver>_zos_zseries.bat script re-executed.

Option 2: Manually FTP the install files:

Do not perform this step if you installed the files by using the tvisntall_<ver>_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=45 sec=15 tr

      where &custvol is a valid disk volser and &custunit is a valid disk device type.

   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> = current version such as 900

      For example:

      put tvsldf1_900_zos_zseries.xmit 'TVISION.SLD900.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_<ver>_zos_zseries.bat in step "Transfer files from the installation media to the zSeries platform." on page 156.
- Manually execute the needed TSO RECEIVE commands.

Option 1: Run the (TSORECV) Rexx Script

From a TSO command line or TSO/ISPF command line, run the (TSORECV) Rexx script created in "Transfer files from the installation media to the zSeries platform." on page 156. This results in the execution of TSO RECEIVE commands for all z/OS target datasets specified.

As the (TSORECV) Rexx script executes, IEBCOPY output messages will be directed to your TSO/ISPF terminal session. Press <ENTER> several times to scroll through the output while checking for any error messages.

The RECEIVEd datasets will be allocated using the same high level qualifier, however the character 'A' will be added as a <prodcode> prefix. For example, if input pattern=TVISION.SLD900.filequal, then the output pattern would be TVISION.ASLD900.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><ver>.filequal)'.

Command	Filename
RECEIVE INDSNAME('&hlq.SL_900.F1')	DSN('&hlq.ASL_900.F1')
RECEIVE INDSNAME('&hlq.SL_900.F2')	DSN('&hlq.ASL_900.F2')
RECEIVE INDSNAME('&hlq.SL_900.F3')	DSN('&hlq.ASL_900.F3')
RECEIVE INDSNAME('&hlq.SL_900.MCS')	DSN('&hlq.ASL_900.SMPMCS')

Step 3: Tailor and Execute the SMP/E Installation Jobs

Tailor and execute the jobs in the order presented. Jobs are in dataset **&custhlq.A<prodcode><ver>.F3**. After substituting the third 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.

Caution: If you do not have CICS and/or IMS installed on your system you must exeucte one or both of the DUMY* JCL members. These jobs create dummy IMS and CICS interface modules sufficient to satisfy binder (linkage editor) reqirements. If for example, you have CICS installed, but not IMS, you only need to execute the DUMYIMS member. The DUMY* jobs should be executed before SL_DDDEF.

Additionally, SL_DDDEF should be further customized so that references to IMS or CICS libraries are replaced by a reference to the **%tvhlq..SSLDLOAD** library. In general, the intent of this modification is to create and make available dummy interface modules for IMS and/or CICS by assembling and linking them into the TransactionVision SSLDLOAD library and then make all references to the SDFHLOAD (CICS) or SDFSRESL (IMS) libraries refer to the TransactionVision SSLDLOAD library. This customization is only needed when the customer does not have one or both of these IBM products installed.

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

Order	Job	Description
5.	SL_DDDEF	Defines SMP/E DDDEFs **Caution:** See required SLD configuration for shops that do not have either CICS or IMS installed. The configuration must be performed before SLDDDDEF is executed.
6.	SL_RECV	SMP/E Receive
7.	SL_APPLY	SMP/E Apply
8.	SL_IVP	Installation Verification
9.	SL_ACCPT	SMP/E Accept **Caution:** 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.

Step 4: Review the Installation Libraries

DS#	Dataset Name	Description
1	TVISION.SSL_AUTH	Agent Load Modules (APF Auth)
2	TVISION.SSL_INST	Agent Installation JCL Samples
3	TVISION.SSL_LOAD	Agent Load Modules
4	TVISION.SSL_PROC	Agent Sample JCL

- 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 5: Repeat Base Component Installation Procedure (iF required)

Depending upon the mainframe agent components licensed at your installation, it may be necessary to repeat steps 3-4 of the base component installation procedure using the second agent product code (SLM or SLD). Most customers are licensed for all agents, so if you have just completed the base installation steps using the SLD product code, repeat the steps using the SLM or vice versa.

Basic Configuration Steps for the SLD Agents: CICS, WMQ Batch, WMQ IMS, and WMQ CICS

1 Required for the CICS and WMQ CICS Agents - Update your CICS CSD files with the required resource definitions for SLD Agents by customizing and running job SLDCICSD, found in DS#2 - Agent 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 Agents.

This job step defines a program resource definition for CSQCAPX in your CICS region. Note - if your region already has a resource definition for CSQCAPX and the CSQCAPX program is already installed, then it is necessary to remove the previous version of CSQCAPX from your DFHRPL concatenation before the WMQ CICS Agent can be used. This requirement is the result of a non-dynamic naming scheme used by the WMQ CICS Adapter. As a result, only one crossing exit can be installed and operational and it must be named CSQCAPX. If this restriction results in a reduction of WMQ CICS Adapter functionality, contact your HP TransactionVision Marketing representative or HP Software Support.

2 Required for the CICS and WMQ CICS Agents - Place the CICS and WMQ CICS Agent program load modules in a library within 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 Agent.

3 Optional for CICS Agent - To automatically enable and disable the CICS Agent's exit programs at CICS startup and shutdown time, do the following:

a Define SLDPCSX as a second phase startup program (PLTPI), and define SLDPCPX as a second phase shutdown program (PLTSD). For details, see the "DFHPLT" section in the *CICS Resource Definition Guide*.

b Add the definition to your existing DFHPLTxx. If a new PLT is defined, add references to it in the CICS System Initialization Table (SIT). For details see "Specifying CICS system initialization parameters" in the *CICS System Definition Guide*.

c Repeat this step 3 for each CICS region to be monitored by the Agent.

Note: If this optional step is not performed, CICS transaction SLDS can be manually executed to enable the Agents.

Sample SIT entry:

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

Sample startup PLT definition with BI suffix:

```
//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
//*
```

Sample shutdown PLT definition with BS suffix:

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

4. Required - 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. Required - Create appropriate agent configuration and event queues on the WebSphere MQ queue manager to be used for communication between the Agent Manager components and the TransactionVision Analyzer. Customize and run job SLDCRTQS in DS#2 - Agent Installation JCL.
6. Required - Customize the sample TransactionVision Manager startup procedure: TVISION
7. Required - Customize the sample Agent Manager startup procedures: TVISIONC, TVISIONQ, and TVISIONM found in DS#4 and copy them to the appropriate procedure library for your site. It may not be necessary to copy all of the procedures if it is known that a particular agent type is not going to be deployed.
8. Required for WMQ CICS Agent - It is necessary to associate a WMQ CICS Agent to WMQ CICS Agent Manager. This is accomplished by defining a 4 byte alphanumeric agent SYSID and making it known to both the WMQ CICS Agent via the CICS INITPARM system initialization parameter, and the WMQ CICS Agent Manager via the SYSID start command parameter when starting up an Agent Manager address space. This step may need to be repeated if your environment consists of multiple CICS regions that need to be monitored by the WMQ CICS Agents. (The agent SYSID value may or may not correspond to the CICS SYSID value.)

Sample INITPARM Entry used by the WMQ-CICS Agent:

```
INITPARM=(……..,SLDPQMX='TVX1')
```

Corresponding WMQ CICS Agent Manager startup command:

```
F TVISION,START MQCICS SYSID(TVX1)
```

9 Optional for WMQ CICS Agent - To automatically start the WMQ CICS Agent's infrastructure initialization and monitoring programs at CICS startup time, define SLDPQME as a second phase PLTPI. Refer to the 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 requiring a WMQ CICS Agent. (Note - if this optional step is not performed, CICS transaction SLQE can be manually executed to enable the WMQ CICS Agent.)

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=SLDPQME
   DFHPLT TYPE=FINAL
   END
//*
```

10 Required for all z/OS TransactionVision Agents - The userid (account) associated with TransactionVision Agent Manager address spaces must have an OMVS segment defined to it.

11 Starting with z/OS version 1.9, the default setting for system parmlib DIAGnn member statement VSM ALLOWUSERKEYCSA has changed to **NO**. This parameter disallows User Key CSA storage for security reasons. In order for TransactionVision 9.00 agents to execute successfully under z/OS, the VSM ALLOWUSERKEYCSA parameter value must be set to **YES**.

See Chapter 15, "z/OS Components–Operation and Customization" for additional information regarding agent operation and architecture. Reading through this material will enhance your understanding of TransactionVision Agents and their interaction with other system components.

In addition, see "Guidelines for Efficient Operation of Agent and Agent Manager Components" on page 203 for information about how to minimize overhead the z/OS environment.

Basic Configuration Steps for the SLM Agent: WMQ IMS Bridge

1 Required - 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.)

2 Required - Create appropriate agent configuration and event queues on the WebSphere MQ queue manager to be used for communication between the Agent Manager components and the TransactionVision Analyzer. Customize and run job SLDCRTQS in DS#2 - Agent Installation JCL. (If you have already performed this step as part of SLD agent configuration, you may bypass it here.)

3 Required - Customize the sample TransactionVision Manager startup procedure: TVISION (If you have already performed this step as part of SLD agent configuration, you may bypass it here.)

4 Required - 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 agent and not necessarily the queue managers 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. See "The TVISIONB Buffer Queue" on page 199.
- The EDPROC parameter specifies the name of the procedure to start the TVISIOND address space. Copy the TVISIOND startup procedure to an appropriate PROCLIB
- The IMSJOB parameter specifies the jobname of the IMS control region for the IMS system to be monitored.

5 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 STEPLIB dataset already in use by the IMS control region.

(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.

6 Required for all z/OS TransactionVision Agents - The userid (account) associated with TransactionVision Agent Manager address spaces must have an OMVS segment defined to it.

7 Starting with z/OS version 1.9, the default setting for system parmlib DIAGnn member statement VSM ALLOWUSERKEYCSA has changed to **NO**. This parameter disallows User Key CSA storage for security reasons. In order for TransactionVision 9.00 agents to execute successfully under z/OS, the VSM ALLOWUSERKEYCSA parameter value must be set to **YES**.

See Chapter 15, "z/OS Components–Operation and Customization" for additional information regarding agent operation and architecture. Reading through this material will enhance your understanding of TransactionVision Agents and their interaction with other system components.

15

z/OS Components–Operation and Customization

This chapter includes:

Component Overview

All TransactionVision z/OS Agents are implemented as a combination of common base components: The TransactionVision Manager, one or more technology-specific Agent Managers, and agents (known as Agent Event Collectors). See the diagram in "Component Hierarchy" on page 174 which shows the individual components and interactions.

TransactionVision Manager

The TransactionVision Manager (TVM) is an authorized program that runs as a started task. It controls one or more Agent Manager subtasks and provides functionality common to all Agent Managers. The TVM may control only one Agent Manager subtask or several dependent upon the technologies in use and customer requirements. Once the TVM has been started, MVS modify commands are used to communicate requests to the TVM.

Agent Managers

Agent Managers operate as subtasks of the TVM and provide an interface between technology specific event collectors and the TransactionVision Analyzer. Agent Managers are not controlled directly via operator commands instead they are controlled indirectly via operator commands issued to the TVM.

Agent Managers are also technology specific. One or more Agent Manager subtasks can be started for a specific technology type. Technology types include: CICS, MQCICS, MQBATCH, and MQIMS.

Associated with each Agent Manager is a single data space buffer queue. Events are written to the data space buffer queue by an Agent (or Agent Event Collectors), and destructively retrieved from the data space buffer queue by the Agent Manager. After retrieving an event the Agent Manager typically performs marshalling and additional filtering before passing the event on to the TransactionVision Analyzer by way of the TransactionVision Event Queue.

The Agent Manager communicates with the TransactionVision Analyzer via WebSphere MQ. In response to configuration messages from the Analyzer, the Agent Manager indicates to the Agent Event Collectors 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.

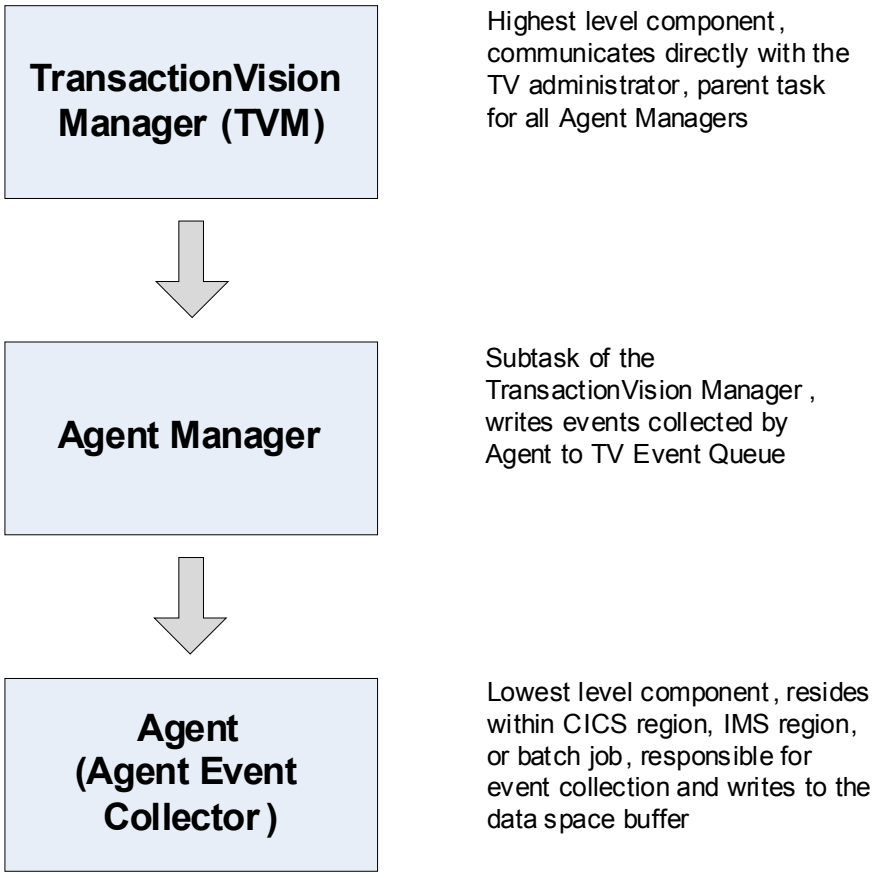
Agents or Agent Event Collectors

Agent Event Collectors capture data for a particular technology type. What they capture is determined by the Data Collection Filter Criteria in effect as defined to the TransactionVision Analyzer. Agent Event Collectors are instrumented to collect data in different ways dependent upon the technology type. Some utilize system exits and others require use of a WebSphere MQ replacement stub program. Regardless of the instrumentation style, Agent Event Collectors write the collected data to a data space buffer associated with the Agent Manager.

Depending on the technology type in use, Agent Event Collectors may be controlled by means of technology specific transactions or an administrative user interface.

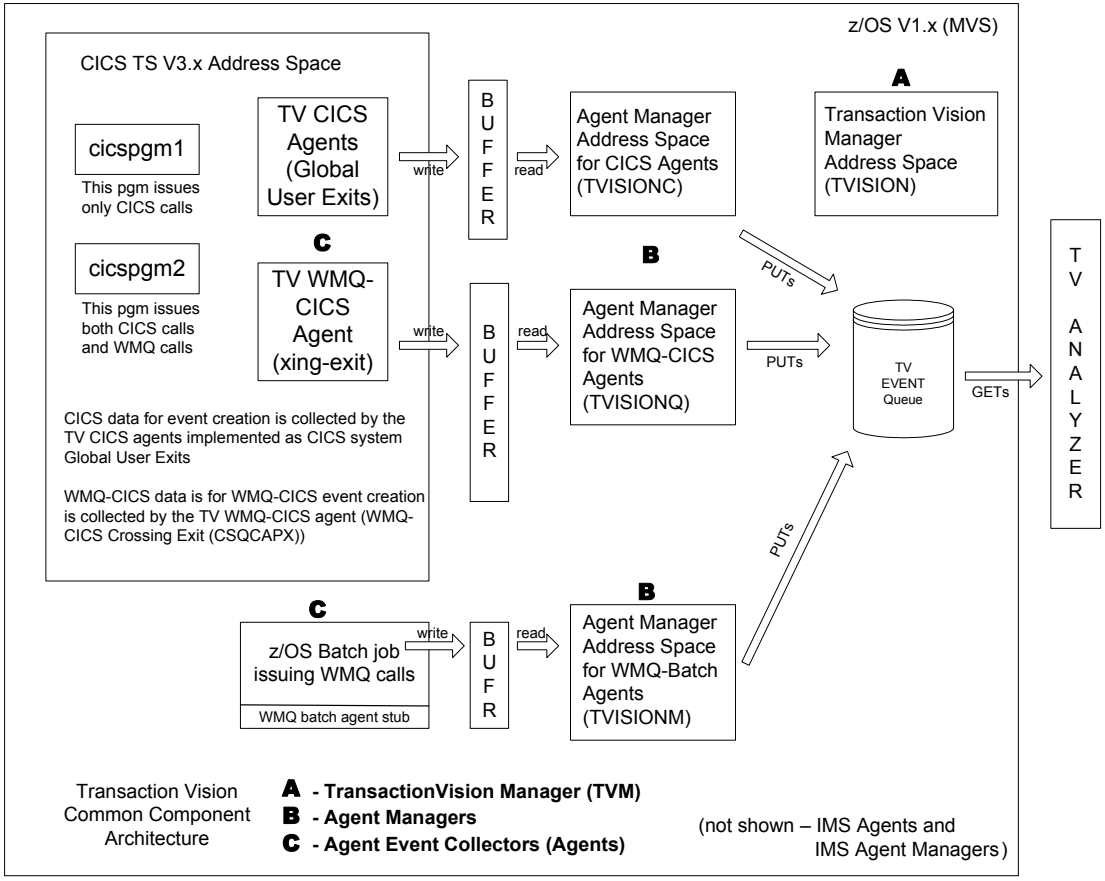
Component Hierarchy

The following diagram shows the component hierarchy.



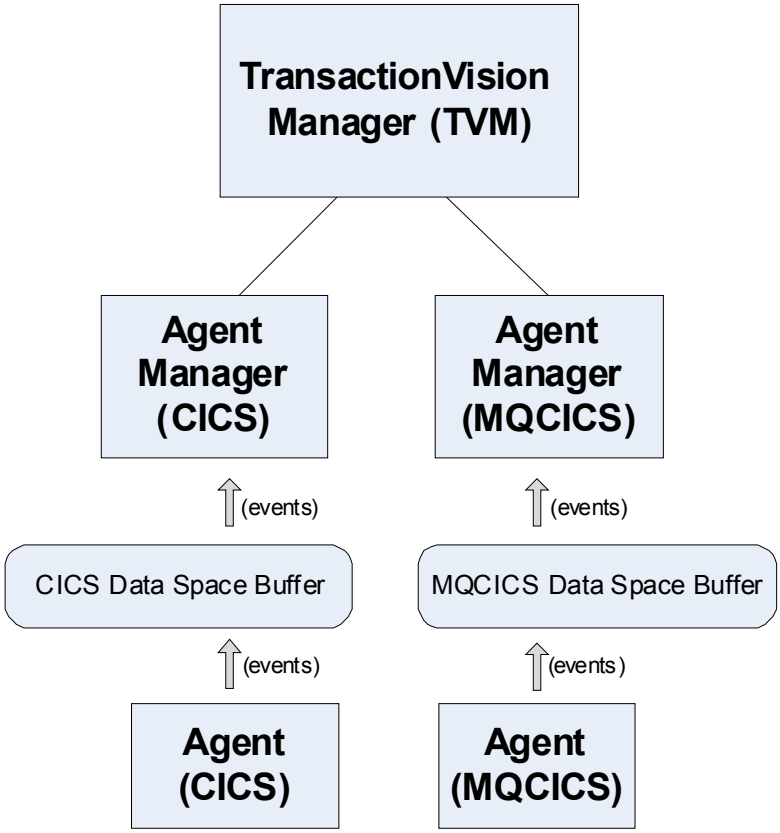
Architectural Overview

A single CICS or WMQ CICS Agent Manager can service events created by Agents in multiple CICS address spaces. The 'BUFFER' components are actually MVS data spaces. At agent manager startup time, the MAXQBLKS and QBLKSIZE parameters may be used to change the default size of the individual data spaces. The procedure names shown, TVISION, TVISIONC, TVISIONQ and TVISIONM are defaults which may be changed to conform to site naming conventions.

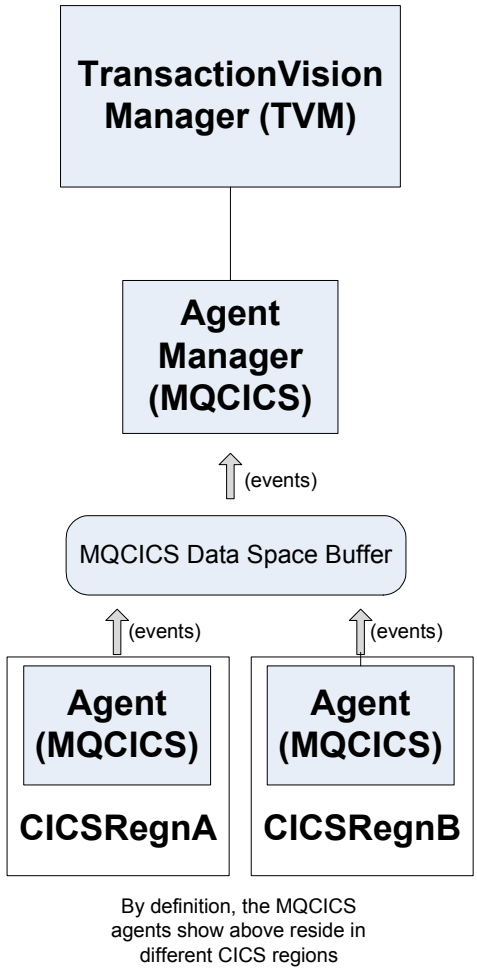


Component Configurations, Single Agent and Multi-Agent

The relationship between an agent manager and an agent of a particular technology type is used to classify the configuration as single-agent (1:1) or multi-agent (1:many). A single agent configuration is shown in the first diagram below



A multi-agent configuration is shown in the following diagram.



Controlling the TransactionVision Manager

Starting the TransactionVision Manager

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

where

procname is the name of a cataloged procedure name (typically TVISION)

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

tvid is a unique system id for an instance of TVM, consisting of four or fewer characters. (default value is 'TV01') Note that tvid is optional parameter, based on the procedure definition. In most environments, a single TVM instance is sufficient to manage all required Agent Managers. Therefore, the optional TVID= parameter is typically not used.

Example:

```
S TVISION
```

When the TransactionVision Manager is started the following messages will be displayed:

```
SLDS400I TVISION TransactionVision Manager startup in progress.
SLDS434I TVISION Hewlett-Packard TransactionVision for z/OS - V9.0
SLDS401I TVISION TV01 TransactionVision Manager startup complete.
```

Stopping the TransactionVision Manager

```
STOP jobname
```

where

jobname is the MVS task/job name specified or defaulted on the start command for the TVM job to be stopped.

Example:

```
P TVISION
```

The TransactionVision Manager component is stopped. Any Agent Manager subtasks still active will also be stopped as a result of terminating the TransactionVision Manager. 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 Agent Manager subtasks were active, their shutdown messages will also be displayed.

Note: All other Agent 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.

Controlling Agent Managers

This section includes:

"Starting an Agent Manager" on page 180

"Stopping an Agent Manager" on page 182

Starting an Agent Manager

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

where:

tvm_jobname is the MVS task/job of the TransactionVision Manager

START is the command name and is required.

agenttype is the type of agent to be started and is required. Specify either CICS, MQCICS, MQBATCH or MQIMS. If the agenttype is MQBATCH, do not specify either IMSID or SYSID.

sysid is required when the agenttype is CICS or MQCICS. When agenttype is CICS, then SYSID must specify the system id of the CICS region to be monitored. When agenttype is MQCICS, then SYSID identifies a customer defined agent SYSID. This should be the same value specified in the CICS INITPARM customization as found in step 8 of "Basic Configuration Steps for the SLD Agents: CICS, WMQ Batch, WMQ IMS, and WMQ CICS" on page 164.

imsid is required when the agenttype 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 used to start the Agent Manager. The default name is: TVISIONC when starting the CICS Agent Manager, TVISIONQ when starting the MQCICS Agent Manager, and TVISIONM when starting the MQBATCH Agent Manager.

qmgr is optional. qmgr specifies the name of the WebSphere MQ queue manager through which the Agent Manager will communicate with the TransactionVision Analyzer. The default qmgr is specified as a parameter in the Agent manager startup JCL procedure.

configq is optional. configq specifies the name of the WebSphere MQ queue from which the Agent manager will receive configuration messages from the TransactionVision Analyzer. The default is specified as a parameter in the Agent Manager startup JCL 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 "Data Space Buffer Queue Considerations" on page 189 for more information.

maxqblks is optional. maxqblks specifies the maximum number of buffer queue blocks that the agent 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 10. See "Data Space Buffer Queue Considerations" on page 189 for more information.

Example 1:

```
F TVISION,START  MQCICS  SYSID(TSTQ) MAXQBLKS(30)
```

The WebSphere MQ CICS Agent Manager (TVISIONQ) is created by means of an MVS MODIFY command to the TransactionVision Manager. The Agent Manager subtask is created in a separate address space and the following messages are displayed:

```
SLDS400I TVISION MQCICS  TransactionVision agent startup in progress.
SLDS401I TVISION MQCICS  TransactionVision agent startup complete.
IEF403I TVISIONQ - STARTED […and other MVS messages issued when starting a
task]
+SLMS278I : TransactionVision agent: WMQ <sysid> Agent manager startup
completed. QMGR=   CONFIGQ=
```

Example 2:

```
F TVISION,START CICS  SYSID(KIX1)
```

The CICS Agent manager is started using the default started task name - TVISIONC. The following messages are displayed:

```
SLDS400I TVISION CICS KIX1 TransactionVision agentstartup in progress.
SLDS401I TVISION CICS KIX1 TransactionVision agentstartup complete.

IEF403I TVISIONC - STARTED […and other MVS messages issued when starting a
task]

+SLMS278I : TransactionVision Agent Manager: CICS KIX1 Agent manager startup
completed. QMGR=    , CONFIGQ=
```

Stopping an Agent Manager

```
MODIFY tvm_jobname,STOP agenttype[SYSID(sysid) | IMSID(imsid)]
```

where:

tvm_jobname is the MVS task/job name specified or defaulted on the start command for the TransactionVision Manager.

STOP is the command name and is required.

Agenttype is the Agent type and is required. Specify: CICS, MQCICS, MQBATCH or MQIMS

If the agenttype is MQBATCH, do not specify SYSID or IMSID.

sysid is required when the agenttype is CICS or MQCICS. When agenttype is CICS, then SYSID must specify the system id of the CICS region to be monitored. When agenttype is MQCICS, then SYSID identifies a customer defined agent SYSID. This should be the same value specified in the CICS INITPARM customization as found in Step 7 of section 'Basic Configuration Steps for the SLD agents' in Chapter 19.

imssid is required if agenttype is MQIMS. imsid must specify the same IMSID that was specified on the start Agent command.

Example 1:

```
F TVISION,STOP MQCICS SYSID(TSTQ)
```

The WebSphere MQ CICS Agent Manager is stopped. The following messages are displayed:

```
SLDS404I TVISION MQCICS TransactionVision agent termination in progress.
SLDS443I TVISION MQCICS agent quiescing: 14 events in buffer queue.
…
SLDS445I TVISION MQCICS Agent quiesce completed.
SLDS448I TVISION MQCICS Agent statistics:
  Events in queue:   0
  Events collected   474
  Events dispatched  474
  Events_lost   0

+SLDS27BI : TransactionVision Agent: MQCICS Agent Manager is ending
IEF404I TVISIONQ - ENDED […and other MVS messages issued when stopping a job]
SLDS405I TVISION MQCICS TransactionVision agent termination complete.
```

Example 2:

```
F TVISION,STOP  CICS SYSID(KIX1)
```

The CICS Agent Manager is stopped. The following message sare displayed:

```
SLDS404I TVISION CICS  KIX1 TransactionVision agent termination in progress.
SLDS443I TVISION CICS  KIX1 agent quiescing: 14 events in buffer queue.
SLDS445I TVISION CICS  KIX1 Agent quiesce completed.
SLDS448I TVISION CICS  KIX1 Agent statistics:
Events in queue:   0
  Events collected   474
  Events dispatched  474
  Events_lost   0

+SLDS27BI : TransactionVision Agent: CICS  KIX1 Agent manager is ending
IEF404I TVISIONC - ENDED […and other MVS messages issued when stopping a job/
task]
SLDS405I TVISION CICS  KIX1  TransactionVision agent termination complete.
```

Inquiring about Event Collection

```
MODIFY tvm_jobname,INQUIRE STATISTICS [agenttype] [SYSID(sysid) |
IMSID(imsid)] [ALL]
```

where

tvm_jobname is the MVS task/job name for the TransactionVision Manager

INQUIRE is the command name and is required.

agenttype is the Agenttype and is required. Specify: CICS, MQCICS, MQBATCH or MQIMS. If agenttype is omitted, ALL Agenttypes are queried.

imsid or SYSID may be specified to identify the MQIMS or CICS or MQCICS Agent manager you wish to inquire of. If an IMSID or SYSID is omitted, all agent managers of the specified type are queried.

Example:

```
F TVISION,INQUIRE STATISTICS
F TVISION,I S  (abbreviated form)
```

The inquire statistics command displays the following:

```
SLDS448I TVISION MQBATCH Agent statistics:
  Events in queue:   56
  Events collected:   530
  Events dispatched:  474
  Events_lost:    0
SLDS448I TVISION MQCICS TSTQ Agent statistics:
  Events in queue:  175
  Events collected:  1068
  Events dispatched: 893
  Events_lost:     0
```

Controlling Agents (AgentEvent Collectors)

If theBasic Configuration Steps for the SLD Agents: CICS, WMQ Batch, WMQ IMS, and WMQ CICS in Chapter 14, "Installing and Configuring Agents on z/OS" are performed, including optional steps involving the definition of CICS PLTPI start-up resources, the following steps should not be necessary, since they will have been taken care of automatically during CICS 2nd phase startup processing. However, if manually enabling the CICS or MQCICS AgentEvent Collectors is desired, the following information may be of value.

Starting CICS AgentEvent Collectors

```
MODIFY cics_jobname, SLDS
```

where

cics_jobname is the MVS task/job name of the CICS region in which CICS AgentEvent Collectors are to be started.

SLDS is the CICS transaction-id which starts all CICS AgentEvent Collectors.

Stopping CICS Agent Event Collectors

```
MODIFY cics_jobname, SLDP
```

where

cics_jobname is the MVS task/job name of the CICS region in which CICS AgentEvent Collectors are to be stopped.

SLDP is the CICS transaction-id which stops all CICS Event Collectors.

Enabling the MQCICS AgentEvent Collectors

```
MODIFY cics_jobname, SLQE
```

where

cics_jobname is the MVS task/job name of the CICS region in which MQCICS AgentEvent Collectors are to be enabled.

SLQE is the CICS transaction-id which enables the MQCICS Event Collectors.

Note: In addition to enabling the infrastructure for the agent, it may also be necessary to logically start or activate the agent. This could happen if a Data Collection Filter was set to prevent any events from being collected by this agent. In this situation the agent is ENABLED for use, but it is deactivated. CICS transaction SLQA initiates an Administrative UI which can be used to manage the WMQ-CICS Agent.

Stopping (Disabling) MQCICS AgentEvent Collectors

```
MODIFY cics_jobname, SLQD
```

where

cics_jobname is the MVS jobname or started task name of the CICS region in which MQCICS AgentEvent Collectors are to be stopped.

SLQD is the CICS transaction-id which stops all MQCICS Event Collectors.

Note: In some situations it may be desirable to leave the Agent in a disabled state, but logically deactivate it. CICS transaction SLQA initiates an Administrative UI session which can be used to manager the WMQ-CICS agent.

MQ CICS Agent Event Collector Admin User Interface

```
SLQA
```

where

SLQA is a CICS transaction-id which initiates a pseudo-conversational dialogue to administer and display status of the WMQ-CICS Agent. By default, the agent will start in an ENABLED and ACTIVATED state, meaning TransactionVision infrastructure components are ENABLED to support event collection and the agent itself is operationally ACTIVE and collecting event data.

From the TransactionVision WMQ-CICS SLQA transaction it is possible to perform the following: ACTIVATE the agent, DEACTIVATE the agent, DISPLAY agent state + current statistics, and RESET agent session statistics. The following describes the WMQ CICS Agent Admin UI:

```
            Transaction Vision WMQ-CICS Agent Admin Interface
          Copyright 2010, Hewlett-Packard Development Company, L.P.
 ------------------------------------------------------------------------
    A - Activate   WMQ-CICS Agent  |  Agent Mgr Status    RUNNING
    D - DeActivate WMQ-CICS Agent  |  Agent Status        ACTIVE
    R - Reset Session Stats        |  Agent Activated     20:02:37  6
    X - Exit this Application      |  Session Started     20:03:04  6
    _ < Selection                  |  Current Time        20:03:35  6
 ------------------------------------------------------------------------
                  S E S S I O N   S T A T I S T I C S
 ------------------------------------------------------------------------
 Tot Calls Processed =            0   Tot Calls Bypassed =
 ------------------------------------------------------------------------
    MQFunc      Call Count    Avg Duration(secs)    Avg Bufr Sz(bytes
    ------    -------------   ------------------    -----------------
    MQOPEN                0              .000                    0
    MQCLOSE               0              .000                    0
    MQGET                 0              .000                    0
    MQPUT                 0              .000                    0
    MQPUTX                0              .000                    0
    MQINQ                 0              .000                    0
    MQSET                 0              .000                    0

PF3=EXIT  CLR=EXIT  ENT=Refresh      Msg: Statistics Display Refreshed
```

WMQ CICS Agent State Considerations

The state of a WMQ CICS Agent is described by both the status of the underlying infrastructure components (ENABLED/DISABLED) and the operational status of the agent itself (ACTIVE/INACTIVE). The operational status of the agent is very much dependent on the status of its infrastructure. For example, if the WMQ-CICS Crossing Exit, a critical part of the agent's infrastructure, is unavailable for some reason then the WMQ CICS Agent will be INACTIVE. If all infrastructure components are available, then agent will be ENABLED (default), and its operational state will initially be ACTIVE (default).

A TransactionVision administrator can control both the state of the infrastructure and operational state of the agent. Using the SLQD and SLQE transactions the state of an operational infrastructure can be controlled. Using the SLQA transaction the operational state of an enabled WMQ-CICS Agent can be controlled.

Data Space Buffer Queue Considerations

To minimize overhead, Agent Event Collector components in application environments store captured event data in a data space referred to as the buffer queue. Each Agent manager and associated AgentData Collector(s) have exclusive use of the data space.

The buffer queue is configured as a number of queue blocks of a certain size. Both of these dimensions can optionally be specified as parameters on the start Agent manager commands. 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. Default values in effect for SLD Agents are as follows:

	MAXQBLKS	QBLKSIZE	Data Space Size
WMQ-CICS	18	5MB	90MB
CICS	5	3MB	15MB
WMQ-Batch	5	3MB	15MB
WMQ-IMS	5	3MB	15MB

The appropriate buffer size varies widely based on several factors: transaction throughput of the monitored application environment, the type and size of events collected, and the throughput capabilities of the Agent 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 Agent Manager, 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 Agent manager 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.

In general, a data space buffer queue serving the needs of CICS agent components will require considerably less queue blocks than corresponding MQCICS agent components. The reasons for this are twofold, CICS events are generally of a fixed size, and MQCICS events are frequently much larger and of a more dynamic nature.

Stub Program Usage by the MQ-Batch and MQ-IMS Agents

This section includes:

z/OS Batch, IMS MPP, IMS BMP

On the z/OS batch and IMS platforms, the SLDLKSTB member of the sample procedure library thlqual.SSLDINST contains a sample job to bind the agent 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 agent batch or IMS stub—SLDBSTUB, SLDBRRSI, SLDBRSTB, or SLDQSTUB. After this bind, the application will invoke the agent 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.SSLDINST.

Note: 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 agent but are simply passed through to WebSphere MQ.

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

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 agent reads to retrieve configuration messages from the Analyzer.

To modify SLMBCNFG, perform the following steps:

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 Agent 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 Agent, see "Using the WebSphere MQ-IMS Bridge Agent" on page 196.

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

If the program SLMCNFQ can be executed by the agent via an EXEC CICS LINK, the agent 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 agent, SLMCNFQ should write the desired configuration queue name to the comm area passed to it, and then return. Subsequently, the agent will read the comm area to retrieve the correct configuration queue name.

Note: The installation procedure for the z/OS CICS Agent does not create an SLMCNFQ program. For instructions on writing this program, see "Writing SLMCNFQ" on page 193.

If the attempt to execute the SLMCNFQ fails, the z/OS CICS WebSphere MQ Agent tries to load the program SLMBCNFG and gets the configuration queue name. For instructions on generating this program, see "To generate SLMBCNFG, follow these steps: " on page 194.

If the attempt to load SLMBCNFG fails, the agent 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 194.

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;
 }
```

To generate SLMBCNFG, follow these steps:

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 Agent 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. 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 Agent library among those CICS regions.

