HP Business Service Management

for the Windows and Linux operating systems

Software Version: 9.25

TransactionVision Deployment

Document Release Date: January 2015
Software Release Date: January 2015

Legal Notices

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

PART V: UPGRADING

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.

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.

A PDF version of all manuals can be found on the HP Software Product Manuals web site at
http://h20230.www2.hp.com/selfsolve/manuals.

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.

➤ The *BSM User Guide* describes how to use the Transaction Management application to view and customize reports and topologies of business transactions.

➤ The *BSM Application Administration Guide* describes how to use the Transaction Management application to set up and configure TransactionVision to track transactions and also how to define business transaction CIs.

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

➤ 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.

➤ The *TransactionVision Release Notes* contain important information regarding limitations and system requirements for a specific release of TransactionVision.

Note: Updates to these guides sometimes occur independently of the software. See "Documentation Updates" on page 17 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:

➤ **Release Notes**. Provides a list of version limitations and last-minute updates. From the HP Business Service Management DVD root directory, double-click **BSM_<version_number>_Release_Notes.pdf**. You can also access the lates release notes 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**.

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 **TransactionVision**, the desired product version and operating system, and click **Search**.

Part I

Introduction to TransactionVision

1

Introduction to TransactionVision

This chapter includes:

About TransactionVision

HP TransactionVision is the transaction tracking solution that 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.

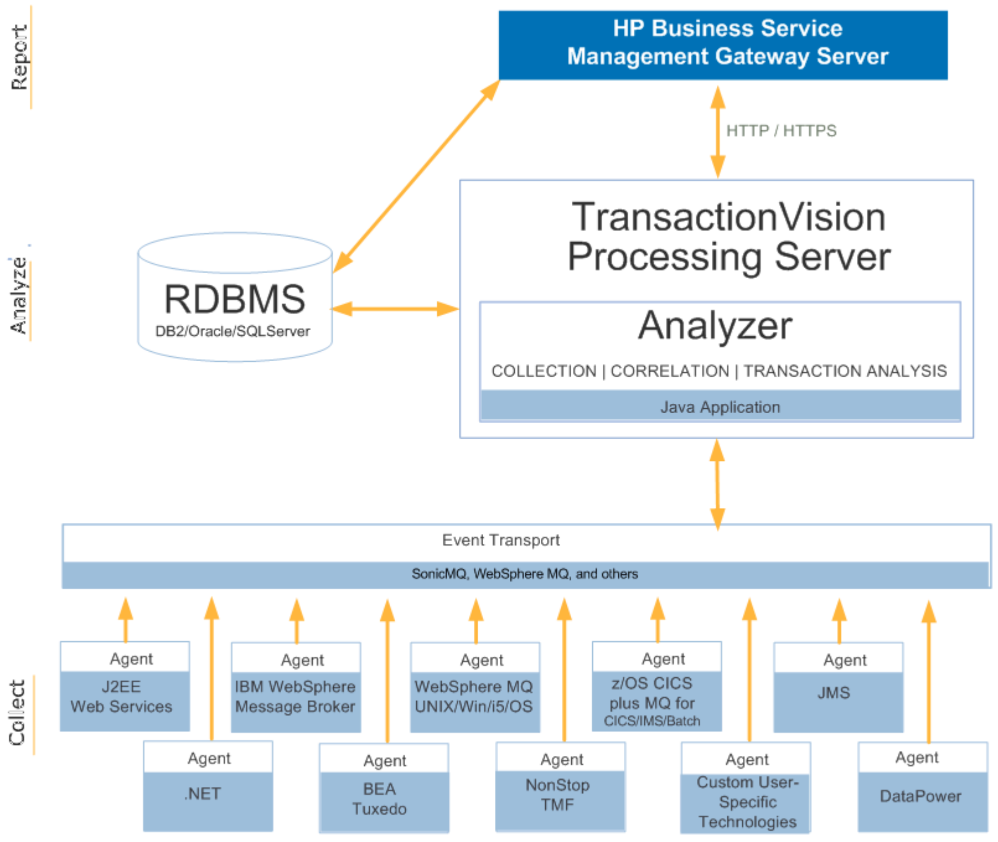
Key capabilities of TransactionVision are:

- ➤ 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 (such as customer identity, financial value).
- ➤ 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.


Report
Analyze
Collect
HP Business Service Management Gateway Server
HTTP / HTTPS
TransactionVision Processing Server
Analyzer
COLLECTION | CORRELATION | TRANSACTION ANALYSIS
Java Application
RDBMS
DB2/Oracle/SQLServer
Event Transport
SonicMQ, WebSphere MQ, and others
Agent
J2EE Web Services
Agent
.NET
Agent
IBM WebSphere Message Broker
Agent
BEA Tuxedo
Agent
WebSphere MQ UNIX/Win/i5/OS
Agent
NonStop TMF
Agent
z/OS CICS plus MQ for CICS/IMS/Batch
Agent
Custom User-Specific Technologies
Agent
JMS
Agent
DataPower

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 "Analyzers" in the *BSM Application Administration Guide* 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 55 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:

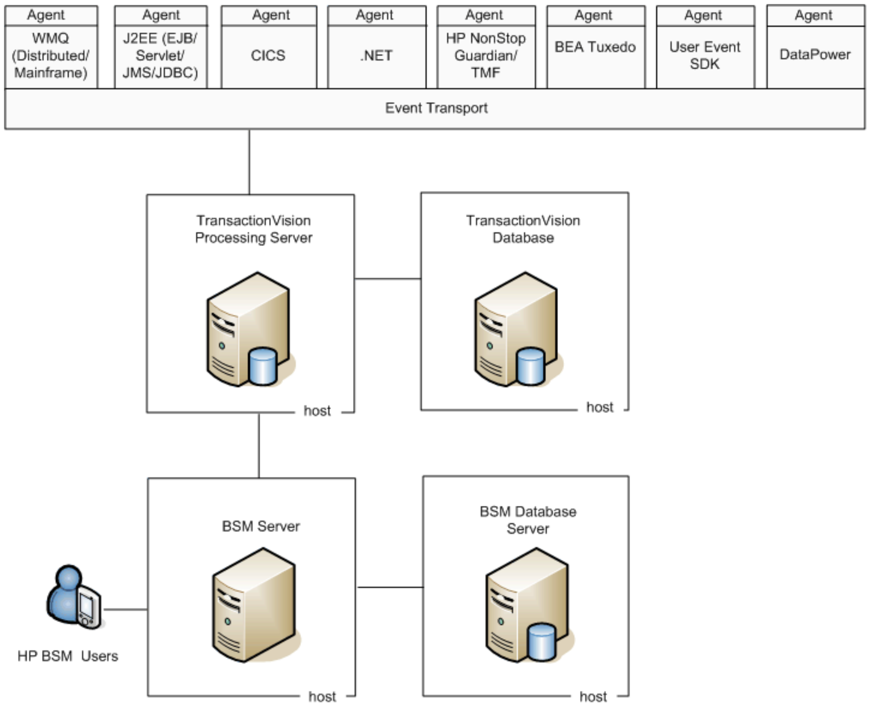
- One-Machine Deployment
- Distributed Deployment

For information about setting up these deployment environments, see the *BSM Planning 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.

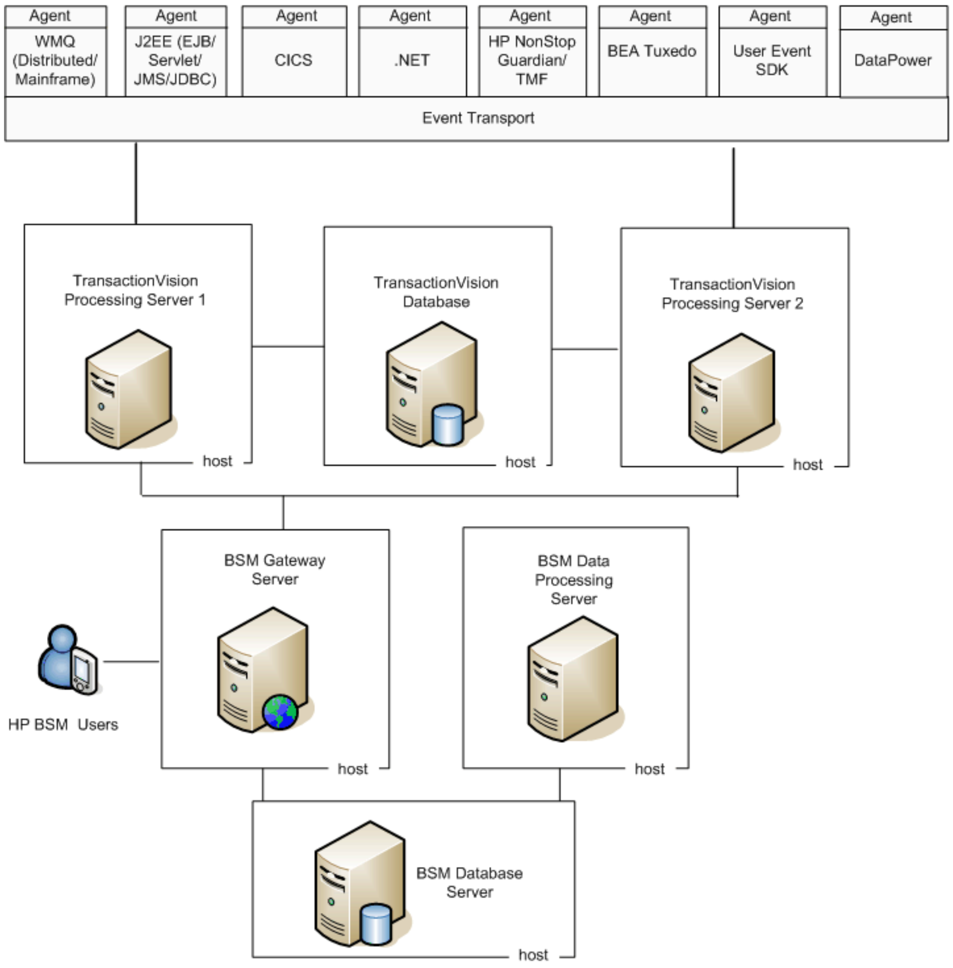


Note: Each TransactionVision Processing Server contains an embedded database that can be used in place of the TransactionVision database for POC or other testing purposes.

Distributed Deployment

A distributed 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. Each of them may share the same database (for simplicity and resource consumption) or use its own database (for performance and scalability).



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 distributed Business Service Management deployment can have various configurations. These configurations differ in terms of the memory and CPU requirements. See the *BSM Planning Guide* PDF.

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

A Closer Look at the TransactionVision Processing Server

Multiple Processing Servers in the Deployment Environment

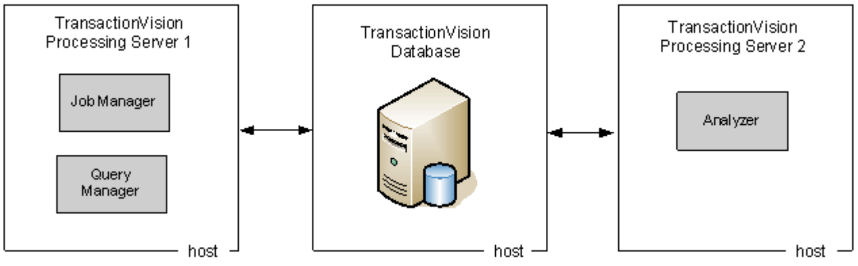
When the deployment environment includes multiple Processing Servers, you have several options in how the Analyzer instances, Job Manager, and Query Manger components are placed.

The guidelines for the deployment environment are:

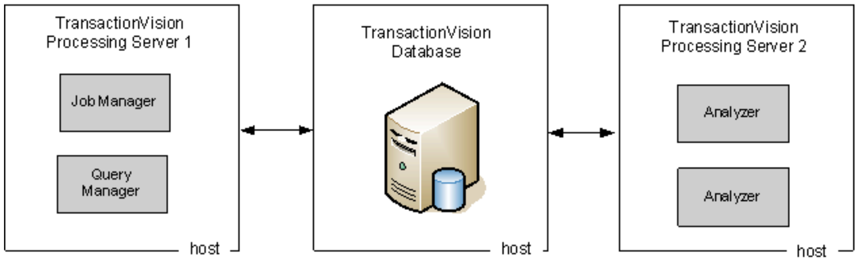
- ➤ The TransactionVision deployment environment must have at least one Processing Server.
- ➤ The TransactionVision deployment environment must have at least one Analyzer, one Job Manager, and one Query Engine. Each of these runs in the context of a Processing Server.
- ➤ The TransactionVision deployment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

Some possible configurations are shown below.

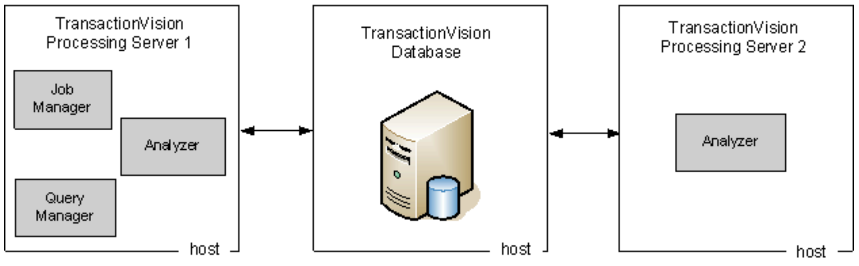
➤ One Processing Server dedicated to the Job and Query Managers and one Processing Server dedicated to the Analyzer:


TransactionVision Processing Server 1
Job Manager
Query Manager
host
TransactionVision Database
host
TransactionVision Processing Server 2
Analyzer
host

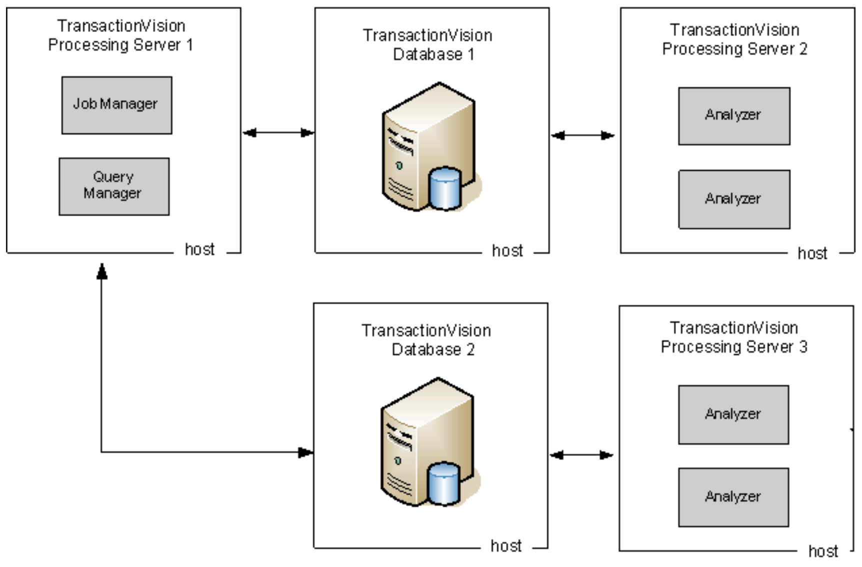
➤ One Processing Server dedicated to the Job and Query Managers, one Processing Server dedicated to two instances of the Analyzer


TransactionVision Processing Server 1
Job Manager
Query Manager
host
TransactionVision Database
host
TransactionVision Processing Server 2
Analyzer
Analyzer
host

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


TransactionVision Processing Server 1
Job Manager
Analyzer
Query Manager
host
TransactionVision Database
host
TransactionVision Processing Server 2
Analyzer
host

➤ One Processing Server dedicated to the Job and Query Managers, the remaining two Processing Servers each running two Analyzers and using its own database:


TransactionVision Processing Server 1
Job Manager
Query Manager
host
TransactionVision Database 1
host
TransactionVision Processing Server 2
Analyzer
Analyzer
host
TransactionVision Database 2
host
TransactionVision Processing Server 3
Analyzer
Analyzer
host

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
9.25	9.25

Note: TransactionVision Processing Servers may work with older BSM 9.2x versions. Contact HP TransactionVision Support for details.

TransactionVision Processing Server and Agents

The following table lists TransactionVision agent versions compatible with the 9.25 Processing Server, as well as the versions of the Processing Server that are compatible with the 9.25 version of that agent.

TransactionVision Agent	Versions of Agent Compatible with 9.25 Processing Server	Versions of Processing Server Compatible with 9.25 Agent[1]
HP Diagnostics/ TransactionVision Java Agent	8.0x, 9.10, 9.2x	Not Applicable
HP Diagnostics/ TransactionVision .NET Agent	8.0x, 9.10, 9.2x	Not Applicable
WebSphere MQ Agent	8.0x, 9.10, 9.2x	9.2x
CICS, WMQ Batch, WMQ CICS, WMQ IMS, and IMS Bridge Agents on z/OS	8.0x, 9.10, 9.2x	9.2x

[1] If you require use of the 9.25 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 downloading from the HP Software Product Manuals site at http://h20230.www2.hp.com/selfsolve/manuals.

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 and TransactionVision release notes may contain additional system requirements. For information about how to access the release notes, see "TransactionVision and Transaction Management Documentation" on page 14.

This chapter includes:

- Java Support on page 42
- Localization/L10N and Internationalization/I18N Support on page 42

Supported Processing Server Platforms

Operating Environment	WebSphere MQ	SonicMQ
Microsoft Windows Server 2008 Enterprise Edition R2 SP1 (64-bit) Microsoft Windows Server 2008 Enterprise Edition SP2 (64-bit) Microsoft Windows Server 2012 (Standard and Datacenter Editions) Microsoft Windows Server 2012 R2 (Standard and Datacenter Editions)	7.0, 7.1, 7.5	8.5, 8.6
RedHat Enterprise Linux 5.x x86/64-bit RedHat Enterprise Linux 6.x x86/64-bit Oracle Enterprise Linux 6.x 64-bit	7.0, 7.1, 7.5	8.5, 8.6

Running the Processing Server on virtual platforms is supported. For details, see "Supported Virtual Platforms" on page 41.