To specify a different queue name for each CICS region, 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 AGENT 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'
```

Using the WebSphere MQ-IMS Bridge Agent

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

This section includes:

Agent Setup

Before using the WebSphere MQ-IMS bridge Agent, 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 agent 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 199.
- 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 Agent Operation

To operate the WebSphere MQ-IMS bridge Agent, 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 agent, 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 agent 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 agent. If an IMS system-specific agent is started while a monitor-all agent is running, monitoring of the targeted IMS system will be switched to the specific agent instance. Conversely, when a specific agent is stopped, monitoring of the targeted IMS system will be switched to the monitor-all agent, if running. To avoid confusion, it is recommended that you run only specific agents or run a monitor-all agent and no specific agents. Only one monitor-all agent will be allowed and only one agent 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 agent is controlled from the TransactionVision UI/Job Server. However, you may disable the agent 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 agent as follows: F TVISIONB,ENABLE MQIMSBDG.

5. Stop the TVISIONB address space as follows: P TVISIONB. This will implicitly disable the agent; 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 agent 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 Agent 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 agent 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" on page 201
- "IMSBridgeObject.xml" on page 201

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: 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

Considerations for Agent Operations

- The TransactionVision Manager should be started before Agent Managers or Agent Event Collectors are started.
- SYSIDs and IMSIDs are specific to a particular Agent Manager, they cannot be shared amongst Agent Managers.
- The prohibition of IMSID/SYSID rules will be enforced across all instances of the TransactionVision Manager, so you cannot start an Agent to collect from a specific IMS system or CICS region if another Agent is already collecting events from that IMS system or CICS region.
- The TVID must not be the same value as the IMSID or SYSID value of any agent. 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.
- Stopping the TransactionVision Manager will automatically stop all the agents under its control.
- TransactionVision Manager termination will not complete until all Agent Managers under its control have terminated.
- If you cancel the TransactionVision Manager, all agents under its control will immediately terminate and all remaining events in the buffer queue will be discarded.

Guidelines for Efficient Operation of Agent and Agent Manager Components

This section contains the following topics:

Enable Event Packaging Functionality of the Data Collection Filter

Event packaging can have a significant impact on the overall efficiency of a TransactionVision enabled environment. Significant CPU reductions were observed for TransactionVision Agent Manager components (TVISIONC and TVISIONQ) and the Win both the TransactionVision WMQ-CICS Sensor Manager (called TVISIONQ) and WMQ Channel Initiator.

Agent Manager CPU overhead reductions were observed using relatively low event/package ratios (10:1 or 20:1 for example). Although higher ratios produced the greatest savings, ratios in excess of 50:1 may result in substantially increased virtual storage utilization requiring the customer to monitor storage utilization more carefully in order to minimize over-commitment side-effects such as paging. It is therefore recommended that preference be given to more modest event packaging ratios.

Enable Data Range Functionality of the Data Collection Filter to Reduce Event Payload Size

Tests have be run to determine the effectiveness of reducing event payload size (Data Range Data Collection Filter criterion) on overall CPU consumption by TransactionVision components. Reducing event payload size within TransactionVision produces a commensurate reduction within CICS and WMQ components as well.

In general, it is only recommended that TransactionVision customers only collect as much of the event payload data as is needed to satisfy monitoring, debugging, or other application related requirements.

Migrate All TransactionVision WMQ Traffic to a Separate WMQ Queue Manager and Channel Initiator Pair

Customers using CICS, WMQ CICS, and WMQ Batch TransactionVision agents which share WMQ resources with production workloads are encouraged to define a separate set of WMQ resources to handle the TransactionVision Event, Configuration, and Exception messages and processing. Segregating TV-WMQ resources provide several benefits:

- Reduced contention for resources among TransactionVision and production application processes by moving the larger packaged messages out of the customer's production environment.
- Allows TransactionVision to be an independently prioritized workload (rather than defaulting to production).
- Eliminates the possibility of TransactionVision impacting a customer's production workload.

The customer is also encouraged to monitor and tune the WMQ channel disconnect interval. The combination of a short disconnect interval large (infrequent) messages can increase CPU consumption due to excessive channel reconnects.

16

Installing and Configuring Agents on BEA Tuxedo

This chapter includes:

Preparing for the Installation

You start the agent installation from the product installation disk .

The following table shows the installation file names for the BEA Tuxedo packages for each supported platform:

Platform	Files
AIX	HPTVTuxedoAgent_<version>_aix.tgz
HP-UX	HPTVTuxedoAgent_<version>_hppa.tgz
Solaris	HPTVTuxedoAgent_<version>_sol.tgz

Running the Installation

To start the agent installation program, perform the following steps:

1 Change to the directory location of the TransactionVision installation file.

Note: On Solaris and HP-UX, you must instead copy the installation files from the product installation disk 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 207. For example:

gunzip HPTVTuxedoAgent_<version>_hppa.tgz

4 Enter the following command to begin the installation procedure:

./tvinstall_<version>_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 WebSphere MQ Agent
3. TransactionVision BEA Tuxedo Agent
4. 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 agents 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 Agent

Unlike the WMQ Agent, the Tuxedo Agent installation does not automatically call the rebind script.

Run the **rebind_tux_sensor** script to relink the agent library. For more information about this script, see ee "Command-Line Utilities" in *Using Transaction Management*.

Uninstalling Agents

To uninstall the Agent or any other TransactionVision components on the host, perform the following steps:

1 Log in as superuser:

su

2 Enter the following command:

./tvinstall_<version>_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 WebSphere MQ Agent
3. TransactionVision BEA Tuxedo Agent
4. 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 Agents

Before you can use TransactionVision to record event data for an application, you must configure the application environment to load the Agent library instead of the standard BEA Tuxedo Agent library.

The required configuration consists of the following:

➤ "Configure the Application Library to Load the Agent Library" on page 211

➤ "Set Required Environment Variables" on page 212

➤ "Setting the Agent Connection URL" on page 213

An optional configuration is:

➤ "Filter Sensitive Data" on page 213

Configure the Application Library to Load the Agent Library

The BEA Tuxedo Agent library is dynamically loaded at runtime by including its library search path before the standard BEA Tuxedo Agent library search path. The standard BEA Tuxedo Agent library is dynamically loaded by the Agent library.

Add the directory location of the Agent 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 Agent library exists as part of the environment value, the agent 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 agent installation locations. If the agent was installed in a location other than the default, specify the directory location for the agent 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
HP-UX	SHLIB_PATH	/opt/HP/TransactionVision/lib64
IBM AIX	LIBPATH	/usr/lpp/HP/TransactionVision/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 agent 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 Transaction Management*.

Set Required Environment Variables

Two environment variables, **TVISION_HOME** and **TUXDIR**, are required by the BEA Tuxedo Agent. These environment variables should be set on the host where the monitored application runs before starting the application.

The **TVISION_HOME** environment variable should be set to the absolute path of the TransactionVision installation directory. For example, on Solaris and HP-UX, **TVISION_HOME** would be **/opt/HP/TransactionVision**; on AIX, it would be **/usr/lpp/HP/TransactionVision**.

The TUXDIR environment variable is typically required by BEA Tuxedo applications and should already be set in the application environment. If not, it should be set to the absolute path of the BEA Tuxedo installation directory.

Setting the Agent Connection URL

You must modify the BEA Tuxedo Agent property file (<**TVISION_HOME**>/**config/TuxedoSensor.properties**) and set the **transport** option in the **static configuration** section to the agent connection URL.

The agent Connection URL is derived from the TransactionVision HTTP Communication link definition. The URL is a combination of the hostname and port number of the Analyzer. A Processing Server host can have up to 5 Analyzers. Be sure to use the correct port number for your Analyzer. For example: http://analyzer.hostname.com:21120/ or http://analyzer.hostname.com:21130/.

Filter Sensitive Data

The BEA Tuxedo Agent 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 Agent property file (<**TVISION_HOME**>/**config/TuxedoSensor.properties**) and set the **collectUserData** option in the **static configuration** section to **no**.

17

Installing and Configuring the .NET Agent

The .NET Agent combines the capabilities of the Diagnostics .NET Probe and the TransactionVision .NET Agent into a single component. The .NET Agent can simultaneously serve as both the TransactionVision Agent and the Diagnostics Probe on a .NET host.

The .NET Agent provides a low-overhead capture solution that works with HP Software's BSM 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 223.
  - 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 captures 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 also need Remoting.points and must setup the application for instrumentation.

    To generate .NET MSMQ Send and Receive Events you also need MSMQ.points and must set up the application for instrumentation. For details, see "Configuring Support for MSMQ Based Communication" in the *Diagnostics Installation and Configuration Guide*.
- Optional configuration. Modify the agent configuration in the **probe_config.xml** file. See "Configuring the .NET Agent" on page 223.
- Events that TransactionVision can trace with the .NET Agent. For a list of the events, see "About .NET Agent" on page 104.

Installing the .NET Agent

The following steps provide detailed instructions for a first time installation of the .NET Agent on a Windows machine.

Note: If there is a pre-existing installation of the .NET Agent on the host machine, see Chapter 24, "Upgrading TransactionVision," for important instructions on how to upgrade the agent systems.

This section includes:

Preparing for the Installation

You must install the .NET Agent on the host machine of the application that you want to monitor. The overhead that the agent for .NET imposes on the system being monitored is extremely low.

The following are the recommendations for memory and disk space that support the agent's processing:

- **Platform**: All Supported Platforms
- **Memory**: 60 MB Additional RAM

- **Free Hard Disk Space**: 200 MB Additional Space
- **.NET Framework**: 2.0 or later

Note: If you must support .NET Framework 1.1, you need to use an earlier version of the .NET Agent (8.x).

Launching the .NET Agent Installer

You can install the .NET Agent in one of the following ways:

- For major releases, launch from the product installation disks.
- For patch or minor releases, download the .NET Agent installer from the Software Support Online (SSO) portal. You must have an HP Passport account.

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

To launch the installer from the BSM product installation product disks:

1 Locate the installation package file for your platform:

Platform	Files
Windows 2003/2008 32-bit	HPDiagTV.NetAgt_<version>_win32.msi
Windows 2003/2008 64-bit	HPDiagTV.NetAgt_<version>_win64.msi

2 Run the installer in the **\Data_Collectors_and_Components\Components\TV** directory of the product installation disks.

3 Continue with "Running the Installation" on page 219.

To launch the Installer from the SSO Portal:

1. Access the SSO portal at http://support.openview.hp.com/selfsolve/patches, using your HP Passport login.
2. Select **TransactionVision** in the Product field, <***patch_number***> for Product Version, **Windows** for Operating system, and click **Search**.
3. Click the appropriate link to download the .NET Agent installer for Windows.
4. Continue with "Running the Installation" on page 219.

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.

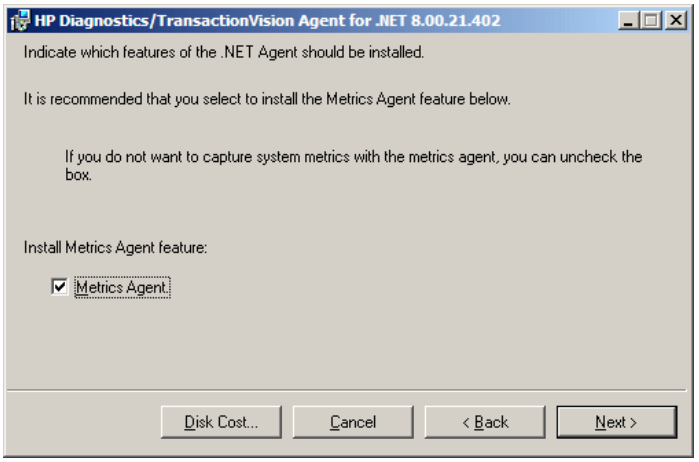
2. Provide the location where you want the Agent installed.

   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.

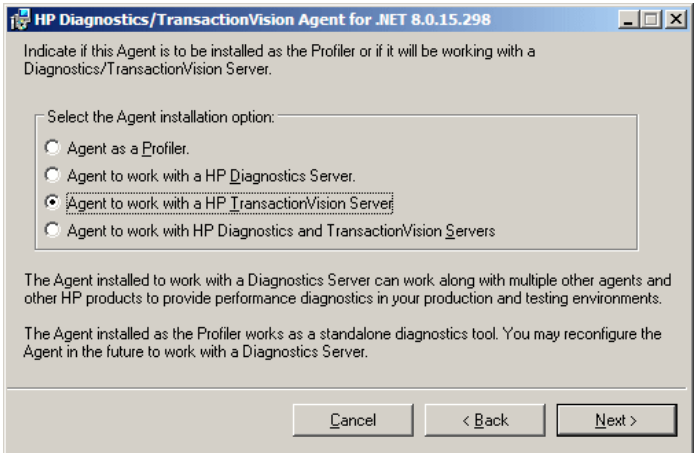
3 Select the .NET Agent features that you want to install.



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.



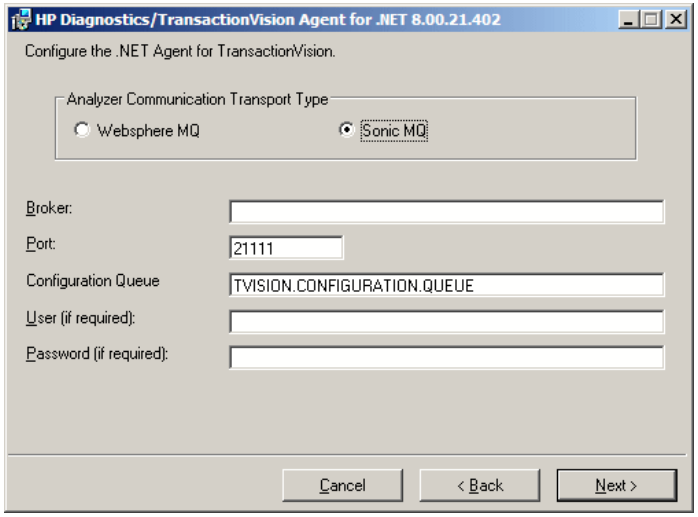
5 Select **Agent to work with an HP TransactionVision Server** and click **Next** to continue.

The Configure the .NET Agent for TransactionVision dialog appears.

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 .NET Agent to Work in the TransactionVision Environment

1 On the Configure the .NET Agent for TransactionVision dialog, select the Messaging Middleware Provider: either **WebSphere MQ** or **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 case, 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.

2 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 "Command-Line Utilities" in *Using Transaction Management*.

3 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 "Command-Line Utilities" in *Using Transaction Management*.

Websphere MQ channel. Channel name for WebSphere MQ queue manager.

Websphere MQ Q Manager. CCSID for WebSphere.

Click **Next** to continue.

4 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 229). 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 **<agent_install_dir>/etc/probe_config** file.

Use the following elements:

- "<tv> element" on page 224
- "<timeskew> element" on page 225
- "<transport> element" on page 226
- "<logging> Element" on page 228
- "<modes> Element" on page 228

The <tv>, <timeskew>, and <transport> elements are supported for TransactionVision only. The <logging> and <modes> elements are supported by TransactionVision and Diagnostics.

For complete information on the **probe_config** file, see the *HP Diagnostics Installation and Configuration Guide*.

This section also includes the following additional configuration topic:

- "Enabling Correlation of .NET Events" on page 229

<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).
eventmemorythreshold	number	250,000,000	(Dynamic) The memory consumed by the internal buffer(Q) after which the .NET Agent tries to 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.

Where:	Is:
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 "Command-Line Utilities" in *Using Transaction Management*.
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>
configurationQueue = <configurationQueue>
```

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 Agent.

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"/>
```

Enables the capture of TransactionVision events. This mode sends events to TransactionVision. This mode can be combined with other modes.

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*.

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.XMLRuleCorrelatio
nBean"/>
```

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** and press **Enter**.

This command restarts the Web publishing service but does not immediately start the .NET Agent. The next time that a Web page in the application is requested, the agent is started, the applications are instrumented, and the agent 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

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. For configuration instructions, see "Configure the .NET Agent to Use SSL" on page 329. For more information, see the *HP Business Service Management Hardening Guide* PDF.

18

Installing and Configuring the IBM WebSphere DataPower Agent

This chapter provides instructions on installing and configuring the HP TransactionVision Agent for IBM WebSphere DataPower.

This chapter includes:

Installing the IBM WebSphere DataPower Agent

The IBM WebSphere DataPower Agent is installed with the TransactionVision **Processing Server**. For information about installing the **Processing Server**, see "Processing Server Installation" on page 51.

Deploying and Configuring the IBM WebSphere DataPower Agent

Following are the major tasks needed to deploy and configure the DataPower Agent on an IBM WebSphere DataPower appliance:

Task 1: Import the IBM WebSphere DataPower Agent

The DataPower Agent is contained in a zip package (HPTVMonitoring.zip) which is imported into all DataPower appliances to be monitored by TransactionVision. The HPTVMonitoring.zip package is installed with the TransactionVision **Processing Server** and is located in the integrations directory within the TransactionVision installation directory.

To import the DataPower Agent, follow these steps:

1. Login to the DataPower default domain.
2. In the main DataPower Control Panel, click the **Import Configuration** icon.

3 Choose from **ZIP** and click **Browse** to select the HPTVMonitoring.zip file.

Note: You may need to transfer the HPTVMonitoring.zip file from the host where the Processing Server was installed to your local host.

---

4 Click **Next**.

5 Under **Domains contained in this Package**, check **HPTVMonitoring** to create the dedicated **HPTVMonitoring** domain and click **Next**.

The details of the configuration to be imported are listed.

6 Click **Import** to import the DataPower Agent configuration.

Import results are listed.

7 Click **Done**, then click **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

8 Check **default** and **HPTVMonitoring** domains to save.

9 Click **Save selected domains**, then click **Close Window**.

Task 2: Configure DataPower for WS-Management Monitoring

In addition to importing the DataPower Agent configuration, you need to configure and activate Web Services Management (WS-Management) Monitoring on DataPower in the following main steps:

➤ "Step 1: Configure the DataPower WS-Management Interface" on page 236

➤ "Step 2: Configure the DataPower Admin User Name and Password for the DataPower User Agent" on page 236

➤ "Step 3: Define the domains to be monitored by TransactionVision" on page 237

➤ "Step 4: Configure Agent settings in properties.xml" on page 237

➤ "Step 5: Define TransactionVision Multi-Protocol Gateway HTTP Header Injections" on page 238

➤ "Step 6: Enable the Scheduled Subscription Rule" on page 239

➤ "Step 7: Enable WS-Management Agent in Monitored Domains" on page 239

Step 1: Configure the DataPower WS-Management Interface

1. From the DataPower Control Panel in the **default** domain, select **Network** > **XML Management Interface**.
2. Enable **Admin State**.
3. Set **Local IP Address** to **0.0.0.0.**
4. Set **Port Number** to **5550**.
5. Ensure **WS-Management Endpoint** is checked (others may be checked as well).
6. Click **Apply** then click **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

Step 2: Configure the DataPower Admin User Name and Password for the DataPower User Agent

1. From the DataPower Control Panel in the HPTVMonitoring domain, select **Objects** > **User Agent**.
2. Click **scheduled_rules_ua**.
3. Select the **Basic-Auth Policy** tab.
4. Click the **Edit** icon for the ***/ws-management** URL Matching Expression.
5. In the popup windows, enter the DataPower Admin User Name and Password.
6. Click **Apply** to apply and close the popup window.
7. Click **Apply** in the Configure User Agent page.
8. Click **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

Step 3: Define the domains to be monitored by TransactionVision

1. From the DataPower Control Panel in the HPTVMonitoring domain, click the **File Management** icon.
2. Expand the **local:** folder.
3. Edit the **domain-subscriptions.xml** file.
4. Enter the name of the DataPower domain on this appliance to be monitored by the DataPower Agent, by entering the name with the **<domain>** tags (replace the **demo-web-services** example).

Note: Multiple domains can be monitored by listing multiple <domain> elements within <subscriptions>.

---

5. Click **Submit** to save the changes to **domain-subscriptions.xml**.

Step 4: Configure Agent settings in properties.xml

1. From the DataPower Control Panel in the **HPTVMonitoring** domain, click the **File Management** icon.
2. Expand the **local:** folder and edit the **properties.xml** file.
3. Define TransactionVision's SonicMQ HTTP Listener URL to which the TransactionVision Multi-Protocol Gateway sends events by replacing the URL between the **<monitor-url>** tags with the URL to TransactionVision's SonicMQ HTTP Listener.

   By default, the listener runs on port **21113** with a URI path of **tv_datapower**. For example:

   http://analyzer_hostname:21113/tv_datapower/

4 Set the DataPower appliance hostname between the **<datapower-hostname-for-events>** tags.

Note: This should be the short hostname without the fully qualified domain name.

---

5 Click **Submit** to save the changes to **properties.xml**.

Step 5: Define TransactionVision Multi-Protocol Gateway HTTP Header Injections

1 From the DataPower Control Panel in the **HPTVMonitoring** domain, click the **Multi-Protocol Gateway** icon.

2 Click on **hptv-wsman-subscriber**.

3 Click on the **Headers** tab.

4 Add the following **Header Injection Parameters**:

Direction	Header Name	Header Value
Back	X-JMS-CorrelationID	DataPowerEvents
Back	X-JMS-DestinationQueue	HTTP.EVENT.QUEUE
Back	X-JMS-MessageType	TEXT
Back	X-JMS-DeliveryMode	NON_PERSISTENT or PERSISTENT

Note: If you want events to be persisted on the event queue when the SonicMQ broker is restarted, set X-JMS-DeliveryMode to NON_PERSISTENT or PERSISTENT.

---

5 Click **Apply**, then click **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

Step 6: Enable the Scheduled Subscription Rule

1 From the DataPower Control Panel in the **HPTVMonitoring** domain, select **Objects** > **XML Manager**.

2 Click on **scheduled_rules**.

3 Enable **Admin State**.

4 Click **Apply** then click **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

Step 7: Enable WS-Management Agent in Monitored Domains

1 In each domain to be monitored by the TransactionVision Agent, select **Objects** > **Web Services Management Agent**.

2 Enable **Admin State**.

3 Set **Maximum Record Size** (such as 1000000).

4 Set **Maximum Memory Usage** (such as 128000).

5 Set **Capture Mode** to **All**.

6 Click **Apply** then click **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

Task 3: Configure Monitored Processing Rules

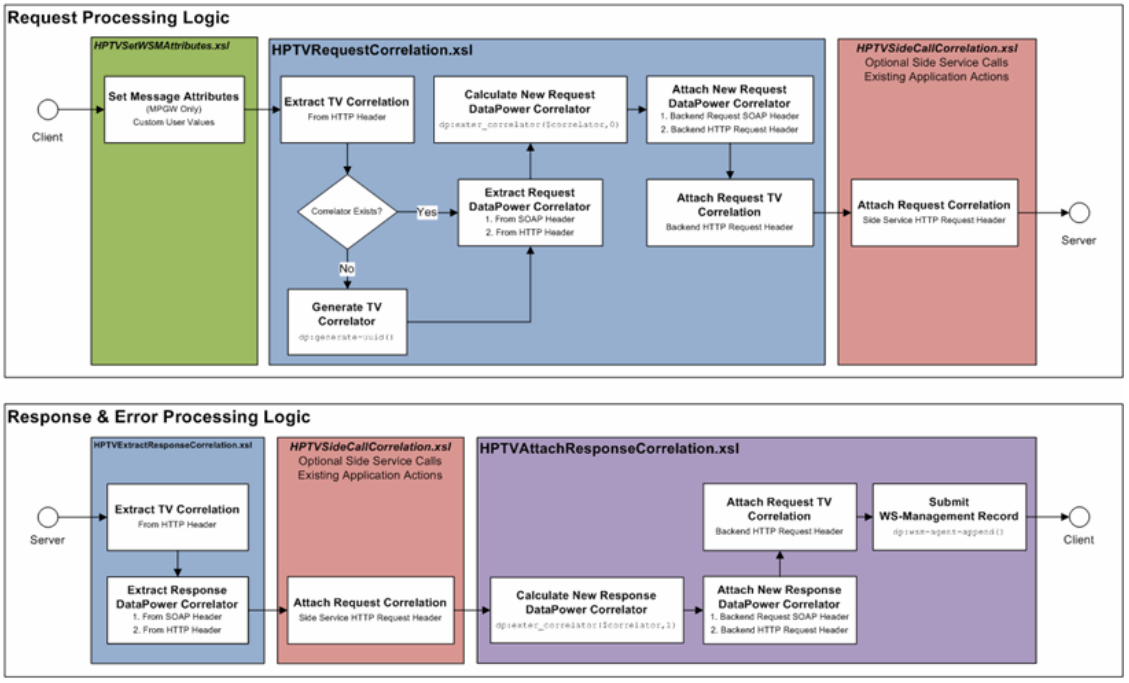
For TransactionVision to generate DataPower events and properly correlate between them (such as a DataPower service calling another DataPower service), additional XSL stylesheets need to be added to the action rules in each processing rule for each monitored DataPower service.

In the rare case where simple Web Service Proxies are being monitored and there is no need to correlate them with any other events from the DataPower Agent or other Agents, it is not required to add the additional XSL stylesheets to the service processing rules. However, it is highly recommended to always use the XSL stylesheets provided by TransactionVision, since the stylesheets provide additional information in the collected events.

TransactionVision provides five stylesheets to instrument proper event correlation and generation within the DataPower services to be monitored. These stylesheets can be found in the integrations directory of the TransactionVision **Processing Server** installation, and should be uploaded to all domains on all DataPower appliances that are to be monitored by TransactionVision.

- **HPTVSetWSMAttributes.xsl**
- **HPTVRequestCorrelation.xsl**
- **HPTVExtractResponseCorrelation.xsl**
- **HPTVAttachResponseCorrelation.xsl**
- **HPTVSideCallCorrelation.xsl**

The following diagram shows how these stylesheets need to be applied in the various rules within a DataPower service to be monitored. It is not as important to understand the logic within each TransactionVision correlation stylesheet as it is to know and understand their general purpose and the proper location to include them in the processing rules.



Note: All processing rules of a DataPower service to be monitored need to be instrumented, including any error processing rules.

It is not necessary to pass the output of the stylesheets as they do not make any mandatory changes to their input. However, they do pass the input as output and can be used in that way, if needed for a PIPE chain within the processing rule.

Following are the processing rules that can be configured:

Configuring Request Processing Rules

There are two basic configurations that define which stylesheets to use for instrumentation in the request processing rule:

Simple Web Service Proxy

This simplest configuration is a Web Service Proxy. In this case, the required **HPTVRequestCorrelation.xsl** (top blue section) can be added at any location within the request processing rule, but it is recommended to be placed as close to the beginning of the rule as possible to properly catch and correlate exceptions that could occur within the processing rule. The output of **HPTVRequestCorrelation.xsl** can be set to NULL or passed along in a pipe.

Multi-Protocol Gateway

Multi-Protocol Gateways are instrumented much like Web Service Proxies. However, there is an additional requirement to define the following service variables to identify the service since there is not a Web Service Definition Language (WSDL) to define these service variables. The provided **HPTVSetWSMAttributes.xsl** (green section) stylesheet can be used as an example to create such DataPower service-specific stylesheets. These service variables must be defined before **HPTVRequestCorrelation.xsl** is executed in the request processing rule. It, too, can have its output set to NULL or passed along in a pipe.

Below are the required services identifying DataPower service variables to be set for Multi-Protocol Gateways:

- **var://service/wsm/operation**

  This is the namespace-qualified operation name. For example: *{www.mynamespace.com}myOperation*

- **var://service/wsm/service-port**

  This is the namespace-qualified name of the service port.

- **var://service/soap-oneway-mep**

  This variable should be set to true() if this operation is one-way, or false() otherwise.

- **var://service/wsm/service**

  This is the namespace-qualified name of the service.

Configuring Response Processing Rules

Instrumenting response rules for TransactionVision monitoring is the same for Web Service Proxies and Multi-Protocol Gateways. There are two required stylesheets:

- **HPTVExtractResponseCorrelation.xsl** (bottom blue section)
- **HPTVAttachResponseCorrelation.xsl** (bottom purple section)

Separate stylesheets are needed because if there are any side service calls within the response processing rule, they must be made after correlation information is retrieved from the backend call, but before the response DataPower correlation is calculated.

HPTVExtractResponseCorrelation.xsl needs to be executed before **HPTVAttachResponseCorrelation.xsl**, so it should be placed in the beginning of the response rule or as early as possible.
HPTVAttachResponseCorrelation.xsl should be executed at the end of the response rule (or as late as possible). Both stylesheets can output to NULL or to a pipe.

Configuring Error Processing Rules

In order to properly capture events when errors occur within DataPower services, error processing rules need to be defined and instrumented. Error processing rules are instrumented in the same way that response processing rules are instrumented. Use the same **HPTVExtractResponseCorrelation.xsl** and **HPTVAttachResponseCorrelation.xsl** stylesheets.

Configuring Rules With Side Service Calls

The red sections in the graph above represent any existing processing actions within the service's processing rules. In some cases, these existing actions may include transforms which are making side service calls via DataPower extensions like **dp:url-open** or **dp:soap-call()**. In order to maintain event correlation through such side calls, TransactionVision requires its correlation information to be passed to the side service call via the HTTP request header.

Note: HPTVRequestCorrelation.xsl must be executed before any side service calls in the request processing rule. For response processing rules, the side service calls must be done between the execution of **HPTVExtractResponseCorrelation.xsl** and **HPTVAttachResponseCorrelation.xsl**.

HPTVSideCallCorrelation.xsl provides a named template (**HPTVGetCorrelation**) which can be used to construct the proper HTTP request header.

dp:url-open Example:

```
<xsl:include href="local:///HPTVSideCallCorrelation.xsl"/>
      .
      .
      .
   <xsl:variable name="httpHeaders">
      <xsl:call-template name="HPTVGetCorrelation"/>
   </xsl:variable>

   <xsl:variable name="url-response">
      <dp:url-open target="http://wasapp.rose.hp.com:3011/"
         http-headers="$httpHeaders">
         <soap:Envelope
            xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:echo="http://com/ibm/was/wssample/sei/echo/">
            <soap:Header/>
            <soap:Body>
               <echo:echoStringInput>
                  <echoInput>Echo Test</echoInput>
               </echo:echoStringInput>
            </soap:Body>
         </soap:Envelope>
      </dp:url-open>
   </xsl:variable>
```

dp:soap-call() Example:

```
<xsl:include href="local:///HPTVSideCallCorrelation.xsl"/>
      .
      .
      .
   <xsl:variable name="httpHeaders">
      <header name="Content-Type">application/soap+xml</header>
      <xsl:call-template name="HPTVGetCorrelation"/>
   </xsl:variable>

   <xsl:variable name="soap-request">
      <soap:Envelope
         xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:echo="http://com/ibm/was/wssample/sei/echo/">
         <soap:Header/>
         <soap:Body>
            <echo:echoStringInput>
               <echoInput>Echo Test</echoInput>
            </echo:echoStringInput>
         </soap:Body>
      </soap:Envelope>
   </xsl:variable>

   <xsl:variable name="xml-in"
      select="dp:soap-call('http://wasapp.rose.hp.com:3011/',
         $soap-request/*,'',0,'',$httpHeaders/*)"/>
```

Configuring Loopback Request Rule

In the rare case that the backside is skipped using var://service/mpgw/skip-backside and no response rule is defined, all instrumenting stylesheets need to be added to the request rule. **HPTVRequestCorrelation.xsl** should be added as close to the beginning of the request rule as possible followed by existing actions in the processing rule. At the end of the request rule, **HPTVExtractResponseCorrelation.xsl** should be added followed immediately by **HPTVAttachResponseCorrelation.xsl**.

Task 4: Configure the DataPower Agent to Transform the WS-Management Events (Optional)

There are two different ways DataPower monitoring can be configured in TransactionVision:

➤ "No Transformation on DataPower" on page 247

➤ "Transformation on DataPower" on page 247

No Transformation on DataPower

DataPower WS-Management events are sent directly from the monitored DataPower appliance to the TransactionVision **Analyzer** with minimal manipulation within DataPower (there is no additional configuration). These WS-Management events are then transformed within the **Analyzer** to TransactionVision events to be processed and analyzed.

With this configuration, minimal overhead is added to DataPower to monitor and deliver events to TransactionVision. However, additional overhead is needed in the **Analyzer** to process the DataPower events. This could be a concern as the **Analyzer** can, inherently, become a bottleneck for incoming DataPower events when multiple DataPower appliances are monitored.

If you want to use this option, skip to the next optional step.

Transformation on DataPower

DataPower WS-Management events are transformed to the TransactionVision User Event format within the TransactionVision Multi-Protocol Gateway before sending the event to the **Analyzer**.

With this configuration, additional overhead is added to DataPower to monitor and deliver events to TransactionVision. This may be desired when multiple DataPower appliances are being monitored in order to better distribute the workload of transforming the events to TransactionVision events rather than having the **Analyzer** do all of the transformation work.

If this option is the better solution for the given environment, follow these additional configuration steps:

➤ "Step 1: Add TransactionVision's DataPowerEvents.xsl Transformation to the Multi-Protocol Gateway Processing Policy" on page 248

➤ "Step 2: Define TransactionVision Multi-Protocol Gateway HTTP Header Injections" on page 249

Step 1: Add TransactionVision's DataPowerEvents.xsl Transformation to the Multi-Protocol Gateway Processing Policy

1. From the DataPower Control Panel in the **HPTVMonitoring** domain, click the **Multi-Protocol Gateway** icon.
2. Click on **hptv-wsman-subscriber**.
3. Click on the **Edit selected Processing Policy** button to edit **hptv-wsman-subscriber-policy**.
4. In the **Client to Server** rule, drag a new Transform just after the first Transform (**hptv-wsman-subscriber-policy_req-rule_xform_0**).
5. Double-click on the new Transform to edit it.
6. Select **INPUT** for Input and **PIPE** for Output.
7. For the **Processing Control** file, click **Upload** to upload the **DataPowerEvents.xsl** transformation file to the **local:** folder.
8. Click **Browse** to select the local **DataPowerEvents.xsl** file.

   DataPowerEvents.xsl is installed with the TransactionVision **Processing Server** under the **config/services** directory of the TransactionVision installation directory. It may be needed to transfer this file to the local host if the **Processing Server** was installed on a different host.
9. Click **Upload** to upload **DataPowerEvents.xsl** to the **local:** directory on DataPower and click **Continue** on the confirmation page.
10. Click **Done** on the **Configure Transform Action** page.
11. Double-click the **Results** icon in the **Client to Server** rule to edit it.
12. Change the Results **Input** to **PIPE** and click **Done**.

13 Click **Apply Policy** to save the new policy changes and close the Policy Edit window.

14 Click **Apply** on the Configure Multi-Protocol Gateway page.

15 Click **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

Step 2: Define TransactionVision Multi-Protocol Gateway HTTP Header Injections

1 From the DataPower Control Panel in the HPTVMonitoring domain, click the Multi-Protocol Gateway icon.

2 Click on **hptv-wsman-subscriber** and click on the **Headers** tab.

3 Add the following Header Injection Parameters. Notice the X-JMS-CorrelationID is different, in this case.

Direction	Header Name	Header Value
Back	X-JMS-CorrelationID	TVisionUserEvents
Back	X-JMS-DestinationQueue	HTTP.EVENT.QUEUE
Back	X-JMS-MessageType	TEXT
Back	X-JMS-DeliveryMode	NON_PERSISTENT or PERSISTENT

Note: If you want events to be persisted on the event queue when the SonicMQ broker is restarted, set X-JMS-DeliveryMode to NON_PERSISTENT or PERSISTENT.

---

4 Click **Apply**, then **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

Configuring the WS-Management Agent

The TransactionVision DataPower Agent uses the WS-Management Agent to receive WS-Management events which ultimately become TransactionVision DataPower events. Since the WS-Management Agent buffer is limited by the resources of the DataPower appliance, it is important to review and plan the settings described in this chapter to avoid events from being discarded by the WS-Management Agent if limits are reached from high event volume.

The DataPower WS-Management Agent provides a standard WS-Management event source. The TransactionVision DataPower Agent subscribes to this WS-Management event source. The subscription defines events to be published to the **hptv-wsman-subscriber** Multi-Protocol Gateway which ultimately delivers the events to the TransactionVision Analyzer through the Processing Server's SonicMQ Broker.

WS-Management events are delivered asynchronously, and, therefore, are buffered in the appliance's memory until they get published to the **hptv-wsman-subscriber** Multi-Protocol Gateway. Since memory on the appliance is limited, the intermediate WS-Management Agent's event buffer also has a limited size. With such a limitation, it has to be understood that with very high volumes of traffic, the asynchronous delivery may not be able to keep up, and the event buffer could reach its limits. When this occurs, the WS-Management Agent will discard any events it cannot fit in the full event buffer, and these events will not be recorded by TransactionVision.

There are several settings for configuring both the WS-Management Agent and the subscription the TransactionVision DataPower Agent makes to it. When tuned properly for the types of events expected in an environment, the chances of events being discarded from event buffer overflow can be reduced.

This section includes the following topics:

WS-Management Agent Settings

Each DataPower domain contains a WS-Management Agent. As described, the WS-Management Agent maintains an event buffer to intermediately retain events as they're being asynchronously delivered to the subscribers (TransactionVision DataPower Agent's hptv-wsman-subscriber).

The WS-Management Agent settings can be found under **Objects** > **Device Management** > **Web Services Management** within the DataPower Web user interface. Each setting is described as follows:

Maximum Record Size

The maximum number of WS-Management event records allowed to be stored within the event buffer. If the maximum records are stored in the event buffer before they can be delivered to the subscribers, additional WS-Management events will be discarded.

- Default: **3000 records**
- Recommended for TransactionVision: **1000000 records**

Maximum Memory Usage

The maximum amount of memory (in KB) allowed to be used by the event buffer. If the maximum memory usage is reached, additional WS-Management events will be discarded.

- Default: **64000 KB**
- Recommended for TransactionVision: **128000+ KB**

Capture Mode

Selects which WS-Management events to capture for further analysis:

- **None** - Captures no WS-Management events.
- **Faults only** - Captures only WS-Management events that contain SOAP faults.
- **All** - Captures all WS-Management events.

- Default: **Faults only**
- Recommended for TransactionVision: **All**

Buffering Mode

Selects the behavior of the WS-Management Agent when there are no registered consumers of the WS-Management events:

- **Discard** - Discards WS-Management events.
- **Buffer** - Buffers WS-Management events for any future subscribers.

Although buffering reduces the loss of events, it consumes more memory. Buffered events are accumulated until a configured limit (records or memory) is reached. After this threshold, new events are dropped.

The default value of **Discard** is suited to a single subscriber where a low transaction volume is being handled by the appliance. Buffering mode Buffer is a better choice when initially configuring data collection, or when processing high volumes.

In high volume scenarios, setting Buffering Mode to **Buffer** simplifies how subscriptions are handled. When a subscriber collects data, all the WS-Management events that have been written to the WS-Management buffer are collected. With the default setting of **Discard**, complications can occur if a new subscriber replaces a former subscriber, or if high volumes cause the Complete Records Count (seen in **Status** > **Web Service** > **WSM Agent Status**) to reach the configured Maximum Record Size limit. Set Buffering Mode to **Buffer** and collection can proceed.

- Default: **Discard**
- Recommended for TransactionVision: **Buffer**

For more information see IBM Technote:
http://www 01.ibm.com/support/docview.wss?uid=swg21394557.

DataPower Agent's WS-Management Subscription Settings

The TransactionVision DataPower Agent subscribes to the WS-Management Agent to receive and deliver events to the TransactionVision Analyzer. The subscription to the WS-Management Agent contains settings that affect the behavior of event publishing from the WS-Management Agent.

<wsman:MaxElements>

The maximum number of events to batch into a single WS-Management SOAP envelope. This acts much like the TransactionVision Data Collection Filter's Event Packaging Maximum Package Count. The default value MaxElements defined by the DataPower Agent is 50. Consider changing this value based on average event and payload sizes expected in the environment. For example, if very large payloads are expected to be collected, one may consider decreasing the MaxElements to prevent messages sent to the Analyzer from being too large.

To change these settings:

1. Make a backup copy of **local:///generate_subscriptions.xsl** in the HPTVMonitoring domain.
2. Modify the **local:///generate_subscriptions.xsl** file.
3. Locate **<wsman:MaxElements>**.
4. Change the value enclosed in the **<wsman:MaxElements>** tags to the desired value.

Monitoring WS-Management Publishing

DataPower provides status pages within the Web user interface to monitor WS-Management subscribers and the WS-Management Agent:

Status > Web Service > WSM Agent Subscribers

This page shows a list of WS-Management subscribers and details about each subscription. Typically, expect to see only one subscription if the TransactionVision DataPower Agent has been configured to monitor the current domain.

Status > Web Service > WSM Agent Status

This page shows a list of WS-Management Agent runtime statistics. The help page describes each of these statistics.

In the event that lost WS-Management events may be occurring, these status pages can be used to help diagnose the problem.

To diagnose a problem with WS-Management events:

1. Select **Status > Web Service > WSM Agent Status** in the DataPower Web user interface. The WSM Agent Status shows the number of active subscribers and a count of metrics records that were lost or are complete.
2. Make a note of the complete record count; it should have a value less than the defined Maximum Record Size setting on the WS-Management Agent at any given moment. This setting is configurable at **Objects > Device Management > Web Services Management Agent**.
3. Select **Refresh** multiple times on the WSM Agent Status page and monitor the complete record count. When the WS-Management event is published to the hptv-wsman-subscriber Multi-Protocol Gateway, the complete record count should reset to a low number, and increase again with each refresh of this page until the next WS-Management event is published.

4 Select **Objects** > **Device Management** > **Web Services Management Agent** and change Buffering Mode to **Buffer** to resolve either of two issues:

- If active subscribers and complete records have a zero value, the lost record count may increase with each refresh of this page. Records that are candidates for data collection are being discarded because there are no subscribers to collect them.
- A non-zero complete record count that remains static indicates that records are not being collected. A complete record count of the defined Maximum Record Size setting on the WS-Management Agent may indicate there was an issue associated with high volume. WS-Management Agent settings may have to be adjusted to accommodate for the high volume. An Active Subscribers count greater than one may indicate that managing multiple subscribers may be a contributor to the problem. You can review page **Status** > **Management** > **WSM Subscriber Status** to see details on the subscribers.