Supported Database Management Systems

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

DBMS Servers
DB2 9.5
DB2 9.7
Oracle 10.2.0.5 Enterprise Edition
Oracle 10.2.0.5 RAC Enterprise Edition
Oracle 11.2 (11g R2) Enterprise Edition
Oracle 12c Enterprise Edition
Oracle 12c RAC Enterprise Edition
Microsoft SQL Server 2008 Enterprise R2 SP1, SP2 (64-bit)
Microsoft SQL Server 2008 Enterprise SP2, SP3 (64-bit)
Microsoft SQL Server 2012 Enterprise Edition (64-bit)
Microsoft SQL Server 2012 Enterprise Edition - with failover clustering (64-bit)

These database server configurations can be accessed remotely through the DataDirect Connect JDBC drivers. 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 for Agent Event Transport

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	✓				

[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

Note: TransactionVision 9.2x does not include an updated version of the WebSphere MQ Agent for i5/OS. Customers who wish to use the WebSphere MQ Agent under i5/OS can use the latest version of the Agent released with TransactionVision 8.0, which is compatible with the TransactionVision 9.2x Processing Server.

Platform	Operating Environment	WebSphere MQ	Supports WMQ API Exit Agent
Intel x86	Windows Server 2008(64-bit) Windows Server 2008 R2 (64-bit) Windows Server 2012 (Standard and Datacenter Editions)	7.0, 7.1, 7.5[1, 4]	Yes
	RedHat Enterprise Linux 4.x, 5.x, 6.x (32-bit and 64-bit)[2] SUSE Linux Enterprise Server 11 (32-bit and 64-bit)[2]	7.0, 7.1, 7.5[1, 4, 5](32-bit) 7.0, 7.1, 7.5[1, 4, 5](64-bit)	Yes
HP PA-RISC/ Intel Itanium	HP-UX 11i v3[2]	7.0, 7.1, 7.5[1, 4, 5](32-bit) 7.0, 7.1, 7.5[1, 4, 5](64-bit)	
Oracle SPARC	Solaris 9, 10, 11[2]	7.0 , 7.1, 7.5[1, 4, 5](32-bit) 7.0, 7.1, 7.5[1, 4, 5](64-bit)	Yes
IBM Power	AIX 6.1[2] ,7.1	7.0, 7.1, 7.5[1, 4, 5](32-bit) 6.0, 7.0, 7.1, 7.5[1, 4](64-bit)	Yes

Platform	Operating Environment	WebSphere MQ	Supports WMQ API Exit Agent
IBM System zSeries	z/OS 1.13 CICS TS 3.2, 4.1, 4.2[3] IMS 10, 11, 12[3] Batch [3]	7.0, 7.1, 7.5[1, 4]	N/A
	SUSE Linux Enterprise Server 11 (s390x)[2]	7.0, 7.1, 7.5[1]	Yes

[1] TransactionVision does support WebSphere MQ applications running against WebSphere MQ 7.x. However, events will not be generated from new WebSphere MQ 7.x APIs added in that release, with the exception of the Pub/Sub API.

[2] **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*.

[3] 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.

[4] For WebSphere MQ 7.0, Fix Pack 7.0.0.1 or greater is required to use the WebSphere MQ API Exit Agent and the Analyzer.

[5] The TransactionVision 9.2x WMQ Agent with WebSphere MQ 7.1 or later has these limitations due to the new feature where multiple versions of WebSphere MQ may not be run on a single system: (1) The TransactionVision 9.2x WebSphere MQ Agent may only be configured to run with one version of WebSphere MQ on a system. (2) WebSphere MQ 7.1 no longer requires that the software be installed under the traditional locations on Linux/Unix systems, but TransactionVision 9.2x depends on this. If WebSphere MQ 7.1 is installed in a non-default location, a symbolic link must be generated from the **/usr/lpp/mqm** directory on AIX, or from the **/opt/mqm** directory on all other Linux/Unix distributions to the actual installation directory.

Supported Java Agent Platforms

Platforms	Operating Environments
x86/x64	Windows Server 2008 SP1 Windows Server 2008 SP2 (64-bit) Windows Server 2012 (Standard and Datacenter Editions) Windows Server 2012 R2 (Standard and Datacenter Editions)
x86/x64	RedHat Enterprise Linux 5.x, 6.x x86/64-bit Oracle Enterprise Linux 6.x 64-bit SUSE Enterprise Linux 10, 11 (32-bit and 64-bit)
Oracle SPARC	Solaris 9, 10, 11
IBM POWER	AIX 6.1, 7.1
HP PA-RISC/ Intel Itanium	HP-UX 11i v2, v3
IBM System zSeries	z/OS 1.13 SUSE Linux Enterprise Server 11 (s390x)

Running Java Agent on virtual platforms is supported. For details, see "Supported Virtual Platforms" on page 41.

Supported Java Agent Technologies

Application Servers
WebSphere Application Server 7[4], 8.0[4], 8.5[4],
WebLogic Application Server 10, 11g, 12c
JBoss Application Server 5.1, 6.1[5, 6], 7.1, EAP 6.1
Jetty 6.1[5]
Tomcat 5.5, 6.0, 7.0
TIBCO BusinessWorks 5.6, 5.9[7]

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

[5]Java Agent support for JBoss Application Server and Jetty may require a manual configuration change in the Java Agent configuration. See "Additional Configuration of the Applications to Be Monitored" on page 137.

[6] Java Agent supports monitoring EJB 3 but not EJB 2 on the JBoss Application Server.

[7] Java Agent collection from TIBCO BusinessWorks only supports collection of TIBCO EMS events and Servlet events from the underlying Tomcat server.

JMS Providers
WebSphere MQ 6.0[1], 7.0, 7.1, 7.5
TIBCO EMS 4.2.0, 4.4.2, 5.1, 6.0
SonicMQ 7.6.2, 8.5
WebLogic JMS 9.2.3, 10, 11, 12

[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, 9.7 SQL Server 2005, 2008, 2012

Supported CICS Agent Platforms

Platform	**Operating Environment**	**WebSphere MQ**
IBM System zSeries	z/OS 1.13 CICS TS 3.2, 4.1, 4.2	6.0, 7.0, 7.1

Supported .NET Agent Platforms

Platform	**Operating Environment**	**.NET**
Intel x86	Windows Server 2008 SP2 (64-bit) Windows Server 2008 R2 (64-bit) Windows Server 2012 (Standard and Datacenter Editions) Microsoft .NET Framework 4.5.1 and 4.5.	2.0, 3.0, 3.5, 4.0

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 and VMware ESX 5.x virtualization platforms are 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 *BSM System Requirements and Support Matrixes* PDF.

LDAP Support

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

Java Support

The TransactionVision Processing Server includes the Java 1.7 Runtime Environment in its install.

The applets that display the Component Topology Analysis, Aggregated Topology and Instance Topology views require Version 6 update 45, or Version 7 update 51 and higher in the client browser.

Java Agent monitoring of WebSphere and WebLogic Application Servers is only supported with the JVMs that are distributed with these application servers.

Flash Player Support

Some TransactionVision reports and topologies require Adobe Flash Player. ee the *BSM System Requirements and Support Matrixes* PDF for version information.

Localization/L10N and Internationalization/I18N Support

TransactionVision is both localized and internationalized. The Transaction Management user interface supports the following languages: English, Japanese, Korean, Simplified Chinese, French, German, Spanish, and Russian.

Transaction Management is I18N compliant and supports display of event fields, user data, and reports in non-English locales, as well as non-English query names.

The following TransactionVision content is not localized:

- Product names, application names
- User-definable objects (such as jobs, filters, and communication links)
- Configuration files
- Most date formats

For notes and limitations about BSM's multilingual support, see the BSM Release Notes.

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 *BSM Application Administration Guide*.

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 "Administration Utilities" chapter in *BSM Application Administration Guide*.

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 33.

3 Install the Processing Server.

For installing on a host running a Windows operating system, see "Installing the Processing Server" on page 48.

For installing on a host running a Linux operating system, see "Installing the Processing Server on Linux" on page 50.

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 for Agent Event Transport" on page 35.

---

5 Set up the database.

See Chapter 6, "Configuring Databases."

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

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 18, "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, from the Downloads page in BSM, 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:

   ➤ Copy the Process Server installation file from the product install disks.

   ➤ Download the Processing Server installation file from the Downloads page in BSM by selecting **Admin** > **Platform** > **Setup and Maintenance** > **Downloads**. Select the **Category** for **TransactionVision**.

   ➤ For a patch 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.tgz**) 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 product install disks.
- ➤ For a patch 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**.

7 Specify the user name under which the Processing Server runs each time it is started. The following prompt is displayed:

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

➤ To run the Processing Server as the **root** user, press **Enter**.

➤ To run the Processing Server as a user other than **root**, enter that user name and then press **Enter**.

This sets appropriate permissions on the installed files for that user, and is necessary in deployments where there are restrictions on running software as **root**.

8. 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.

9. Type **q** and press **Enter** to quit the installation process.
10. 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 50.

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 *BSM Application Administration Guide*.

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 34.

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 it is expected that TransactionVision will be used in a relatively simple environment where minimal database connections are required; no special configuration is required for the Oracle DBMS.

However, environments where a large number of users will be simultaneously accessing the 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 show up in the Processing Server logs if this limit is exceeded. To increase the number of Open Cursors, execute the following command:

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

This change 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

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

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 "DBMS Performance Testing" in the *TransactionVision Planning Guide*.

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 "Administration Utilities" in *BSM Application Administration Guide*.

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 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 on page 64
- ➤ TransactionVision Database Properties on page 64

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 "Administration Utilities" in *BSM Application Administration Guide*.

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 "Administration Utilities" in *BSM Application Administration Guide*.

Custom XDM Tables

Custom user-defined event or transaction tables can also be managed by the TransactionVision partitioning tools. 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) will 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 Concepts for Analyzers" in *BSM Application Administration Guide*.

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, and calculated based on OS instances. An OS instance is one of the following TransactionVision-specific object types:

- Physical or virtual hosts where applications are run that are being monitored by the TV WMQ Agent, Java Agent, .NET Agent, or custom developed agents
- CICS Regions - for events collected from the z/OS CICS or WMQ-CICS Agents
- IMS Regions - for events from the z/OS WMQ-IMS or WMQ IMS Bridge Agents
- WMQ Queue Managers - only for events collected from the z/OS WMQ Batch Agent

Capacity is based on the total number of OS instances used over the last 24-hour period. Except in the case of z/OS, if events are collected from multiple agents on a single host OS, they will count as only one OS instance. 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 (OS instance count).

➤ The purchased TransactionVision license capacity (maximum number of OS 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 the *BSM Platform Administration Guide*.

Note: Initially you are assigned a 60 day time-based evaluation license. This license covers the TransactionVision application and an unlimited number of OS 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 228.

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 *BSM Application Administration Guide*.

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:

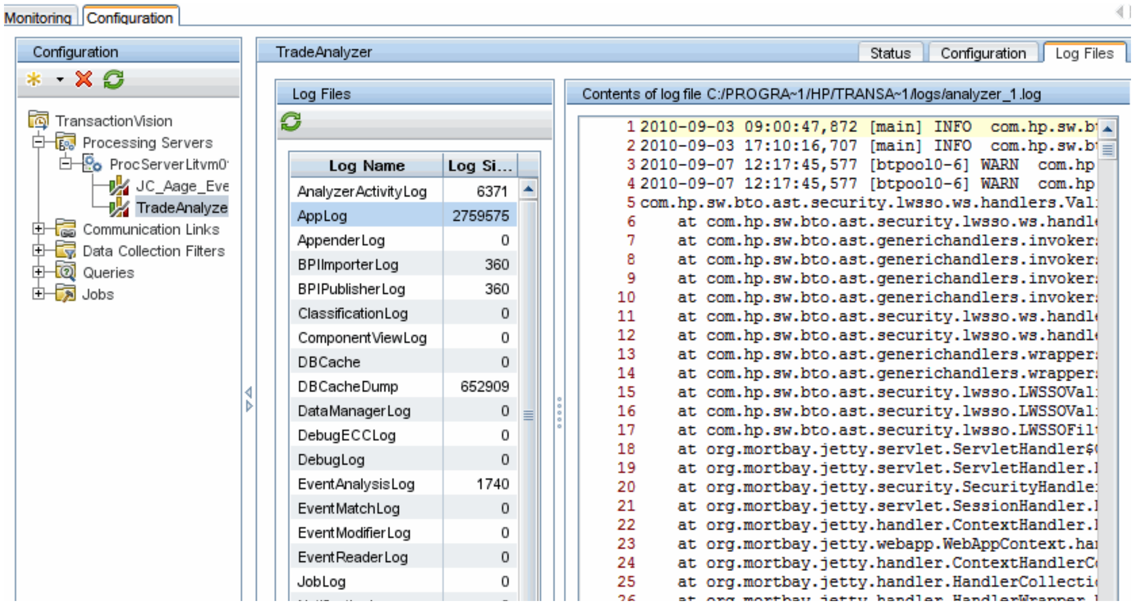
1 Log into BSM as a user with Transaction Management Administration privileges.

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:



Log Name	Log Si...
AnalyzerActivityLog	6371
AppLog	2759575
AppenderLog	0
BPIImporterLog	360
BPIPublisherLog	360
ClassificationLog	0
ComponentViewLog	0
DBCache	0
DBCacheDump	652909
DataManagerLog	0
DebugECCLog	0
DebugLog	0
EventAnalysisLog	1740
EventMatchLog	0
EventModifierLog	0
EventReaderLog	0
JobLog	0

Contents of log file C:/PROGRA~1/HP/TRANSA~1/logs/analyzer_1.log

```
2010-09-03 09:00:47,872 [main] INFO  com.hp.sw.b
2010-09-03 17:10:16,707 [main] INFO  com.hp.sw.b
2010-09-07 12:17:45,577 [btpool0-6] WARN  com.hp
2010-09-07 12:17:45,577 [btpool0-6] WARN  com.hp
com.hp.sw.bto.ast.security.lwsso.ws.handlers.Val
    at com.hp.sw.bto.ast.security.lwsso.ws.handl
    at com.hp.sw.bto.ast.generichandlers.invoker
    at com.hp.sw.bto.ast.generichandlers.invoker
    at com.hp.sw.bto.ast.generichandlers.invoker
    at com.hp.sw.bto.ast.generichandlers.invoker
    at com.hp.sw.bto.ast.security.lwsso.ws.handl
    at com.hp.sw.bto.ast.security.lwsso.ws.handl
    at com.hp.sw.bto.ast.generichandlers.wrapper
    at com.hp.sw.bto.ast.generichandlers.wrapper
    at com.hp.sw.bto.ast.security.lwsso.LWSSOVal
    at com.hp.sw.bto.ast.security.lwsso.LWSSOVal
    at com.hp.sw.bto.ast.security.lwsso.LWSSOFil
    at org.mortbay.jetty.servlet.ServletHandler$
    at org.mortbay.jetty.servlet.ServletHandler.
    at org.mortbay.jetty.security.SecurityHandle
    at org.mortbay.jetty.servlet.SessionHandler.
    at org.mortbay.jetty.handler.ContextHandler.
    at org.mortbay.jetty.webapp.WebAppContext.ha
    at org.mortbay.jetty.handler.ContextHandlerC
    at org.mortbay.jetty.handler.HandlerCollecti
```

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 77.

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 79

➤ "UNIX Event Appender" on page 79

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

Users may use an SMTP appender to send an email to an SMTP server when an error log event at the specified threshold reaches the appender. To define an SMTP appender named EMAIL, modify the following according to your environment and add it to the **Analyzer.Logging** properties in your Analyzer configuration:

```
<appender name="EMAIL" class="com.bristol.tvision.util.log.smtp.SMTPAuthenticateAppender">
           <param name="SMTPHost" value="smtp.mycompany.net" />
           <param name="To" value="analyzer_admin@mycompany.net" />
           <param name="From" value="administrator@mycompany.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="error" />
          <param name="Subject" value="Analyzer error" />
           <layout class="org.apache.log4j.PatternLayout">
                <param name="ConversionPattern" value="%d [%t] %-5p %c %x - %m%n" />
           </layout>
     </appender>
```

You also need to add a reference to this EMAIL appender in one or more logging categories. For example, to send an email when an Analyzer encounters an error, add the following line to the AppLog category in the **Analyzer.Logging** properties:

```
<appender-ref ref="EMAIL"/>
```

The final AppLog category may look like the following:

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

In the EMAIL appender definition, the Threshold parameter specifies the logging level that is allowed to append into the SMTP appender.