For more information see IBM Technote:
http://www 01.ibm.com/support/docview.wss?uid=swg21394557.

Event and Payload Trimming and Filtering

User payload passed through DataPower can be large, and there may not be a need to collect the entire payload in TransactionVision. Many times, only a few fields are of interest for business transaction correlation and attributes.

To minimize the data traffic between the monitored DataPower appliances and the TransactionVision **Analyzer**, you may want to trim down or filter specific fields in the message payload before sending the event to the **Analyzer**.

To implement this, a custom transformation needs to be written specific to your data. Such a transformation can be placed first thing in the TransactionVision Multi-Protocol Gateway or after the DataPowerEvents.xsl (if used) depending on if the custom transformation accepts a WS-Management XML format or a TransactionVision User Event XML format.

Typically such a custom transformation would simply select for known important XPaths within the different payload formats expected to pass through the monitored DataPower services.

Note: The custom transformation should consider all four payloads-frontend request/response and backend request/response.

HPTVStripFields.xsl Example Stylesheet

HPTVStripFields.xsl is an example stylesheet provided with TransactionVision to demonstrate Payload Trimming. This stylesheet is located in the integrations directory of the **Processing Server** installation. It can be used as a template to customize a user-specific stylesheet for trimming fields from the request/response payloads (for example, inline attachments).

HPTVStripFields.xsl first extracts the service-port field and removes the "{URI}" section for a simpler value to match against. This value is then passed to all templates for matching against.

For each possible XPath to be stripped from the event, a new template needs to be added to **HPTVStripFields.xsl**. **HPTVStripFields.xsl** includes a few examples, and are in two separate sections of the file:

- Global/Common

  These templates are for stripping fields no matter what the matching value (service) is.

- Match Specific

  These templates receive the matching value as a parameter. They test against the matching value before stripping the field.

Deploying HPTVStripFields.xsl

To use HPTVStripFields.xsl to trim fields from the request/response payloads in the events, follow these steps:

1. Modify a copy of **HPTVStripFields.xsl** with XPaths and service names specific to the DataPower services and application payload expected in the monitored environment.
2. Upload the custom **HPTVStripFields.xml** to the **HPTVMonitoring** domain.
3. Edit the **hptv-wsman-subscriber-policy** in the **hptv-wsman-subscriber Multi-Protocol Gateway**.
4. In the **Request Rule**, add a Transform after the first existing Transform (**set-route.xsl**).
5. Define the newly added Transform to use the uploaded custom **HPTVStripFields.xsl** with **Input** set to **INPUT** and **Output** set to **PIPE**.
6. Be sure following action in the policy rule has **Input** set to **PIPE**.
7. Click **Apply Policy** and **Save Config**.

Troubleshooting the IBM WebSphere DataPower Agent

This section describes troubleshooting and limitations for DataPower Agent.

➤ "Troubleshooting Checklist" on page 257

➤ "Diagnosing Problems With the DataPower Agent" on page 258

For troubleshooting DataPower Event Correlation, see "Troubleshooting" in "Analyzers" in *Using Transaction Management*.

Troubleshooting Checklist

- Be sure all required HP TransactionVision stylesheets have been added to all processing rules of the services being monitored.
- Check the DataPower logs of the service in question. The HP TransactionVision stylesheets will produce errors under the **HPTV** category if they are executed in the wrong place or in the wrong order.

- For Multi-Protocol Gateways, be sure the service identifying attributes have been set as shown in the example **HPTVSetWSMAttributes.xsl** stylesheet. **HPTVRequestCorrelation.xsl** produces an error in the DataPower logs if the identifying attributes have not been properly set.
- Be sure any and all side service calls are properly instrumented. A multistep probe can be used on the service being called to ensure TransactionVision correlation information is being properly passed in the request header. All TransactionVision header fields begin with "X-HPTV-".

Diagnosing Problems With the DataPower Agent

There are a few services and logging facilities that can be enabled in DataPower to help diagnose problems with the DataPower Agent. Note, it is not suggested to turn any of these on except temporarily to troubleshoot any problems within the DataPower Agent.

Debug Logging

In order to troubleshoot problems with the DataPower Agent, you can enable debug logging in the **HPTVMonitoring** domain.

To enable debug logging, follow these steps:

1. Switch to the **HPTVMonitoring** domain.
2. From the DataPower Control Panel in the **HPTVMonitoring** domain, click the **Troubleshooting** icon.
3. Under **Set Log Level**, set **Log Level** to **debug**.
4. Click **Set Log Level**.
5. Click **Confirm** and **Close** in the popup window.
6. Click **Save Config** in the upper right corner of the DataPower Web UI to finalize and save all configuration changes.

When debug logging is enabled, extra informational and debug logs can be seen when clicking on the View Logs icon of the DataPower Control Panel within the **HPTVMonitoring** domain.

Multi-Protocol Gateway Multistep Probe

DataPower provides a mechanism for tracing steps through policy rules called a Multistep Probe. A Multistep Probe can be enabled on the TransactionVision Multi-Protocol Gateway to trace messages passing through its processing rules.

To enable the Multipstep Probe, follow these steps:

1. From the DataPower Control Panel in the **HPTVMonitoring** domain, click the **Multi-Protocol Gateway** icon.
2. Click **hptv-wsman-subscriber**.
3. Click **Show Probe**.
4. In the popup window, click **Enable Probe**.

Alternatively, the Multistep Probe can be enabled/disabled and viewed by clicking on the **Troubleshooting** icon on the DataPower Control Panel, and looking under the **Debug Probe** tab.

With the Multistep Probe enabled, details of every message passing through the TransactionVision Multi-Protocol Gateway is logged at each step within the processing rules. This allows you to examine the state and format of the message as it passes through each action in the processing rule. This can be especially useful when designing a custom payload trimming/filtering transformation for the TransactionVision Multi-Protocol Gateway.

19

Installing and Configuring Agents on NonStop TMF

This chapter includes:

About the NonStop TMF Agent

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 Agent monitors the TMF audited trail database for changes (adds, deletes, and modifies) on Enscribe databases. The NonStop TMF Agent 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 Agent 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 agent installation from the HP Business Service Management product installation disk or from the Downloads page in BSM.

The following table shows the installation file names for the NonStop TMF packages for each supported platform:

Platform	Files
Guardian	HPTVTMFAgent_<version>_ns.zip

Installing the NonStop TMF Agent

To start the agent installation program, perform the following steps:

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 Agent has three TACL macros that are used for startup and shutdown:

- **COLDMON**. Starts the agent for the first time.
- **STRTMON**. Starts the agent after it has been stopped.
- **STOPMON**. Stops the agent.

COLDMON

This macro purges all temporary data used by the agent (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 Agent is installed, and run COLDMON.

After starting COLDMON you are 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 agent to begin searching in a specific audit trail file, input that file, otherwise, simply begin with the first file displayed (select '1').

The agent begins searching for events that fit the filter criteria up to the present. The first time the agent is started; there will not be any events to collect until a filter is configured. The agent will scan through the audit files, and wait for new audit trail records.

STRTMON

When the agent 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 Agent, after it has been stopped, run the TACL macro STRTMON, for example:

```
run STRTMON
```

STOPMON

To stop the NonStop Agent, run the TACL macro STOPMON, for example:

```
run STOPMON
```

Configuring the NonStop TMF Agent

The NonStop TMF Agent 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 agent, the agent will work with all of the values set by the INSTALL macro.

The NonStop Agent 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 Transaction Management*.

Uninstalling the NonStop TMF Agent

To uninstall the NonStop TMF Agent, follow these steps:

1 Stop the agent by running STOPMON.

2 Remove all files in the temporary installation directory.

3 Remove all files in the installation volume/subvolume.

20

Configuring WebSphere MQ Agents

TransactionVision provides two types of WebSphere MQ Agents: WebSphere MQ Agent library and WebSphere MQ API Exit Agents. This chapter describes configuring both types of agents on Windows, UNIX, and i5/OS platforms.

For information about configuring this agent on z/OS, see Chapter 15, "z/OS Components–Operation and Customization."

This chapter includes:

Configuring the WebSphere MQ Agent Library

The WebSphere MQ Agent 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 Agent library. Before you can use TransactionVision to record event data for an application, you must configure the application environment to load the Agent library instead of the standard WebSphere MQ library.

This section includes:

Caution: When using the Agent 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 Agent Library for 64-bit applications.

UNIX and Windows Platforms

On UNIX and Windows platforms, you must add the directory location of the Agent 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 agent 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

Platform	Environment Variable	Default Directory
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 agent installation locations. If the agent was installed in a location other than the default, specify the directory location for the agent executable.

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

Platform	Environment Variable	Default Directory
Windows	PATH	C:\Program Files\HP\TransactionVision\lib64
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 Agent 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 Transaction Management* for details.

Caution: 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*.

To run applications to be monitored by agents 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 CICS

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

i5/OS Platforms

On the i5/OS platform, the main agent 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 agent 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 agent 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 agent. 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 Agent library TVSENSOR. The Agent library must precede the WebSphere MQ library QMQM in the library list in order to use TransactionVision.

Note: 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 Agent Logging

On some operating systems, there is no additional work to obtain error and trace logging from the WebSphere MQ Agents. However, on UNIX platforms, syslogd may need to be configured to log the logging facility used by the WebSphere MQ Agents. For details on WebSphere MQ Agent logging on a UNIX platform, see Chapter 22, "Configuring Agent Logging."

If you want to disable logging in the WebSphere MQ Agent, TVISION_DISABLE_LOGGING can be defined in the environment with any value. If this environment variable is set at the monitored application startup, the WebSphere MQ Agent disables all logging.

Setting the Configuration Queue Name

By default, agents 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 agents 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 agent configuration queue specified in the communication link for all processes that use the agent.

Setting the Configuration Queue Check Interval

By default, agents 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 an agent, 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 agents. 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 Transaction Management*.

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 Agent

The WebSphere MQ API Exit Agent 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 agent 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 agent is running in the amqsput program address space, not the queue manager space.

You can use the WebSphere MQ API Exit Agent 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 Agent differs from the WebSphere MQ Agent Library in the following ways:

- There is no need to disable FASTPATH_BINDING (see "WebSphere MQ Agents and FASTPATH_BINDING" on page 283 chapter for more information).
- The completion and reason codes for MQDISC calls are not collected by the API Exit Agent 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 Agent collects some TransactionVision internal events generated from WebSphere MQ (WMQ) calls made by the Analyzer, and also internal WMQ events from Java Agents using a client connection to the listener.

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

This section includes:

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

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

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

Configuring Symbolic LInks to the WebSphere MQ API Exit Agent

Because TransactionVision supports both 32-bit and 64-bit versions of WebSphere MQ, you must create symbolic links to the appropriate shared object/DLLs.

For all versions of WebSphere MQ, use the following commands to link the agent:

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 on 64-bit operating systems, also create symbolic links to the 64-bit API Exit Agent libraries as follows:

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 Agent 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 Agent 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.

Caution: Do not specify a path; the module path is determined by the link command you used to link to the correct module (see "Configuring Symbolic LInks to the WebSphere MQ API Exit Agent" on page 276). The location depends on whether you use 32-bit or 64-bit WebSphere MQ libraries. The 32-bit module is located in **<TVISION_HOME>/lib**, while the 64-bit module is located in **<TVISION_HOME>/lib64**.

➤ Data=TVQ=queue_name

To set the queue object names for which the agent 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 agent. 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 282.

➤ 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 agent configuration per queue manager (qm.ini).

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

Configuring the API Exit Agent on Windows Platforms

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

A 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 Agent on Distributed and i5/OS Platforms" on page 275 for a description of required stanzas and attribute values.

When configuring the API Exit Agent on Windows 64-bit, copy the 32-bit and 64-bit API Exit Agent libraries to the appropriate WebSphere MQ exit folders:

```
copy C:\Program Files\HP\TransactionVision\lib\tvisionapiexit C:\Program
Files\IBM\WebSphere MQ\exits
copy C:\Program Files\HP\TransactionVision\lib64\tvisionapiexit C:\Program
Files\IBM\WebSphere MQ\exits64
```

When referring to the API Exit library in the API Exit definition, use only the library name "tvisionapiexit" without the path. WebSphere MQ will load the appropriate API Exit from the default exits directories.

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

Identifying Programs to Monitor

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

- wmq_exit_agent.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.
- wmq_exit_agent.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 Agent and contains some default programs from which events are not collected by the API Exit Agent, 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 wmq_exit_agent.allow and wmq_exit_agent.deny exist, the agent ignores wmq_exit_agent.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 wmq_exit_agent.deny.

Note: When using the WebSphere MQ API Exit Agent, if the wmq_exit_agent.allow file exists and is left empty, no programs will be monitored.

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

```
# File: wmq_exit_agent.allow
# Description: Only collect from WebSphere MQ Listeners
amqcrsta
amqrmppa
runmqlsr
```

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

```
# File: wmq_exit_agent.deny
# Description: Collect any program except those that
# start with “amq”
amq*
```

Discarding WebSphere MQ Events on TransactionVision Queues

By default, the agent 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 Transaction Management*.

WebSphere MQ Agents and FASTPATH_BINDING

For the standard WebSphere MQ Library Agent 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 Agent library instead of WebSphere MQ, the secondary DLL tries to call internal symbols that are not available.

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

Using Agents with WebSphere MQ Samples

If you want to use agents 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 Agent library instead of the standard WebSphere MQ library. This is because the WMQ libraries reside in this same folder and take precedence over the Agent 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 Agent library.
- When using the WebSphere MQ sample amqsgbr, do not use the agent event queue as the first parameter.

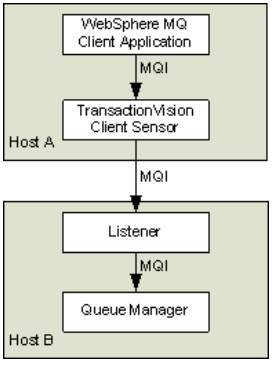
WebSphere MQ Client Application Monitoring

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.

This section includes:

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 agent will be installed on the same host where the client application runs. The agent 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.

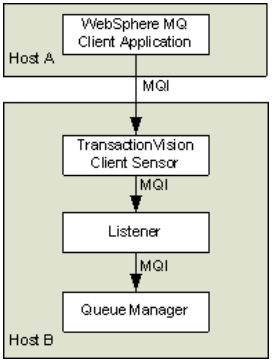
The trace information generated will be based on the client application context. This means information such as program name, program instance, and host name, will be related to the client application directly.

This monitoring scheme requires a client agent installed on each machine where WebSphere MQ client applications run. Moreover, the client agent is capable of monitoring any applications making use of the C language WebSphere MQ client runtime binding. In other words, the client agent 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 Agent and is not supported with the library agent. The following diagram illustrates how the agent works in a centralized monitoring environment:



In this case, the agent is deployed on the host where the listener process and queue manager reside. Instead of direct monitoring of the client application, the agent monitors the listener program instead.

The agent 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 described 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 agent deployment can be centralized around the host machines where the queue manager runs. Unlike the distributed approach, no client agents 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 agent 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 Agents with a client connection to a queue manager through a listener, which is being monitored with the TransactionVision WebSphere MQ Agent, internal TransactionVision events will be captured. It is recommended to either use a separate unmonitored listener for the Servlet, JMS, or EJB Agents or use server binding connections from these agents. 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 agent on each client host	Server agent per queue manager host
Monitors applications using WebSphere MQ C binding	Server agent 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 agents on each host where WebSphere MQ client applications run. Follow the standard agent installation instructions in "Installing and Configuring the Java Agent on Windows" on page 122.

For centralized monitoring, install server agents 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 agent monitoring the listener program will report activities originated from all clients it services, including those clients that may be monitored by client agents 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 agent is disabled.

21

Configuring the Proxy Agent

This chapter includes:

About the Proxy Agent

The TransactionVision Proxy Agent enables TransactionVision to provide a basic level of correlation of business transactions into process that are not monitored using TransactionVision Agents. Some examples of the appropriate applications of the Proxy Agent 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 Agent (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 unmonitored applications are participating in the business transaction, the Proxy Agent enables TransactionVision to provide limited information about the entire business transaction.

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

Application Requirements

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

- The application monitored by the agent must maintain the message ID and correlation IDs in the MQMD.
- The application monitored by the agent 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 Agent

The Proxy Agent 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.

This section includes the following:

Example

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.

Element	Required?	Attributes	Description
Program	Yes	name path	The name and path of the proxy program.
Host	Yes	name os	The name and operating system of the host where the proxy program runs.
Retrieve	No	queue	Causes the Proxy Agent to check the given queue object against its definition rather than the object defined in <Reply>. This element allows the agent 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 Agent to match the message Id of the MQPUT with the correlation Id of the MQGET
matchCorrelIdToMsgId	Causes the Proxy Agent 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 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 *Using Transaction Management*.

22

Configuring Agent Logging

This chapter includes:

Log Files

Log files are different for each agent type.

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 Agents 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 Agents

By default, the WebSphere MQ agents 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 agent 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.

If you want to disable logging in the WebSphere MQ Agent, TVISION_DISABLE_LOGGING can be defined in the environment with any value. If this environment variable is set at the monitored application startup, the WebSphere MQ Agent disables all logging.

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.

This section includes:

Log File Naming

Using the defaults, when a log file (for example, the agent 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 agents. 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, perform the following steps:

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 agent 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 agents to debug agent configuration issues. Trace information for the WebSphere MQ Agent 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 agent trace logging in the communication link, see "Communication Links" in *Using Transaction Management*.

Configuring Separate Log Files for Multiple Agent 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 agent 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 agents 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. This section includes:

➤ "Windows Event Appender" on page 300

➤ "UNIX Event Appender" on page 301

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.

Part IV

Security

23

Configuring Security

This chapter provides information about the security setup procedures of TransactionVision.

This chapter includes:

About Security in TransactionVision

Security in TransactionVision is managed by the following tasks:

- Enabling SSL between TransactionVision and BSM components.
- Controlling user access to Transaction Management reports and topologies, and to TransactionVision components.
- Securing the Processing Server, configuration files, and database.

  The simplest mechanism to control access to a Processing Server 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.

Managing User Permissions

User permissions determine what operations a user can perform in the Transaction Management application and administration pages. User or group permissions can be set by assigning a predefined role or by granting individual permissions on specific resources. The predefined roles relevant to Transaction Management are described below.

BSM enables you to fine tune user permissions by applying permissions at the resource level. All of the resources on which permissions can be applied have been identified and categorized in a hierarchical tree, representing the BSM platform. Transaction Management-specific resources are described below.

The users permissions in a particular session within BSM are determined by the user's login and authentication. User permissions are set up and maintained by accessing **Admin** > **Platform** > **Users and Permissions** > **User Management**. For information about the related workflow, see *Platform Administration*. Permission changes go into effect the next time the user logs in.

This section includes:

Transaction Management Resources and Operations

Within the Transaction Management context, the following resources and associated operations are available:

Resource	Operations that can be granted
TransactionVision Processing Servers	**Change**. User is allowed to modify the configuration of any existing Processing Server as well as add and delete Processing Servers. **Full Control**. User is allowed **Change** operation as well as ability to grant and revoke operations to other users.
TransactionVision Analyzer	**Change**. User is allowed to modify the configuration of any existing Analyzer as well as add and delete Analyzers. **Execute**. User is allowed to start/stop all Analyzers. **Full Control**. User is allowed **Change** and **Execute** operation as well as ability to grant and revoke operations to other users.
TransactionVision Job Manager	**Change**. User is allowed to modify the configuration of any existing Job Manager as well as add and delete Job Managers. **Execute**. User is allowed to start/stop all Job Managers. **Full Control**. User is allowed **Change** and **Execute** operation as well as ability to grant and revoke operations to other users.
TransactionVision Query Engine	**Change**. User is allowed to modify the configuration of any existing Query Engine as well as add and delete Query Engines. **Execute**. User is allowed to start/stop all Query Engines. **Full Control**. User is allowed **Change** and **Execute** operations well as ability to grant and revoke operations to other users.
Administration	**Change**. The Transaction Management Administration page is enabled for the user. **Full Control**. User is allowed **Change** operation as well as ability to grant and revoke operations to other users.
User Data	See "About User Data" on page 310. **View**. User is allowed to view all user data. **Full Control**. User is allowed **View** operations as well as ability to grant and revoke operations to other users.

Resource	Operations that can be granted
User-created Reports	**View**. User is allowed to view all user-created reports. **Full Control**. User is allowed **View** operations as well as ability to grant and revoke operations to other users.
Applications	**Add.** User is allowed to create new application instances. **Change**. User is allowed to create new business transactions under any application as well as modify monitoring and tracing configuration for any business transaction. **View**. User is allowed to view transaction data associated with any application in the reports. **Full Control**. User is allowed **Add**, **Change** and **View** operations as well as ability to grant and revoke operations to other users.
An application instance	**Change**. User is allowed to create new business transactions under an application as well as modify monitoring and tracing configuration for any of its business transactions. **View**. User is allowed to view transaction data associated with the application in the reports. **Full Control**. User is allowed **Change** and **View** operations as well as ability to grant and revoke operations to other users.

BSM Predefined Roles Relevant to Transaction Management

Any role can be created and used to manage access to Transaction Management resources. However, BSM has these built-in roles that are relevant to Transaction Management:

Role	Operations
Transaction Management Administrator	**Full Control** on all Transaction Management resources listed in the previous table except User Data. [1, 2]
Transaction Management Operator	**View** on all Transaction Management resources listed in the previous table. [2]
Transaction Management User	**View** on all Transaction Management resources listed in the previous table. [2]
(BSM) Superuser	**Full Control** on all Transaction Management resources listed in the previous table.
(BSM) Administrator	**Add** on the application resource and **Full Control** on any application instances added.
(BSM) System Modifier	**View** and **Change** on all Transaction Management resources listed in the previous table.
(BSM) System Viewer	**View** on all Transaction Management resources listed in the previous table.

[1] Due to the sensitive nature of User Data and to minimize the possibility of accidentally granting access, only the (BSM) Administrator has User Data access (**Full Control** permission).

[2] These roles do not provide user access to the Transaction Over Time, Transaction Summary, Transaction Tracking, and Aggregated Topology. See "Additional Permission Requirements for Some Reports and Topologies" on page 311.

About User Data

Some transaction and event content collected by TransactionVision agents may contain data that is considered sensitive to the business. This might include items such as a credit card 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 Transaction Management 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 occurs:

- Event Details for an event displays: **You do not have permission to view user data**.
- Transaction Tracking report does not display custom columns.
- Transaction Detail page does not display custom data in the Summary section.
- When defining transaction tracing rules, the Create or Edit Match Condition dialog does not allow selection of a transaction and does not populate the Values pane with transaction values if classification is based on user data.
- Transaction Tracing Rule Wizards and the Transaction Monitoring Wizard are disabled.

Note: The (BSM) Administrator role has rights to grant and revoke all Transaction Management permissions, thus Business Service Management Administrator is effectively granted access to all TransactionVision resources. If you require extra prevention to restrict User Data access but you still have a need for a user to be able to access the Transaction Management Administration tab, then you can assign the user the Transaction Management Administrator role rather than the (BSM) Administrator role.

Additional Permission Requirements for Some Reports and Topologies

The following reports and topologies contain data from the RTSM and are therefore subject to additional RTSM **View** permissions requirements:

- Transaction Summary
- Transaction Tracking
- Transaction Over Time
- Aggregated Topology

If you grant permissions only by assigning any of the three Transaction Management roles and/or grant permissions on individual Transaction Management resources, users will not have permission to view these reports and topology.

You must also grant the **View** operation privilege to the Business Transactions resource (in the RTSM permissions context) to allow users access to these reports and topologies.

Using the BSM roles does not require the additional RTSM Business Transactions resource **View** permission.

Securing With Light Weight Single Sign-On (LW-SSO)

To guarantee secure access to Transaction Management reports and topologies, BSM uses light weight single sign-on (LW-SSO), which is enabled by default. The security tokens used by LW-SSO are passed between TransactionVision Processing Servers and Business Service Management Gateway Servers to ensure that only correctly authenticated users access the content they are authorized to access.

All the TransactionVision-specific processes (Analyzers, Job Managers, Query Engines) on a Processing Server are controlled through Web services from BSM, which are secured by the LW-SSO configuration that is set up in BSM.

When to Add the Processing Server Domain in the Single Sign-On Configuration

If the Processing Server is on a different domain from the Gateway Server domain, you need to add the Processing Server domain to the list of Protected Domains on the BSM user interface Single Sign-On page. You must add the domain before you synchronize the Processing Server on the Transaction Management user interface. For details on synchronizing the Processing Server, see the "Processing Servers" chapter in *Using Transaction Management*.

To add the Processing Server domain in the BSM configuration, select **Admin > Platform > Users and Permissions > Authentication Management**, click the **Configure** button to open the **Authentication Wizard**, and select the **Single Sign-On** option. Lightweight is the default authentication mode. Add the domain in the Trusted Hosts/Domains table and click **Finish**. For detailed instructions on adding the domain, see "Single Sign-On Page" in *Platform Administration* or click **Help > Help on this page** when you are viewing the Single Sign-On page.

When to Disable LW-SSO

In some pre-production deployments, LW-SSO requirements cannot be met. LW-SSO requires that the TransactionVision Processing Servers and/or Business Service Management servers are accessed using a fully-qualified domain name. But, in some cases, using a fully qualified host name for accessing these components is not possible. In these cases LW-SSO in Business Service Management can be disabled.

TransactionVision can work when LW-SSO is disabled—however doing so disables secure access to the TransactionVision Processing Server. When LW-SSO is disabled in BSM, you must set **enableLWSSOFramework="false"** in the **<TVISION_HOME>/config/ui/lwssofmconf.xml** file and restart the Processing Server.

Note: Disabling LW-SSO causes a vulnerability that can allow authentication to be bypassed. This is not a recommended approach if you have sensitive data or need to ensure that access to Transaction Management data and administration is only possible by users with the correct authorizations.

Basic Authentication Configuration

If the BSM Gateway server is configured with basic authentication, the following web resources need to be unprotected to allow the TransactionVision Processing Server access:

/topaz/services/technical/time

/ucmdb-api

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 the Processing Server configuration for each Processing Server that accesses the database.

You will be asked to supply the database user name and password. Both will be stored in the configuration file in the internal database. The underlying mechanisms for user authentication can be different for each database product. TransactionVision supports database authentication for DB2 and Oracle, and NTLM and SQL Server Authentication for SQL Server.

When using Windows Authentication for SQL Server, you must grant access for the Local System accounts so that the Processing Server hosts can access the TransactionVision SQL Server database. The Local System account can be added using **DOMAINNAME\HOSTNAME$**. If the Processing Server services are distributed across hosts, you need to add the hosts for the Query Manager, Job Manager, and Analyzer.

Database Privileges

To run TransactionVision successfully in the given database environment, you need 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, INSERT, UPDATE, and DELETE privileges on all tables in the Analyzer schemas
- CREATE TABLE and CREATE INDEX privileges for creating the Analyzer tables that store the collected event data when creating a new Analyzer in the Transaction Management Administration pages

If a dedicated database is used for TransactionVision, it is often adequate to choose a user with predefined, sufficient database authority (such as a user with the DBA role) for accessing the database. If this is not possible, the above listed privileges need to be granted specifically.

The CREATE TABLE/ CREATE INDEX privileges are not mandatory for running TransactionVision once the tables have been created. Therefore, it is possible to avoid granting these privileges by having a database DBA create the Analyzer tables manually.

The utility **CreateSqlScript** can generate the necessary SQL scripts to create the Analyzer tables, as well as scripts that grant the required access privileges for the configured user to the Analyzer tables:

CreateSqlScript -create -schema SCHEMA

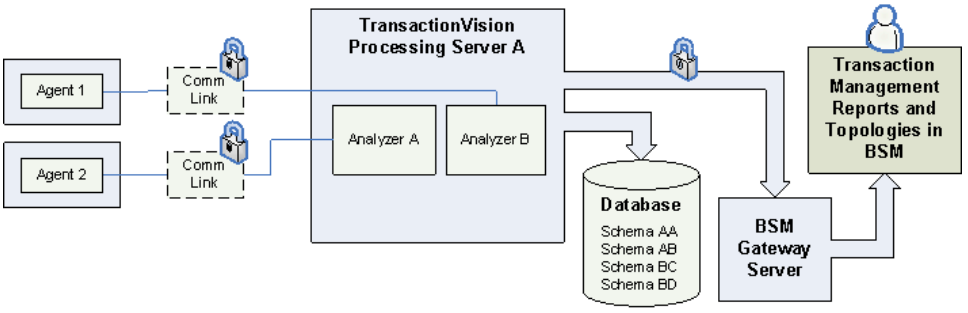
CreateSqlScript -grant -schema SCHEMA

If the configured database user does not have the CREATE TABLE/CREATE INDEX privileges, you cannot create the tables for a new Analyzer from within the Transaction Management Administration pages. Instead, you need to create the tables manually before the Analyzer is created.

Securing with SSL

If the TransactionVision environment includes sensitive data being sent between the components, you can enable SSL for each communication path.

TransactionVision Processing Servers communicate with BSM Gateway Servers and with the communication links collecting events from agents. Each of these data paths are eligible for SSL:



For information about enabling SSL on the data path between Processing Servers and BSM Gateway Servers, see the *HP Business Service Management Hardening Guide* PDF.

Information about enabling SSL on supported messaging middleware and the components communicating across it are described in the following sections. Several sections also describe how to configure appropriate communication link definitions for use with SSL:

- "Enabling SSL for the Messaging Middleware Provider" on page 317
- "Enabling SSL for the Agent" on page 326
- "Enabling SSL for the SonicMQ Communication Link" on page 335
- "Enabling SSL for the HTTP and RUM Communication Links" on page 336
- "Enabling SSL Communication for Events Reported to BPI" on page 337
- "Enabling Authentication in TransactionVision SonicMQ" on page 337

Enabling SSL for the Messaging Middleware Provider

The procedure is specific to the messaging middleware provider used in your deployment environment.

SonicMQ Broker Bundled with TransactionVision

The SonicMQ bundled and installed with the TransactionVision Analyzer contains default acceptors (TV_SSL_ACCEPTOR and SECURE_RUM_ACCEPTOR) that are configured to accept SSL communication. However, some configuration is required for client authentication and the use of your own trusted certificates.

The SSL handshake is based on a mathematical concept called a *one way algorithm*. It consists of two values where if a calculation is done with the first value and a random number, you need to apply the second value to the result to get back to the original random number. In SSL, those two values are known as a *key pair*: one public and the other private.

If you are given a public key (certificate), you can encrypt anything with it, but the only person that can read that value is the person with the private key. Therefore, when a client connects to the SonicMQ broker, the client must have the public key (certificate) from the broker's key pair. The broker shows that during the handshake. When the client is configured to do *trust* checking, it validates that it already knows the broker's public key and has it in its truststore. During the SSL handshake, the client creates a random number, encrypts it with the broker's public key and sends the encrypted result to the broker. The broker uses its private key, which is known only to the broker and is held in the keystore, to reply in a way that the client can *know* it is really talking to the owner of the key pair.

If Client Authentication (mutual SSL) is being done by the broker, the client needs to present its public key and the process is repeated in the opposite direction. Truststores contain only public keys (certificates). When setting up a third-party client to talk to the broker with Client Authentication, the broker's truststore must contain the public key of that client. The client needs to be configured to have a truststore containing the public key from the broker's key pair.

The procedure that follows uses the keytool Java utility to generate example public and private key pairs in which the public key is wrapped in a self-signed certificate. For more information, see http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html. Alternatively, you can use a CA (Certificate Authority).

The following steps provide an example of the process of creating key pairs for the SonicMQ broker and a client for use in a mutual SSL connection (Client Authentication).

To create key pairs for the Sonic MQ broker and client:

1 Generate the server key store from **%SONIC_HOME%/MQ7.6/certs**. The **server.jks** file is delivered with the TransactionVision Analyzer install to allow for the creation of the default TV_SSL_ACCEPTOR and SECURE_RUM_ACCEPTOR acceptors.

The following example shows how the keystore was generated and provides information if you would like to create your own keystore. The alias name should be in lower case to avoid a potential bug in keytool.

```
keytool -genkey -alias tvservercert -keyalg RSA -keystore server.jks
Enter keystore password: changeit
Re-enter new password:  changeit
What is your first and last name?
  [Unknown]:  TransactionVision
What is the name of your organizational unit?
  [Unknown]:  BTM
What is the name of your organization?
  [Unknown]:  TV
What is the name of your City or Locality?
  [Unknown]:  Roseville
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=TransactionVision, OU=BTM, O=TV, L=Roseville, ST=CA, C=US correct?
  [no]:  yes

Enter key password for <tvservercert>
(RETURN if same as keystore password):
```

To verify that the server keystore was generated correctly, run the following:

```
keytool -list -keystore server.jks -v
Enter keystore password: changeit

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: tvservercert
Creation date: Jan 14, 2011
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=TransactionVision, OU=BTM, O=TV, L=Roseville, ST=CA, C=US
Issuer: CN=TransactionVision, OU=BTM, O=TV, L=Roseville, ST=CA, C=US
Serial number: 4d30debc
Valid from: Fri Jan 14 15:39:40 PST 2011 until: Thu Apr 14 16:39:40 PDT 2011
Certificate fingerprints:
        MD5:  75:62:C4:8C:35:3B:22:7E:33:CC:0F:60:E9:F0:E2:7A
        SHA1: 50:C9:AA:CD:5D:63:4A:C6:4D:55:3F:1C:AF:B5:03:3D:BE:D1:B0:E2
        Signature algorithm name: SHA1withRSA
        Version: 3
```

2 Export the public server certificate and save to the CA directory. The **tvservercert.cer** file is also delivered with the TransactionVision Analyzer install to allow for the creation of the default TV_SSL_ACCEPTOR and SECURE_RUM_ACCEPTOR acceptor.

The alias name should be in lower case to avoid a potential bug in keytool.

```
keytool -export -alias tvservercert -keystore server.jks -file CA/tvservercert.cer
Enter keystore password: changeit
Certificate stored in file <CA/tvservercert.cer>
```

3 If you would like to enable clients to use the secure acceptors via SSL then the remaining steps need to be executed from the **%SONIC_HOME%/MQ7.6/certs** directory. Import the public server certificate into a client truststore.

The alias name should be in lower case to avoid a potential bug in keytool.

```
keytool -import -alias tvservercert -file CA/tvservercert.cer -keystore clienttrust.jks
Enter keystore password: changeit
Re-enter new password: changeit
Owner: CN=TransactionVision, OU=BTM, O=TV, L=Roseville, ST=CA, C=US
Issuer: CN=TransactionVision, OU=BTM, O=TV, L=Roseville, ST=CA, C=US
Serial number: 4d30debc
Valid from: Fri Jan 14 15:39:40 PST 2011 until: Thu Apr 14 16:39:40 PDT 2011
Certificate fingerprints:
        MD5:  75:62:C4:8C:35:3B:22:7E:33:CC:0F:60:E9:F0:E2:7A
        SHA1: 50:C9:AA:CD:5D:63:4A:C6:4D:55:3F:1C:AF:B5:03:3D:BE:D1:B0:E2
        Signature algorithm name: SHA1withRSA
        Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

4 Generate the client keystore.

The alias name should be in lower case to avoid a potential bug in keytool.

```
keytool -genkey -alias tvclientcert -keyalg RSA -keystore client.jks
Enter keystore password: changeit
Re-enter new password:  changeit
What is your first and last name?
  [Unknown]:  TransactionVision
What is the name of your organizational unit?
  [Unknown]:  BTM
What is the name of your organization?
  [Unknown]:  TV
What is the name of your City or Locality?
  [Unknown]:  Roseville
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=TransactionVision, OU=BTM, O=TV, L=Roseville, ST=CA, C=US correct?
  [no]:  yes

Enter key password for <tvclientcert>
        (RETURN if same as keystore password):
```

5 Export the public client certificate and save to the CA directory.

The alias name should be in lower case to avoid a potential bug in keytool.

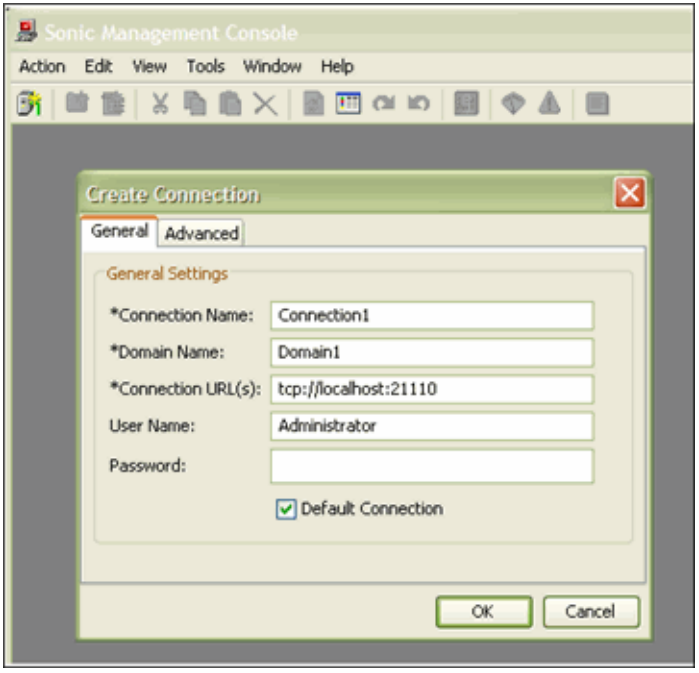
```
keytool -export -alias tvclientcert -keystore client.jks -file CA/tvclientcert.cer
                    Enter keystore password: changeit
                    Certificate stored in file <CA/tvclientcert.cer>
```

6 Generate the server truststore by importing the public client certificate.

The alias name should be in lower case to avoid a potential bug in keytool.