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

```
<param name="EvaluatorClass" class="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

Users may use an SNMP Trap appender to send SNMP traps to an SNMP management host when a specified error level occurs.

To define an SNMP Trap appender named TRAP_LOG using JoeSNMPTrapSender, modify the following according to your environment and add it to the **Analyzer.Logging** properties in your analyzer configuration:

```
 <appender name="TRAP_LOG" class="com.bristol.tvision.util.log.snmp.SNMPTrapAppender">
          <param name="ImplementationClassName"
value="com.bristol.tvision.util.log.snmp.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="org.apache.log4j.PatternLayout">
               <param name="ConversionPattern" value="%d,%p,[%t],[%c],%m%n" />
          </layout>
     </appender>
```

You also need to add a reference to this TRAP_LOG appender in one or more logging categories. For example, to send an SNMP trap when an Analyzer encounters an error, add the following line to the AppLog category in the **Analyzer.Logging** properties:

```
<appender-ref ref="TRAP_LOG"/>
```

The final AppLog category may look like the following:

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

Note: You must add **joesnmp.jar** to the **Additional Classpaths** of your Analyzer configuration 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 the JMS appender 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.

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 that connects 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.

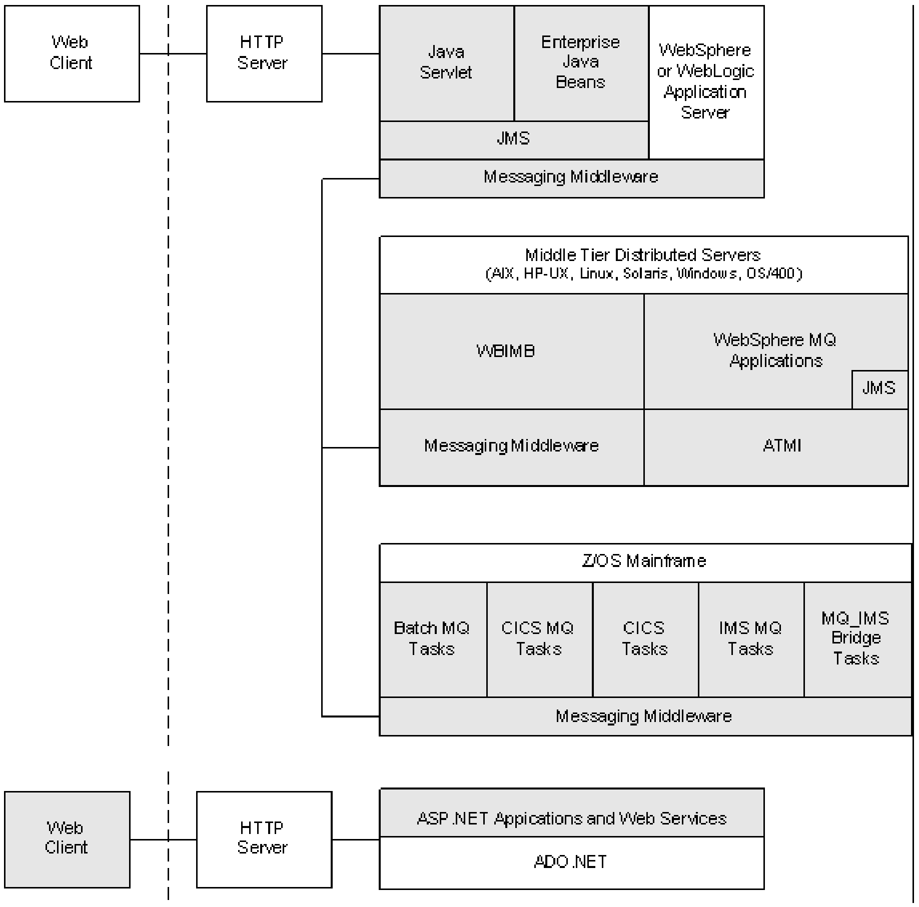
The following table summarizes the limitations per agent.

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	✓		

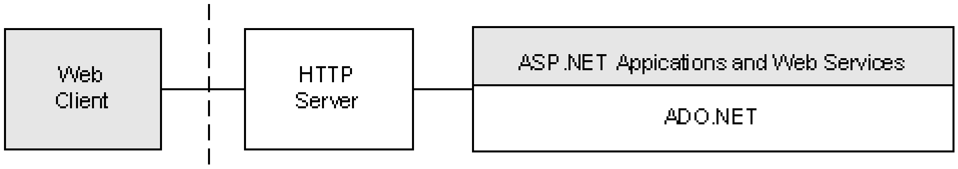
TransactionVision Agent	**Data can be Filtered?**	**Data Used in Transaction Topology?**	**Data Used in Component Topology?**
WebSphere MQ Agent	✓	✓	✓
CICS Agent on z/OS	✓		✓
WMQ Batch, WMQ CICS, WMQ IMS Agents on z/OS	✓	✓	✓
IMS Bridge Agents on z/OS	✓		✓

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.

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, and so on.

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 that run 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, and HTTP_PUT, 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. 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 11, "Installing and Configuring the Java Agent" for details.

About the .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*.

10

Installing and Configuring the WebSphere MQ Agent

This chapter includes:

Installing the WebSphere Agent on Windows

The TransactionVision WebSphere MQ Agent is installed as a single package on Windows. This section 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 MQ 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 280.

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 the product install disks.

➤ Download the Processing Server installation file from the Downloads page in BSM by selecting **Admin** > **Platform** > **Setup and Maintenance** > **Downloads**. Select the **Category** for **WebSphere MQ**.

➤ 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.

Installing the WebSphere MQ Agent on UNIX

This section includes the following:

- "Installation Files for UNIX Platforms" on page 96
- "Installation Steps for UNIX Platforms" on page 96
- "Rebinding the WebSphere MQ Agent on AIX" on page 98

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 HPTVWMQAgent_<version>_zlinux.tgz
Solaris	HPTVWMQAgent_<version>_sol.tgz

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 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 96 for the installation file names. For example:

```
gunzip HPTVWMQAgent_<version>_aix.tgz
tar -xvf HPTVWMQAgent_<version>_aix.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 "Administration Utilities" in *BSM Application Administration Guide*.

Uninstalling the WebSphere MQ Agent on Windows

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.

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.

Configuring WebSphere MQ Agents

TransactionVision provides two types of WebSphere MQ Agents: WebSphere MQ Agent library and WebSphere MQ API Exit Agents. This section describes configuring both types of agents on Windows and UNIX platforms.

For information about configuring this agent on z/OS, see Chapter 13, "z/OS Components–Operation and Customization."

This section includes:

- ➤ "Configuring the WebSphere MQ Agent Library" on page 101
- ➤ "Configuring the WebSphere MQ API Exit Agent" on page 106
- ➤ "WebSphere MQ Agents and FASTPATH_BINDING" on page 114
- ➤ "Configuring TimeSkew Settings for the WebSphere MQ Agent" on page 115
- ➤ "Using Agents with WebSphere MQ Samples" on page 117
- ➤ "WebSphere MQ Client Application Monitoring" on page 117

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:

- ➤ "UNIX and Windows Platforms" on page 102
- ➤ "Configuring Agent Logging" on page 104
- ➤ "Setting the Configuration Queue Name" on page 105
- ➤ "Configuring the WebSphere MQ Messaging System Provider" on page 105

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 to use TransactionVision.

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

All of the directory locations in this table are the default 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" under Transaction Management in the *BSM Application Administration Guide* 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.

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 16, "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 and Windows

On UNIX and Windows 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 and Windows 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 under Transaction Management in the *BSM Application Administration Guide.*

You may create the queues on your queue managers using the 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 114 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 Platforms" on page 107
- "Configuring the API Exit Agent on Windows Platforms" on page 110
- "Identifying Programs to Monitor" on page 112
- "Discarding WebSphere MQ Events on TransactionVision Queues" on page 114

Configuring the API Exit Agent on Distributed Platforms

To use the WebSphere MQ API Exit Agent on distributed platforms, 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=TransactionVisionWMQAgent

  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 "TransactionVisionWMQAgent".

- 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 107). 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 114.

- 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=TransactionVisionWMQAgent
  Sequence=100
  Function=TVisionEntryPoint
  Module=tvisionapiexit
```

Configuring the API Exit Agent on Windows Platforms

To use WebSphere MQ API Exit Agent on Windows, you must perform the following steps:

1 Copy the appropriate WebSphere MQ API Exit Agent DLL files to define WebSphere MQ exit paths.

2 Define the TransactionVision API Exit WebSphere MQ Explorer.

Copying WebSphere MQ API Exit Agent Libraries

Because TransactionVision supports both 32-bit and 64-bit WebSphere MQ programs, you must copy each appropriate API Exit DLL file to the defined WebSphere MQ exits path. By default, these paths are <WebSphereMQInstallPath>\exits for 32-bit and <WebSphereMQInstallPath>\exits64 for 64-bit.

Copy the following API Exit DLL files to their appropriate path:

```
copy <TransactionVisionInstallPath>\lib\tvisionapiexit.dll
<WebSphereMQInstallPath>\exits