```
keytool -import -alias tvclientcert -keystore servertrust.jks -file CA/tvclientcert.cer
Enter keystore password: changeit
Re-enter new password: changeit
Owner: CN=TransactionVision, OU=BTM, O=TV, L=Roseville, ST=CA, C=US
Issuer: CN=TransactionVision, OU=BTM, O=TV, L=Roseville, ST=CA, C=US
Serial number: 4d35e8d0
Valid from: Tue Jan 18 11:24:00 PST 2011 until: Mon Apr 18 12:24:00 PDT 2011
Certificate fingerprints:
        MD5:  76:F4:EF:3B:4B:FE:45:68:31:9F:36:1E:C7:2A:30:0D
        SHA1: 7C:85:3F:10:1C:15:4C:BF:B2:2D:59:74:E3:A1:22:CA:6C:8B:75:C1
        Signature algorithm name: SHA1withRSA
        Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

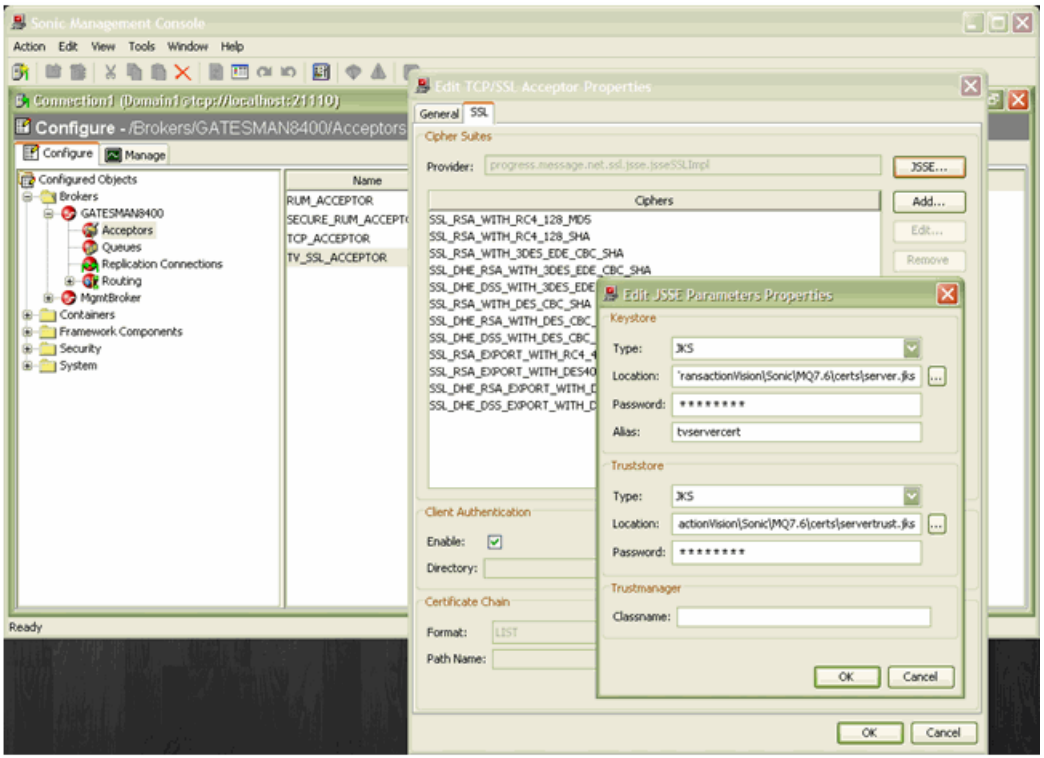
7 Use the Sonic Management Console (**%SONIC_HOME%/MQ7.6/bin/startmc.bat**) to set truststore and keystore information on the SonicMQ broker.



8 Enable Client Authentication as follows:

a Place a check in the **Client Authentication Enabled** check box.

b Add the truststore to the TV_SSL_ACCEPTOR and SECURE_RUM_ACCEPTOR acceptors by pressing the **JSSE** button on each acceptors SSL property tab.



Keystore Location: <**your path**>**certs****server.jks**
Keystore Password: **changeit**
Alias: **tvservercert**
Truststore Location: <**your path**>**certs****servertrust.jks**
Truststore Password: **changeit**

c Restart the SonicMQ broker for Acceptor changes to take effect:

nanny.bat restartService tv_message_broker

9 Any Java client attempting to use the TV_SSL_ACCEPTOR must use the following java options to use the client truststore and keystore:

-Dsonic.mq.ssl.keyStoreClientAlias=tvclientcert

-Djavax.net.ssl.keyStore="%TVISION_HOME%\Sonic\MQ7.6\certs\client.jks"

-Djavax.net.ssl.keyStorePassword=changeit

-Djavax.net.ssl.trustStore="%TVISION_HOME%\Sonic\MQ7.6\certs\clienttrust.jks"

-Djavax.net.ssl.trustStorePassword=changeit

-DSSL_PROVIDER_CLASS=progress.message.net.ssl.jsse.jsseSSLImpl

On the Java client, you can add the following option to output SSL debugging information:

-Djavax.net.debug=all

On the SonicMQ broker, you can add the following advanced parameter on the broker to output SSL debugging information in the broker log:

Name: **BROKER_SSL_PARAMETERS.SSL_DEBUG**

Value: **true**

For additional topics related to securing SonicMQ, see the Progress SonicMQ Deployment Guide.

Standalone SonicMQ Broker

See the procedure in "SonicMQ Broker Bundled with TransactionVision" on page 317.

SonicMQ HTTP Server, WebSphere Queue Managers, TIBCO EMS Server, WebLogic JMS Server

See the documentation of your messaging middleware provider for information on the general configuration of the server and clients.

Typically, to configure the clients (the Analyzer or agents in this case) requires providing vendor-specific information on the Java command line indicating where trusted certificates can be found. To add any applicable Java arguments to the Analyzer startup you can modify the **service_jvm_flags** property in the **<TVISION_HOME>/config/services/Analyer.properties** file. To add properties to a Java Agent, modify the **tvProperties** field found in the **PROBEHOME/etc/TV.properties** file.

Configuring SSL Between the Analyzer and SonicMQ

It is unlikely that there would be a need for SSL encryption between the Analyzer and SonicMQ, since they typically exist on the same host. However, if there is a need for it, follow these steps to enable it.

To configure SSL between the Analyzer and SonicMQ:

1 Configure the Analyzer for SSL.

SSL configuration on the Analyzer is done in its startup script: **SetupEnv.bat** (Windows) or **SetupEnv.sh** (UNIX). By default, the **SetupEnv** script is configured for mutual SSL authentication using a **clienttrust.jks** for the truststore and **client.jks** for the keystore as described in "SonicMQ Broker Bundled with TransactionVision" on page 317. These settings are defined in the variable SONICMQ_SSL_CLIENT.

Note: With this default mutual SSL authentication, the Client Authentication needs to be enabled on the TV_SSL_ACCEPTOR acceptor as described in "SonicMQ Broker Bundled with TransactionVision" on page 317.

If for some reason Client Authentication is not desired, an optional SONICMQ_SSL_CLIENT that is commented out can be used. It only uses the **clienttrust.jks** for a truststore, and does not provide a private keystore for the Analyzer client.

2 Configure the SonicMQ Communication Link for SSL.

By default, the SonicMQ Communication Link is created without SSL support. To enable SSL in the SonicMQ Communication Link for communication between the Analyzer and SonicMQ:

a Copy the default SonicMQ Communication Link.

b Edit the copied SonicMQ Communication Link.

c Change the Analyzer Connections to use the ssl:// protocol and port 21112 on which the TV_SSL_ACCEPTOR listens.

d Assign the newly defined SonicMQ Communication Link to the Analyzer.

Enabling SSL for the Agent

The procedure is specific to the type of agent used in your deployment environment as described in the following subsections:

➤ "Configure Java Agents to Use SSL to TransactionVison's SonicMQ" on page 327

➤ "Configure the Tuxedo Agent to Use SSL" on page 328

➤ "Configure the .NET Agent to Use SSL" on page 329

➤ "Configure the DataPower Agent to Use SSL" on page 331

➤ "SSL and the WebSphere MQ Agents on Windows or UNIX" on page 334

➤ "SSL and the WebSphere MQ and CICS Agents on z/OS" on page 335

Configure Java Agents to Use SSL to TransactionVison's SonicMQ

Complete the following steps to configure Java Agents to use SSL:.

1 Modify the <**DiagnosticsAgent Path**>**/etc/TV.properties**:

a For the **defaultTransport.url** setting, change the protocol to **ssl**:// and the port to **21112** which is the listening port of TV_SSL_ACCEPTOR on the SonicMQ Broker:

ssl://*sonicmq_host*:21112

b Change the **defaultTransport.ssl_cert_dir** if you would like to use a different location in which the Java Agent looks for the certificates to trust.

This path is relative to the **TransactionVisionAgent** directory. It can also be set as an absolute path, if a directory outside of the Agent install directory is desired. Only use forward slashes "/" — even on Windows.

2 Copy the **tvservercert.cer** to the directory defined by **defaultTransport.ssl_cert_dir**.

Note: If the application that is being monitored by a Java Agent is using SSL-enabled Sonic to communicate with queues, you need to consider the following:

- The Java Agent does not support text-based PEM format certificates. When exporting the certificate from a keystore, do not use the -rfc option. For an example of exporting a certificate from a keystore, see "SonicMQ Broker Bundled with TransactionVision" on page 320.
- Because of a limitation in SonicMQ, it is not possible to configure two separate components running within the same process to use differing directory locations for looking up certificate files. Thus, both the Java Agent and the application need to place any trusted certificates in the same location.
- If this is the scenario, you need to 1) change the defaultTransport.ssl_cert_dir property located in the probe_home/etc/TV.properties file to point to where the application is configured to look for its certificates and 2) copy the tvservercert.cer certificate to that directory.

Configure the Tuxedo Agent to Use SSL

Complete the following steps to configure Tuxedo Agent to use SSL:

1. To retrieve configuration information from the Analyzer, edit the Tuxedo Agent property file.

   This can be found in the **<TVHOME>/tuxedo/config/TuxedoSensor.properties** directory on the host on which the Tuxedo Agent is installed.

2. In this property file, change the **transport** field to specify the HTTP over SSL URL for the Analyzer configuration service. By default, this URL is https://myhost:21103.

3 To configure the Tuxedo Agent to send events to the Analyzer using SSL, edit the communication link used by the Tuxedo Agent. For the Agent Connection, set the **Connection URL** field to specify the HTTP over SSL URL for the SonicMQ HTTP Direct for JMS Acceptor. By default, this URL is https://myhost:21114.

Tuxedo Agents configured to use SSL need to have a copy of the SSL certificate in PEM format. Using the default shipped **server.jks** keystore used by SonicMQ as an example, export its **tvservercert** public certificate in PEM Encoded format using a tool like Portecle (http://portecle.sourceforge.net/).

Additional properties controlling the Tuxedo Agent's behavior with respect to SSL are available and documented in the Tuxedo Agent properties file **TuxedoSensor.properties**.

Configure the .NET Agent to Use SSL

Steps 1-19 guide you through the process of installing the certificate. The final step involves editing an entry in the **probe_config.xml** file to enable SSL on the transport.

1 Copy the certificate from the TransactionVision SonicMQ Server to the machine where the .NET Agent is installed.

2 From the Windows taskbar, select **Start** > **Run**.

3 Run the Microsoft Management Console by typing **mmc**, and then clicking **OK**.

4 On the Microsoft Management Console menu, select **File** > **Add/Remove Snap-in** to display the Add/Remove Snap-in dialog.

5 Click **Add** on the Add/Remove Snap-in dialog.

6 Select **Certificates** from the Available Standalone Snap-in list and click **Add**.

7 In the Certificates Snap-in dialog box select **Computer account**, and click **Next**.

8 In the Select Computer dialog box select **Local Computer:** (the computer on which this console is running), and then click **Finish**.

The NT User account under which the monitored process is running needs to have its Certificate Store configured to be able to verify the certificate which the Sonic MQ server is using for the secure communication. For an ASP.NET application running **NETWORK SERVICE** (this is the default) credentials, the Certificate Store is the **Local Computer**. If this has been changed to be a different account, the certificate should be imported under that user, not Local Computer.

9 Click **Close** on the Add Standalone Snap-in.

10 Click **OK** on the Add/Remove Snap-in dialog.

11 On the Microsoft Management Console expand the listing for Certificates (Local Computer) in the left pane of the Console Root dialog.

12 Under Certificates (Local Computer), expand Trusted Root Certification Authorities.

13 Under Trusted Root Certification Authorities, right-click **Certificates** and select **All Tasks** > **Import** to start the Certificate Import Wizard.

14 Click **Next** to move past the Welcome dialog box of the Certificate Import Wizard.

15 Click **Browse** to navigate to directory where you have copied the exported certificate file. (**tvservercert.cer**)

16 Click **Next** to import the file.

17 Click **Next** to accept the default Certificate Store location of **Trusted Root Certification Authorities**.

18 Click **Finish** on Completing the Certificate Import Wizard.

19 Click **OK** on the Certificate Import Wizard confirmation dialog.

20 Modify **probe_config.xml** and change the broker URL for the <transport> element that configures the transport the .NET Agent uses. For example:

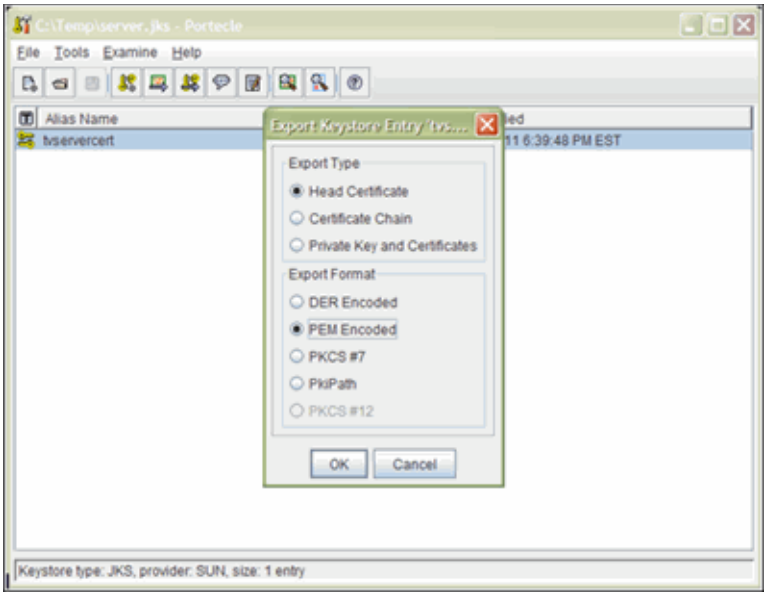
<transport type="**sonicmq**" connectionstring="broker=**ssl://brokerhos**t; port=**21112**; user=; password=; configurationqueue=**TVISION.CONFIGURATION.QUEUE**"/> Note the broker has been prefixed with "ssl://".

Configure the DataPower Agent to Use SSL

The DataPower Agent acts as a client to SonicMQ when delivering events to the TransactionVision Analyzer. The delivery of events in the DataPower Agent is handled by a Multi-Protocol Gateway defined in the HPTVMonitoring domain called **hptv-wsman-subscriber**. This Multi-Protocol Gateway needs to be configured with an SSL Client Crypto Profile to deliver to SonicMQ via HTTPS.

To configure the DataPower Agent Multi-Protocol Gateway for SSL:

1. Using the default shipped **server.jks** keystore used by SonicMQ as an example, export its **tvservercert** public certificate in PEM Encoded format using a tool like Portecle (http://portecle.sourceforge.net/).

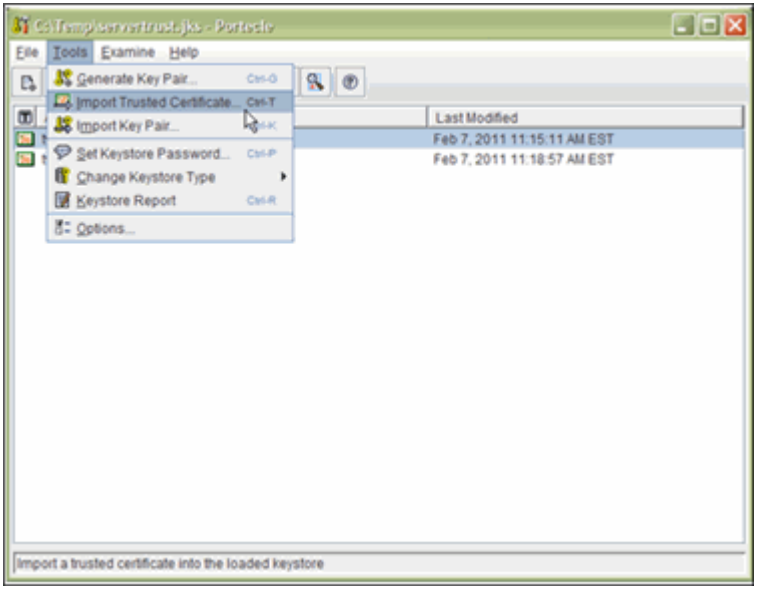


2. Create a new Crypto Profile in the HPTVMonitoring domain. Provide the exported **tvservercert.pem** certificate from step 1 for the Validation Credentials.

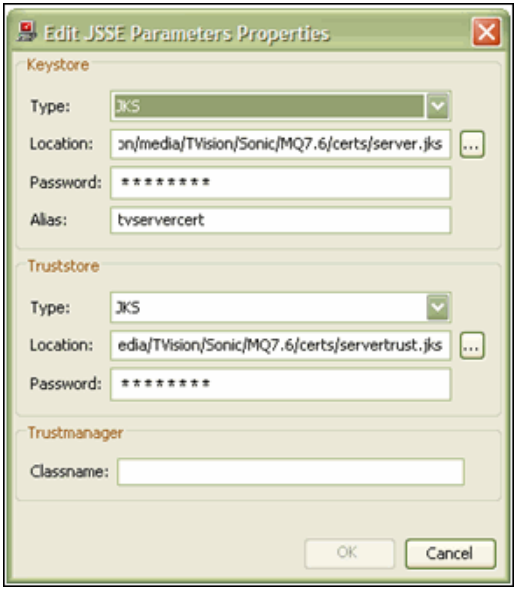
3. If Client Authentication is desired for the SSL connection, provide Identification Credentials with a created Crypto Key and Certificate.

4 Assign the newly defined Crypto Profile to the **hptv-wsman-subscriber** Multi-Protocol Gateway in the HPTVMonitoring domain.

5 Download and transfer the PEM certificate used in the Identification Credentials Certificate to the Analyzer, and import it into the **servertrust.jks** used by SonicMQ using a tool like Portecle (http://portecle.sourceforge.net/).



6 Ensure the keystore and truststore (if using Client Authentication) are properly configured on the SECURE_RUM_ACCEPTOR using the Sonic Management Console.



7 If using Client Authentication, ensure the Client Authentication Enabled check box is checked in the SECURE_RUM_ACCEPTOR SSL properties.

8 Restart the SonicMQ broker for Acceptor changes to take effect:

nanny.bat restartService tv_message_broker

SSL and the WebSphere MQ Agents on Windows or UNIX

If the WebSphere MQ Agent is monitoring a WebSphere MQ application making client connections, the WebSphere MQ Agent cannot open a connection independent of the type of connection the WebSphere MQ application makes. Thus if the application is making non-secured WebSphere MQ calls, the WebSphere MQ Agent is limited to that. If on the other hand the application is using an SSL connection, the WebSphere MQ Agent would use that secured connection.

If the WMQ Agent is monitoring a WMQ application making server connections, and there is a remote channel between the application's queue manager and a dedicated TransactionVision queue manager, SSL would be used. In this case, only a WebSphere MQ configuration is required — there are no TransactionVision specific tasks to be made to configure SSL.

If the analyzer is also using a server WebSphere MQ connection to retrieve events, it similarly would not need SSL enabled. If the analyzer were making a client WebSphere MQ connection to retrieve the messages through a SSL-enabled channel, then it would need to be configured accordingly (see "Enabling SSL for the Messaging Middleware Provider" on page 317).

SSL and the WebSphere MQ and CICS Agents on z/OS

Since the z/OS Agents only use server connections, SSL configuration is not needed. If there is a remote channel between the application's queue manager and a dedicated TransactionVision queue manager, SSL would be configured during a WebSphere MQ configuration — there would be no TransactionVision specific tasks to be made to configure SSL.

If the analyzer reading these messages is also using a server WMQ connection to retrieve events, it would not need SSL enabled. If the analyzer was making a client WMQ connection to retrieve the messages through a SSL-enabled channel, then it would need to be configured accordingly (see "Enabling SSL for the Messaging Middleware Provider" on page 317).

Enabling SSL for the SonicMQ Communication Link

By default, the SonicMQ Communication Link is created without SSL support.

To enable SSL in the SonicMQ Communication Link:

1. Copy the default SonicMQ Communication Link.
2. Edit the copied SonicMQ Communication Link.
3. Change the Agent Connection Protocol to ssl and Port to 21112 on which the TV_SSL_ACCEPTOR listens.

4. If there is a need for SSL between the Analyzer and SonicMQ (if they are on different hosts, for example), similarly, change the Protocol and Port for the Analyzer Connections.
5. Assign newly defined SonicMQ Communication Link to Analyzer.

Enabling SSL for the HTTP and RUM Communication Links

By default, the HTTP and RUM Communication Links are created without SSL support.