copy <TransactionVisionInstallPath>\lib64\tvisionapiexit.dll
<WebSphereMQInstallPath>\exits64
```

Note, the default WebSphere MQ exits paths can be configured to other locations as seen and described, below.

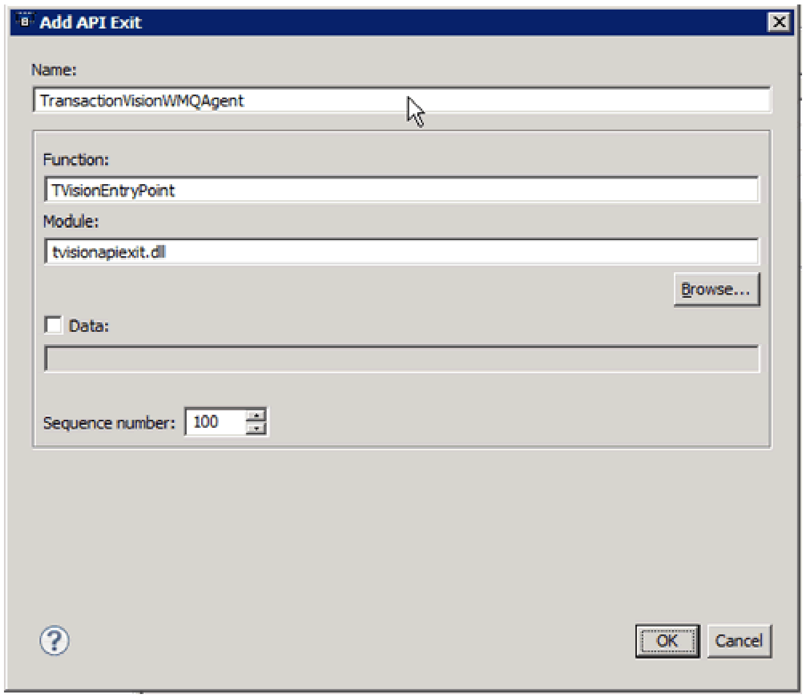
Defining the TransactionVision API Exit in WebSphere MQ Explorer

Configuring API Exits in WebSphere MQ on Windows is done through WebSphere MQ Explorer.

To add and configure the TransactionVision API Exit on a queue manager, follow these steps:

1. In WebSphere MQ Explorer, right click on the queue manager, and select **Properties...**
2. On the Exits page, verify that the exits default paths are configured to the paths that the TransactionVision API Exit DLL files were copied to.
3. Click the **Add...** button.
4. Fill out the API Exit Stanza Values as described in "Stanza Attributes and Values" on page 109.

   Note, even though the Module field has a Browse... button which will fill out the field with the full path, the Module field must contain only the DLL file name with no path specified. The DLL is located by searching for the defined Module file name within the defined default exit paths based on whether the program is 32-bit or 64-bit.



Name: TransactionVisionWMQAgent

Function: TVisionEntryPoint

Module: tvisionapiexit.dll

Data:

Sequence number: 100

5 Click **OK**.

6 Restart the queue manager for the API Exit to be activated.

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.

➤ **wmq_exit_agent.deny** defines which programs will not be monitored. All other programs will be monitored. 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.

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" under Transaction Management in the *BSM Application Administration Guide*.

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.

Configuring TimeSkew Settings for the WebSphere MQ Agent

When the monitored WebSphere MQ queue manager is running on a VMware guest host, it is recommended to use the TransactionVision Time Server as an alternative event time skew mechanism (defined on the WebSphere MQ Communication Link).

The WebSphere MQ Agent has configurable options that can override the default behavior of how it calculates and maintains time skews obtained from the TransactionVision Time Server. You set the options by defining them as environment variables before running the monitored WebSphere MQ application or applications.

You can set the following environment variables to override default Time Server time skew calculation behavior in the WebSphere MQ Agent:

Environment Variable	Description
TVISION_TIMESKEW_DEBUG	**Default**: Not Set Set this environment variable to anything (such as **TVISION_TIMESKEW_DEBUG=1**) to enable debug output in Time Server time skew calculation.
TVISION_TIMESKEW_CHECK_INTERVAL	**Default**: 3600000 (1 hour) Interval in milliseconds in which to periodically check the Time Server TimeSkew.

Environment Variable	Description
TVISION_TIMESKEW_LATENCY_THRESHOLD	**Default**: 200 The number of milliseconds to accept when sampling time skews. If more than the specified time passes when waiting for a time skew reply, another time skew sampling is taken in hopes it has a lower latency. Any time skew samples that surpass the threshold are still kept in the time skew history, because it is possible no samplings are under the suggested latency threshold.
TVISION_TIMESKEW_RETRY_THRESHOLD	**Default**: 8 The maximum number of time skew samples to take at a single time skew check interval. At each time skew check interval, the WebSphere MQ Agent continues to retry up to this number of times if the sampling exceeds the time skew latency threshold.
TVISION_TIMESKEW_HISTORY_SIZE	**Default**: 24 The number of previous time skew samplings to consider for the best time skew to use. Having a larger history size improves the chance of finding a time skew with smaller latency, but it also increases the chance of picking a time skew that may have occurred long ago to represent a time skew that may be more off due to clock drifting. A value of zero (0) causes the time skew that was taken with the smallest latency to be used. This time skew is not discarded until a new time skew with equal or less latency is taken. Using this value disregards clock drift, which occurs frequently.

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

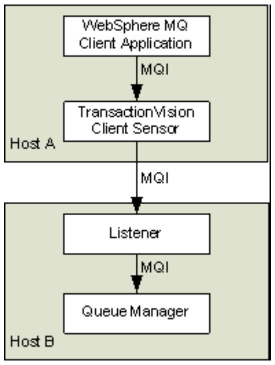
For applications using WebSphere MQ client bindings, TransactionVision is capable of monitoring and tracing these applications' messaging activities in either a distributed or centralized mode.

This section includes:

- "Distributed Monitoring" on page 118
- "Centralized Monitoring" on page 119
- "Installation and Configuration Considerations" on page 121

Distributed Monitoring

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


WebSphere MQ Client Application
MQI
TransactionVision Client Sensor
Host A
MQI
Listener
MQI
Queue Manager
Host B

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.

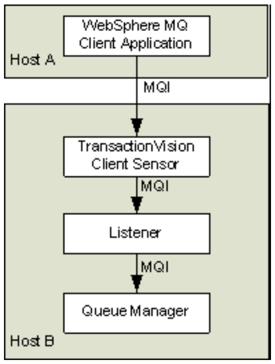
The trace information generated will be based on the client application context. This means information such as program name, program instance, and host name, will be related to the client application directly.

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:


WebSphere MQ Client Application
Host A
MQI
TransactionVision Client Sensor
MQI
Listener
MQI
Queue Manager
Host B

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 Chapter 11, "Installing and Configuring the Java Agent".

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. 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.

11

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 Java Agent Guide*, this pdf is available with the agent installation <agent_install_dir>\DiagnosticsAgent\docs\Diagnostics_Java_Agent_Guide .pdf. The same information is in 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:

- ➤ "Determine if the Predefined Java Agent Will Collect the Desired Events" on page 125
- ➤ "Accessing the Java Agent Installer" on page 125
- ➤ "Installing the Java Agent" on page 127
- ➤ "Running the Java Agent Setup Module" on page 128
- ➤ "Preparing the Application Server for Monitoring" on page 135
- ➤ "Additional Configuration of the Applications to Be Monitored" on page 137
- ➤ "Setting Up Messaging Queues and Communication Links" on page 140

You can also perform the following optional tasks:

- ➤ "Configuring Custom User Events (Optional)" on page 142
- ➤ "Silent Installation of the Java Agent (Optional)" on page 146

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 "Configuring Custom User Events (Optional)" on page 142.

Accessing the Java Agent Installer

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

Platform	**Files**
Windows	HPDiagTVJavaAgt_<version>.zip

Platform	Files
z/OS	HPDiagTVJavaAgt_<version>_zos.tgz
All other platforms, including AIX, HP-UX, Linux, and Solaris	HPDiagTVJavaAgt_<version>.zip or HPDiagTVJavaAgt_<version>.tgz

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

To access the installer:

1 Copy the Java Agent installation package to the target host. You typically obtain the package from one of the following locations:

- ➤ The HP Diagnostics release media.
- ➤ The Software Patches page on the HP Software Support Online web site at http://support.openview.hp.com/selfsolve/patches. Go to **Software Support Online** > **Downloads** > **Software Patches** and select Diagnostics as the product.

Note: The most recent versions of Diagnostics are posted here.

- ➤ The Downloads page in BSM; select **Admin** > **Platform Administration** > **Setup and Maintenance** > **Category** > **Diagnostics**.

2 Continue with "Installing the Java Agent" on page 127.

Installing the Java Agent

The following steps provide detailed instructions for a first time installation of the Java Agent on Windows or UNIX systems.

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

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 282.

For information on other types of installation, see the appropriate section:

- "Installing the Java Agent on a z/OS Mainframe" on page 143
- "Installing the Java Agent Using the Generic Installer" on page 145
- "Silent Installation of the Java Agent (Optional)" on page 146

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

For the UNIX installer the following applies:

1. Extract all content of the installation package to a directory on the target host.

   If you are extracting the **.tgz** package for Unix systems, use the following command to extract the files with their permissions:
   tar -pxvzf HPDiagTVJavaAgt_<release number>.tgz

Note: Do not extract the zip contents to a temp directory.

   Within the extracted files you see the **JavaAgent/DiagnosticsAgent/** directory. This location is hereafter referred to as <agent_install_directory>.

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 145.

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.

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.

Running the Java Agent Setup Module

The Java Agent can be configured as a Profiler without any connection to a Diagnostics Server (or as an agent that works with a Diagnostics Server and/ or TransactionVision Server). When the agent is initially configured as a Profiler only, you can, at a later time, configure the agent to work with a Diagnostics server by re-running the Java Agent Setup Module.

You can start the Java Agent Setup Module at any time on Windows systems by running:

<agent_install_directory>\setup.cmd

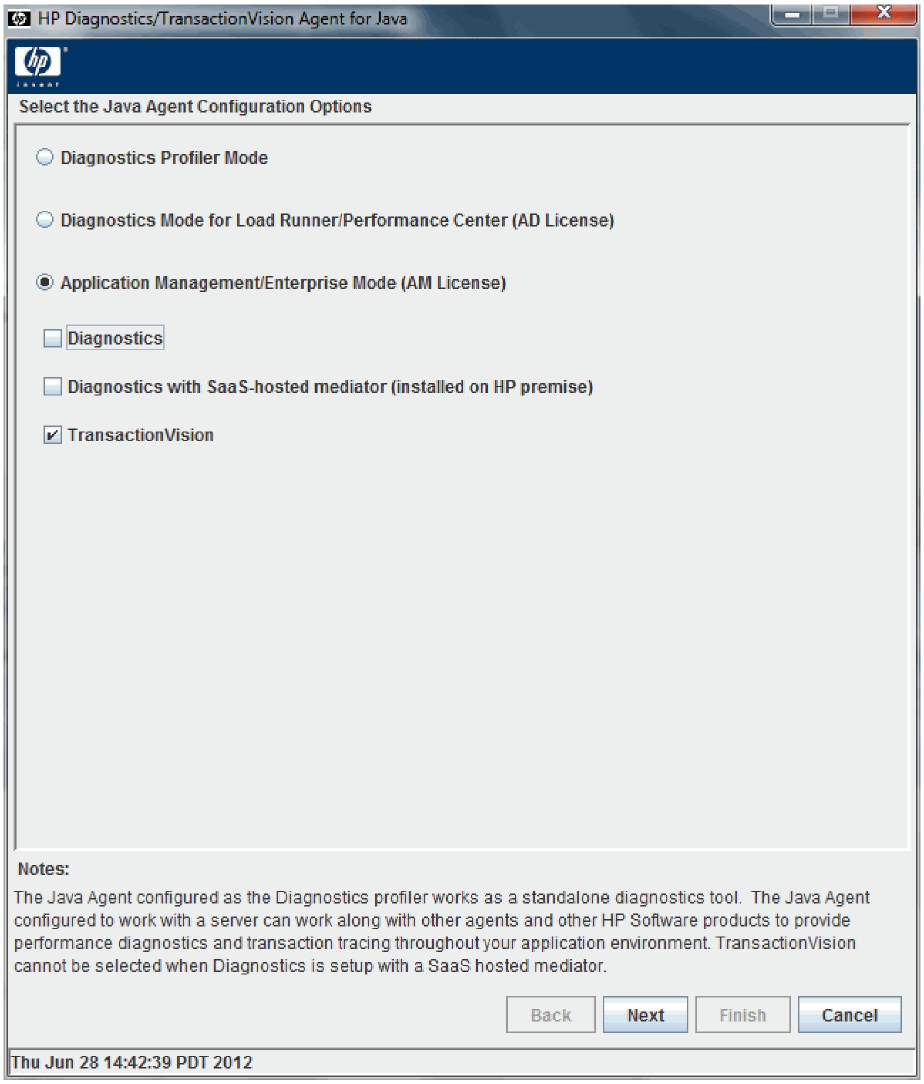
You can start the Java Agent Setup Module on UNIX systems at any time by running:

<java_agent_install_dir>/DiagnosticsAgent/bin/setupModule.sh

where <**java_agent_install_dir**> refers to the path of your Java Agent installation directory.

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

1 Select the **Application Management/Enterprise Mode (AM License)** and **TransactionVision** options to install the agent for use with a TransactionVision Server in an enterprise (or production) environment:



Select the Java Agent Configuration Options

○ Diagnostics Profiler Mode

○ Diagnostics Mode for Load Runner/Performance Center (AD License)

● Application Management/Enterprise Mode (AM License)

☐ Diagnostics

☐ Diagnostics with SaaS-hosted mediator (installed on HP premise)

☑ TransactionVision

Notes:

The Java Agent configured as the Diagnostics profiler works as a standalone diagnostics tool. The Java Agent configured to work with a server can work along with other agents and other HP Software products to provide performance diagnostics and transaction tracing throughout your application environment. TransactionVision cannot be selected when Diagnostics is setup with a SaaS hosted mediator.

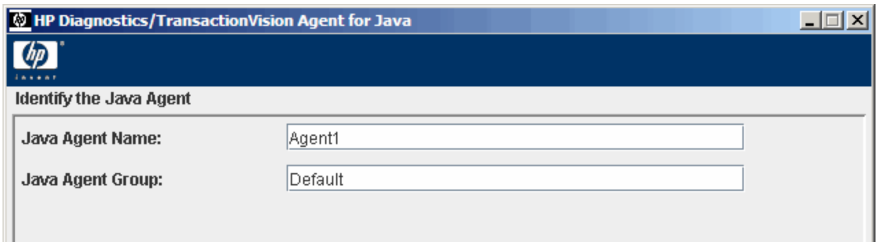
Thu Jun 28 14:42:39 PDT 2012

These options set the value of the **active.properties** property in the **etc/probe.properties** file to **TV** mode.

Diagnostics Information: If you are configuring the Java Agent as a Diagnostics Profiler only or as a Java Agent reporting to a Diagnostics Server, see "Running the Java Agent Setup Module" in the *HP Diagnostics Java Agent Guide* for details on setup options specific to HP Diagnostics. This pdf is available with the agent installation <agent_install_dir>\DiagnosticsAgent\docs\Diagnostics_Java_Agent_Guide.pdf.

Click **Next** (in console mode **Enter**) to proceed.

2 Assign a name to the Java Agent and specify the group to which it belongs. If this is a TransactionVision Agent installation only, simply accept the default settings by pressing the Next button to continue with your configuration.


HP Diagnostics/TransactionVision Agent for Java

Identify the Java Agent

Java Agent Name: Agent1

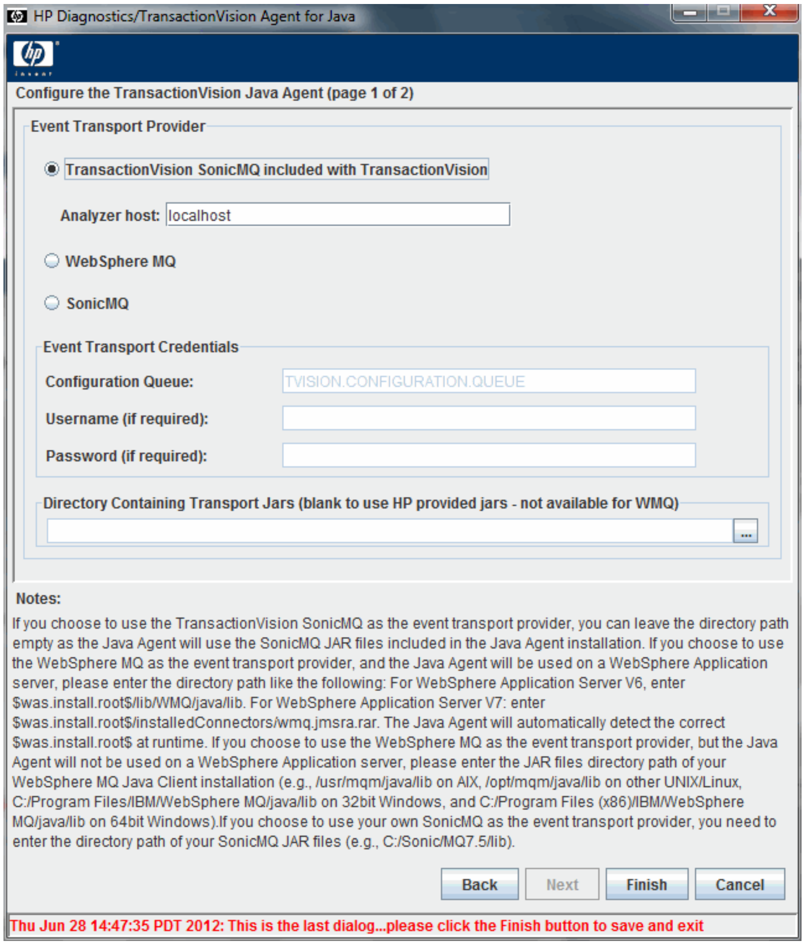
Java Agent Group: Default

Refer to to "Running the Java Agent Setup Module" in the *HP Diagnostics Java Agent Guide* for details on setting up Java Agent Name and Java Agent Group for HP Diagnostics.

Click **Next** (in console mode **Enter**) 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.

In console mode enter **Y** for one of the event transport providers and specify the credentials. Enter N to skip the other transport providers.



HP Diagnostics/TransactionVision Agent for Java

Configure the TransactionVision Java Agent (page 1 of 2)

Event Transport Provider

(●) TransactionVision SonicMQ included with TransactionVision

Analyzer host: localhost

( ) WebSphere MQ

( ) SonicMQ

Event Transport Credentials

Configuration Queue: TVISION.CONFIGURATION.QUEUE

Username (if required):

Password (if required):

Directory Containing Transport Jars (blank to use HP provided jars - not available for WMQ)

Notes:

If you choose to use the TransactionVision SonicMQ as the event transport provider, you can leave the directory path empty as the Java Agent will use the SonicMQ JAR files included in the Java Agent installation. If you choose to use the WebSphere MQ as the event transport provider, and the Java Agent will be used on a WebSphere Application server, please enter the directory path like the following: For WebSphere Application Server V6, enter $was.install.root$/lib/WMQ/java/lib. For WebSphere Application Server V7: enter $was.install.root$/installedConnectors/wmq.jmsra.rar. The Java Agent will automatically detect 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 a WebSphere Application server, please enter the JAR files directory path of your WebSphere MQ Java Client installation (e.g., /usr/mqm/java/lib on AIX, /opt/mqm/java/lib on other UNIX/Linux, C:/Program Files/IBM/WebSphere MQ/java/lib on 32bit Windows, and C:/Program Files (x86)/IBM/WebSphere MQ/java/lib on 64bit Windows).If you choose to use your own SonicMQ as the event transport provider, you need to enter the directory path of your SonicMQ JAR files (e.g., C:/SonicMQ7.5/lib).

Thu Jun 28 14:47:35 PDT 2012: This is the last dialog...please click the Finish button to save and exit

➤ 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.