To enable SSL in the HTTP and RUM Communication Links:

1. Copy the default HTTP or RUM Communication Link.
2. Edit the copied Communication Link.
3. Change the Agent Connection URL to include https protocol and 21114 port on which the SECURE_RUM_ACCEPTOR listens:

   https://procserver_host:21114/tv_...

4. If there is a need for SSL between the Analyzer and SonicMQ (if they are on different hosts, for example), change the Protocol and Port for the Analyzer Connections.

   ssl://procserver_host:21112

5. Assign newly defined SonicMQ Communication Link to Analyzer.

Enabling SSL Communication for Events Reported to BPI

If BPI actions are defined in your classification rules, the Analyzer will send events to the BPI Engine when those corresponding transactions occur. These events are sent to BPI via the SonicMQ broker. By default, this communication is not secured.

To use a secure connection for sending BPI events, perform the following steps:

1. Go to the HP Business Process Insight page.
2. Change the JMS Connection Factory Name from its default, BPIQueueFactory, to BPISSLQueueFactory.

   Both these JNDI objects are defined by default within the TransactionVision SonicMQ server.

Enabling Authentication in TransactionVision SonicMQ

This section describes how to enable authentication in SonicMQ. In addition to securing messages through encryption using SSL, you can secure the SonicMQ Acceptors to require authentication from the agents and the Analyzer.

The following steps summarize how to define users and enable authentication on the broker. After performing these steps, define the communication link used by the agents and the Analyzer to use the user names specified here.

Similarly, for configuring RUM to use the Analyzer, the user name and password need to be configured by running **TVisionSetupInfo** or by manually changing the Settings Manager field in the Business Service Management user interface.

To define users and enable authentication on the broker:

1. Shut down the nanny using the appropriate command for your system:

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

   <TVISION_HOME>/bin/run_topaz stop (UNIX)

2. Start the SonicMQ domain manager using the appropriate script for your system:

   <TVISION_HOME>\Sonic\MQ7.6\bin\startdm.bat (Windows)

   <TVISION_HOME>/Sonic/MQ7.6/bin/startdm.sh (UNIX)

3. Start the SonicMQ broker using the appropriate script for your system:

   <TVISION_HOME>\Sonic\MQ7.6\bin\startbroker.bat (Windows)

   <TVISION_HOME>/Sonic/MQ7.6/bin/startbroker.sh (UNIX)

4. Start the Management Console using the appropriate script for your system:

   <TVISION_HOME>\Sonic\MQ7.6\bin\startmc.bat (Windows)

   <TVISION_HOME>/Sonic/MQ7.6/bin/startmc.sh (UNIX)

5. Turn on security in the broker, which is typically named the same as the hostname (without the domain name):

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

   b Click on the first '+' button named **Brokers** to expand the list of brokers configured in SonicMQ.

   c Select the Broker named the same as the host name and right-click to view the drop-down menu; select **Properties** from the menu.

   d On the Edit Broker Properties Window, enable Security by checking the check box next to **Security** in the Security section.

   e Set the broker password by clicking the **Set Broker Password** button in the Security section on the window and entering the password.

Note: By default, the Default Authentication Domain and Default Authentication Policy is selected. You can find more information on creating and configuring Authentication Domains and Policies in chapter 12 of the SonicMQ Configuration and Management Guide (<TVISION_HOME>\Sonic\Docs7.6\mq_config_manager.pdf).

---

6. Create new users and set the same password for the new users as described in step 2.
7. Exit the management console.
8. Stop the SonicMQ domain manager using the appropriate script for your system:

   <TVISION_HOME>\Sonic\MQ7.6\bin\stopdm.bat (Windows)

   <TVISION_HOME/Sonic/MQ7.6/bin/stopdm.sh (UNIX)
9. Stop the SonicMQ broker using the appropriate script for your system:

   <TVISION_HOME>\Sonic\MQ7.6\bin\stopbroker.bat (Windows)

   <TVISION_HOME/Sonic/MQ7.6/bin/stopbroker.sh (UNIX)
10. cd to <SONICMQ_HOME>, which is the same as **<TVISION_HOME>\Sonic\MQ7.6**.
11. Edit **<broker name>\db.ini** and set ENABLE_SECURITY=**true**.
12. Run the command: **bin\dbtool /f <broker name>\db.ini /d a**
13. Run the command: **bin\dbtool /f <broker name>\db.ini /c a**
14. Restart the nanny using the appropriate command for your system:

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

    <TVISION_HOME/bin/run_topaz start (UNIX)

Part V

Upgrading

24

Upgrading TransactionVision

This chapter provides instructions for upgrading TransactionVision from versions 9.00 or 9.01 to 9.02. The instructions are also the same for installing a component from a patch release.

This chapter includes:

Upgrading TransactionVision Overview

Upgrades from versions of TransactionVision prior to 9.00 are not supported. TransactionVision 9.02 only works with BSM 9.01. If you are running BSM 9.00, you must first install BSM 9.01.

The upgrade consists of two parts: upgrading the TransactionVision components (Processing Server and database), and upgrading the Transaction Management application in BSM, which is applied to BSM 9.01. The upgrade procedure includes upgrading both parts. This procedure requires that you access the TransactionVision Processing Server system(s) and the BSM 9.01 system.

See the TransactionVision 9.02 Readme for last-minute information regarding upgrading TransactionVision.

Notes and Limitations

Before you start the upgrade process, you need to be aware of the following:

- The following agents can be upgraded as described in the following sections:
  - "Upgrading the WebSphere MQ Agent on Windows" on page 348
  - "Upgrading the Java Agent" on page 350
  - "Upgrading the .NET Agent" on page 354
- The remaining agents do not contain configurations that can be migrated. Some agents need to be uninstalled and then installed to the new version. For details, see Part III, "Agent Installation and Configuration."

Upgrading TransactionVision to 9.02

This section contains instructions for upgrading TransactionVision to 9.02 from versions 9.00 or 9.01.

To upgrade TransactionVision to 9.02:

1 Check versions of prerequisite software.

TransactionVision version 9.02 might require different versions of the prerequisite software. Check the Processing Server and database management systems requirements in Chapter 3, "Reviewing System Requirements." Check the agents' versions compatibility with the TransactionVision 9.02 Processing Server in "Compatibility Matrixes" on page 34.

2 Backup your system:

Before starting the upgrade, back up your machine and databases.

- For your BSM system:
  - Create a copy of the files in **C:\<HP Business Service Management root directory>\HPBSM**.
  - Copy your BSM databases.

  For details, see "Configuration and Data File Backup" in the the *HP Business Service Management Deployment Guide* PDF.
- For your TransactionVision system:
  - Create a copy of the files in **C:\Program Files\HP\TransactionVision**.
  - Copy your TransactionVision database.

3 On the BSM system, stop BSM.

On Windows, select **Start** > **All Programs** > **HP Business Service Management** > **Administration** > **Disable HP Business Service Management**.

4 On the TransactionVision Processing Server, shut down your TransactionVision services:

- On Windows run: **<TVISION_HOME>\bin\SupervisorStop.bat**
- On Linux run: **<TVISION_HOME>/bin/run_topaz stop**

5 Download the TransactionVision Transaction Management UI zip package, **HPTVTMUI_9.02_win.zip**, from the SSO portal at http://support.openview.hp.com/selfsolve/patches, to the machine on which BSM 9.01 is installed, using your HP Passport login:

a Select **TransactionVision** in the Product field, **9.02** for the Product Version, desired Operating system, and click **Search**.

b Click the link for the Transaction Management UI package to download.

c Unzip the Transaction Management UI package on the machine where BSM 9.01 is installed, in a temporary directory.

6 On the BSM 9.01 machine, run the TransactionVision 9.02 install script from the temporary directory to which you copied the Transaction Management package.

tminstall_902.bat

Follow the prompts to install Transaction Management 9.02.

7 On the BSM 9.01 machine, start BSM.

On Windows, select **Start** > **All Programs** > **HP Business Service Management** > **Administration** > **Enable HP Business Service Management**.

8 Install the TransactionVision 9.02 Processing Server over the 9.0x Processing Server as follows:

a From a login account with Administrator privileges download the appropriate installation file for the TransactionVision Processing Server to a temporary directory on the machine on which the Processing Server exists:

- Access the SSO portal at http://support.openview.hp.com/selfsolve/patches, using your HP Passport login.
- Select **TransactionVision** in the Product field, **9.02** for the Product Version, desired Operating system, and click **Search**.
- Download the appropriate installation file:

  For Windows: **HPTVProcServer_9.02_win.exe**

  For Linux: **HPTVProcServer_9.02_linux.tgz**

b Run the appropriate installation file for the 9.02 installation. For details, see "Installing the Processing Server on Windows" on page 57 or "Installing the Processing Server on Linux" on page 59.

Note the following during the installation process:

- On Windows select **Reinstall** to upgrade the components installed by the previous setup.
- On Linux enter **y** when prompted to uninstall the current packages.

9 On the TransactionVision Processing Server, start the Processing Server.

- On Windows run: **<TVISION_HOME>\bin\SupervisorStart.bat**
- On Linux run: **<TVISION_HOME>/bin/run_topaz start**

10 In the BSM user interface, initialize the Processing Server:

a Select **Admin** > **Transaction Management**.

b (left pane) Click the **Configuration** tab.

c Expand Processing Servers and select the Processing Server to initialize.

d Click the **Initialize** button.

11 In the BSM user interface, stop the event collection for all analyzers:

a Select **Admin** > **Transaction Management**.

b (left pane) Click the **Configuration** tab.

c Expand the initialized Processing Server to display the Analyzer or Analyzers.

d For each Analyzer, select the **Status** tab > **General** tab, and click the **Force Stop Analyzer**  icon on the Analyzer Status page.

The analyzer status should state:

The Analyzer status is currently stopped.

12 Migrate project databases with the **MigrateDB** script after TransactionVision is upgraded and Initialize has been run.

- On Windows run: **<TVISION_HOME>/bin/MigrateDB.bat**
- On Linux run: **<TVISION_HOME>/bin/MigrateDB.sh**

The upgrade from TransactionVision 9.0x to 9.02 is complete.

13 After running **MigrateDB**, restart the Analyzers that you had stopped in step 11 by clicking **Start Analyzer** on the Analyzer Status page.

For more details, see the "Analyzers" chapter in *Using Transaction Management*.

You can now upgrade the agents to version 9.02 in the following sections:

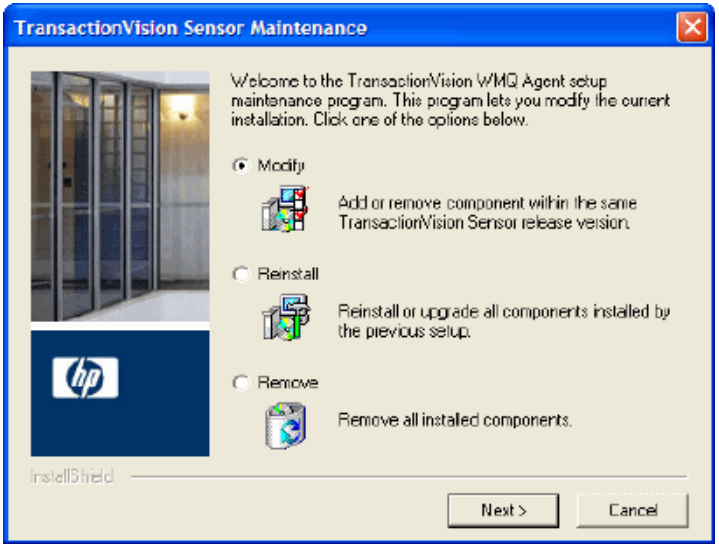
- "Upgrading the WebSphere MQ Agent on Windows" on page 348
- "Upgrading the Java Agent" on page 350
- "Upgrading the .NET Agent" on page 354

Upgrading the WebSphere MQ Agent on Windows

This section contains instructions for upgrading your TransactionVision WebSphere MQ Agent on Windows to version 9.02. For detailed information about the WebSphere MQ Agent, see Chapter 10, "Installing the WebSphere MQ Agent on Windows."

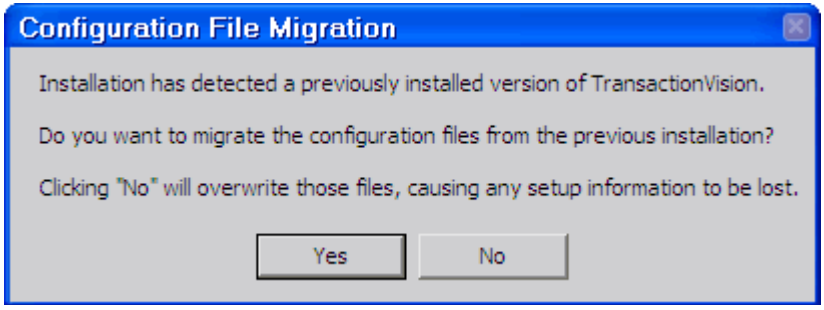
To upgrade the WebSphere MQ Agent on Windows, perform the following steps:

1 Double-click **HPTVWMQAgent_<version>_win.exe** to display the InstallShield Welcome screen and click **Next**. The agent setup maintenance menu displays:



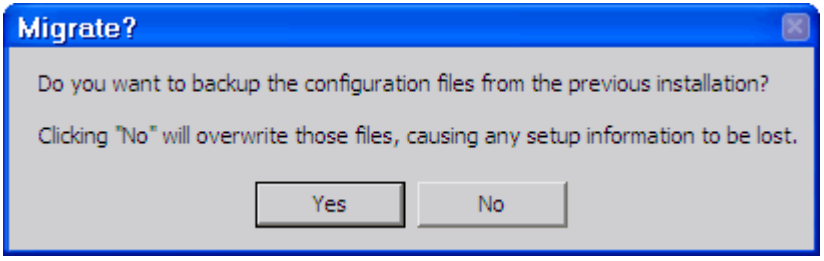
2 Select one of the following to upgrade the TransactionVision installation:

- If you want to install the WebSphere MQ Agent 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 the WebSphere MQ Agent using the settings from the previous installation. The Configuration File Migration dialog appears:



3 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 the WebSphere MQ Agent, 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 the WebSphere MQ Agent, overwriting existing configuration files, and displays the Setup Complete screen.

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

Upgrading the Java Agent

This section contains instructions for upgrading your Diagnostics/ TransactionVision Java Agent from an earlier version to 9.02.

Note: You must upgrade the Diagnostics Server and/or TransactionVision Processing Server before upgrading the agents that are connected to it because servers are typically incompatible with newer versions of the agents. For details, see "Upgrading TransactionVision to 9.02" on page 344.

To upgrade a Java Agent:

Note: The new agent installation does not begin monitoring your applications until you have updated the startup scripts to start the new agent and restarted the applications as described in these instructions.

---

1 Install the Diagnostics/TransactionVision Agent for Java into a different directory than the current agent's installation directory.

During the installation, be sure to:

- Configure the Java Agent to work with a TransactionVision Processing Server. The Java Agent can also be configured to work with a Diagnostics Server if desired.
- For the agent name, use the same probe name as used by the previous agent.

- For the agent group name, use the same group name as used by the previous agent.
- For the mediator server name and port (available only when working with a Diagnostics Server), use the same information as used by the previous agent.
- For the event transport provider (available only when working with a TransactionVision Processing Server), use the same information as used by the previous agent.
- Run the **JRE instrumenter** regardless of the JRE version you use and even if you did not run the JRE Instrumenter before.

This ensures that the persisted data for your application will match up with the metrics captured by the new agent.

The installer creates a **<probe_install_dir>\etc** directory in the new installation directory.

In 7.50 and later releases, the default directory of **<probe_install_dir>** is: **C:\MercuryDiagnostics\JavaAgent\DiagnosticsAgent** on Windows and **/opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent** on UNIX.

Note: If you want to perform silent installation on multiple machines, be sure to re-record the Java Agent silent install response files for each new release of Java Agent. For details, see "Optional Task: Silent Installation of the Java Agent" on page 151.

---

2 Compare the new agent's **\etc** directory and the previous agent's **\etc** directory so that you can determine the differences between the two.

HP recommends that you execute the **Property Scanner** utility provided with the Java Agent which will indicate the differences (properties and points) between two different Java Agent installations. To execute the Property Scanner utility, change the current directory to **<probe_install_dir>/contrib/JASMUtilities/Snapins** and execute the **runPropertyScanner.cmd –console** (**.sh** for Unix) command as follows:

```
runPropertyScanner –console –diffOnly yes –Source1 ..\..\..\etc –Source2
OtherEtc
```

Sample Input:

```
C:\MercuryDiagnostics\JavaAgent8\DiagnosticsAgent\contrib\JASMUtilities\Snapins>runPropertyScanner -console -diffOnly yes -Source1 C:\MercuryDiagnostics\JavaAgent\DiagnosticsAgent\etc -Source2 C:\MercuryDiagnostics\JavaAgent8\DiagnosticsAgent\etc
```

Sample Output:

```
****** Property dispatcher.properties:stack.trace.method.calls.max
PropertyFile=dispatcher.properties
Property=stack.trace.method.calls.max
Source1=
Source2=1000
```

Apply any differences that were caused by the customizations that you made to the previous agent's **\etc** directory to the new agent's **\etc** directory so that they will not be lost. You should look for the following changes:

Property File	Configuration Properties To Be Copied to the New Diagnostics Agent
auto_detect.points	Copy custom points that you have created and points that you have modified from the **auto_detect.points** file in the old **etc** directory to the new **etc** directory. Be sure to check the points for RMI, LWMD, args_by_class when looking for points you may have modified.
capture.properties	Depth and latency trimming.
inst.properties	define.pre.process
dispatcher.properties	minimum.sql.latency sql.parsing.mode
dynamic.properties	cpu.timestamp.collection.method
metrics.config	Verify that any metric that you uncommented in the previous version is also uncommented in the new version so that you can continue to use the metrics that you are used to.

Property File	Configuration Properties To Be Copied to the New Diagnostics Agent
security.properties	If the system is set up for SSL mode, set all properties and copy the certificates from the old property file to the property file.
TV.properties	If you have tweaked any paths or properties, or use Point Expressions to extract payload or arguments.

3 Update the applications startup script to point to the upgraded agent installation. This includes the **-javaagent** and/or **-Xbootclasspath** provided by the new JRE Instrumenter.

4 At an approved time, shut down the applications that were being monitored by the old agent.

5 Restart the applications to allow the new version of the agent to begin monitoring your applications.

6 If you configured the Java Agent to work with a TransactionVision Processing Server, you can verify that the upgraded Java Agent is running and sending events, by checking the Transaction Management reports such as Transaction Tracking report or Event Analysis report, on the Business Service Management user interface.

7 If you configured the Java Agent to work with a Diagnostics Server, you can verify that the upgraded Java Agent is running and properly registered, by checking the version in System Health. Browse to the System Health http://<commanding_server>:<port>/registrar/health, and double-click the agent icon. The version listed under Configuration should be the latest version, if the upgrade was successful and the new version of the Java Agent was started.

8 When all your applications have been migrated over to be version 9.02 and everything is working properly, you can delete the old directory. Do not try to uninstall the old version because this will actually uninstall the new version.

Upgrading the .NET Agent

This section contains instructions for upgrading your Diagnostics/ TransactionVision .NET Agent from an earlier version to 9.02.

Note: You must upgrade the Diagnostics Server and/or TransactionVision Processing Server before upgrading the agents that are connected to it because servers are typically incompatible with newer versions of the agents. For details, see "Upgrading TransactionVision to 9.02" on page 344.

To upgrade a .NET Agent:

1. Install the new Diagnostics/TransactionVision Agent for .NET.

   The upgrade will take effect when the probed applications are restarted.

2. To force the upgrade to take effect:

   a. Shut down all applications that are being monitored by the current .NET Probe.

   b. Restart IIS.

   c. Restart the applications that were being monitored by the old probe.

3. You can manually verify the version of the **HP.Profiler.dll** file found in the .NET Agent\bin folder.

Index

E

F

H

I

J

S

T

U

W

Z