Note: If your monitored Java application or application server is run by a user different from the user that installs the Java Agent, your Java Agent installation may not have the proper write permission set on the following directory when using WebSphere MQ as the event transport. To enable access to the directory, set the owner of this directory to the user that runs the application server (if the application server is the only thing to be monitored), or enable the write permission on the following directory to a group and/or others that run the application server to be monitored.
<JavaAgent_Home>/TransactionVisionAgent/integrations/OOBMQ

➤ **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, and the Java Agent is to be used on other application servers, enter the JAR files directory path of your WebSphere MQ Java Client installation:

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

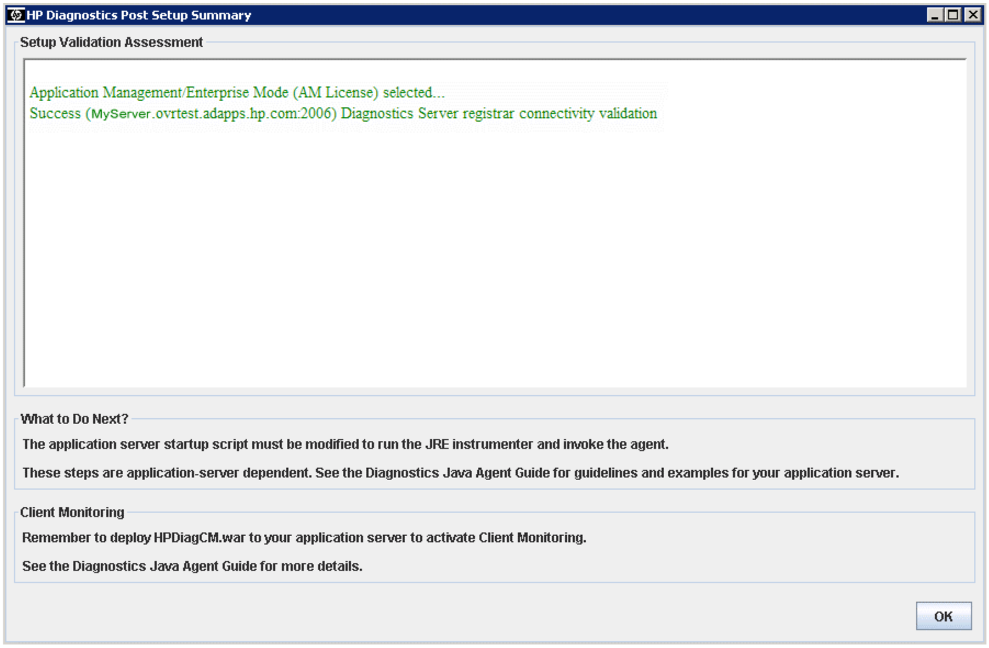
    For 64-bit Windows: **C:/Program Files (x86)/IBM/WebSphere MQ/java/lib**

- Click **Finish** to continue. In console mode enter 0 (zero) to complete the setup and when prompted to save your changes to the Java Agent Setup Module, enter **Y**.

If you selected to setup the agent for Diagnostics or for use in a SaaS environment then you will see additional dialog boxes in the Java Agent Setup Module. See the *HP Diagnostics Java Agent Guide* for details on these installation options.

4 Post Setup Summary

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


HP Diagnostics Post Setup Summary
Setup Validation Assessment

Application Management/Enterprise Mode (AM License) selected...
Success (MyServer.ovrtest.adapps.hp.com:2006) Diagnostics Server registrar connectivity validation

What to Do Next?

The application server startup script must be modified to run the JRE instrumenter and invoke the agent.

These steps are application-server dependent. See the Diagnostics Java Agent Guide for guidelines and examples for your application server.

Client Monitoring

Remember to deploy HPDiagCM.war to your application server to activate Client Monitoring.

See the Diagnostics Java Agent Guide for more details.

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

Depending on your application server, you should use the automatic JRE instrumentation options instead of manually running the JRE Intrumenter utility. Therefore the checkbox for running the JRE Instrumenter utility is left blank by default.

See "Preparing the Application Server for Monitoring" on page 135 to continue with the next steps to instrument the JRE used by your application server and configure the JVM paramerts to invoke the Java Agent.

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

Preparing the Application Server for Monitoring

The next steps are to instrument the JRE used by your application server and configure your application server JVM parameters to invoke the Java Agent.

Follow the instructions in "Preparing Application Servers for Monitoring with the Java Agent"of the *HP Diagnostics Java Agent Guide* for how to instrument the JRE used by your application server and how to configure the JVM parameters for specific application servers to invoke the Java Agent. This chapter of the Diagnostics manual includes detailed steps for instrumenting most commonly used application servers such as JBoss, Tomcat, WebLogic and WebSphere. The *HP Diagnostics Java Agent Guide* pdf is available with the agent installation <agent_install_dir>\DiagnosticsAgent\docs\Diagnostics_Java_Agent_Guide .pdf.

After you prepare the application server for monitoring by the Java Agent then you restart the application server and the Java Agent will be invoked to begin monitoring the application.

Instrumentation Overview

The JRE Instrumenter is a utility to instrument a JRE so that the Java Agent can provid advanced features such as the patent-pending Collection Leak Pinpointing (CLP). It does not modify the installed JRE in any way, but rather places copies of instrumented classes somewhere under the <JavaAgent_install_dir>/DiagnosticsAgent/classes directory. You can use the JRE Instrumenter to instrument multiple JREs if they are installed on your system.

The JRE Instrumenter instruments some standard Java classes used by the application server JVM and applications running on it. 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.

With different command-line options, the JRE Instrumenter can be invoked and used in three different ways, each of which has its own advantages and limitations. You will use one of these methods according to the characteristics of your application servers.

- **Automatic Explicit Mode.** If your application server is or can be started by a script, it is recommended that you use this mode. To use this mode, you add a line to your application server startup script to explicitly and non-interactively run the JRE Instrumenter to instrument the JRE. Your script will continue to start the application server JVM (with additional parameters) using the freshly instrumented JRE.
- **Automatic Implicit Mode.** With this mode, you do not need to explicitly run the JRE Instrumenter — you only need to modify your application server JVM parameters to invoke the Java Agent and ask it to run the JRE Instrumenter as needed. When the Java Agent is used for the first time, it implicitly runs the JRE Instrumenter to instrument the JRE. However, the first time this instrumented JRE will not be used; your application server will be using an uninstrumented JRE. The next time your application server is started, it will use the instrumented JRE. Therefore, if you want to use the full monitoring features of the Java Agent, you need to restart your application server twice after you enable the Java Agent.

- **Manual Mode.** With this mode, you need to manually and interactively run the JRE Instrumenter, either at the end of the Java Agent installation or at a later time, to instrument the JRE. You then modify your application server JVM parameters according to the parameters provided by the JRE Instrumenter. This method is how the JRE Instrumenter works in earlier versions of the Java Agent.

Additional Configuration of the Applications to Be Monitored

This section includes the following topics:

- "Enabling Java Agent to Collect JMS Events on JBoss" on page 137
- "Enabling Java Agent to Monitor Jetty" on page 139
- "Enabling Java Agent to Monitor Multiple JVMs Running on the Same Machine" on page 140

For troubleshooting tips on monitoring WebSphere MQ 7.0 JMS and TIBCO, see:

- "Troubleshooting for Monitoring WebSphere MQ 7.0 JMS" on page 149
- "Troubleshooting for Monitoring TIBCO ActiveMatrix BusinessWorks and Service Bus" on page 150

Enabling Java Agent to Collect JMS Events on JBoss

For the Java Agent to collect JMS events on the JBoss Application Server platforms, you may need to implement a workaround to resolve a class loading issue if you get **ClassNotFoundException** in the **probe.log**. This workaround involves the following two changes:

- Add the path to certain JMS provider JAR files to the **appSensorLoadPath** property in the **<JavaAgent_Home>/DiagnosticsAgent/etc/TV.properties** file.

For example, for WebSphere MQ 6.0 on 32-bit Windows, your **appSensorLoadPath** property may look similar to the following with highlighted changes:

```
appSensorLoadPath=appCL.jar;appOrProbeCL.jar;platformImpl_appCL.jar;cal
lbacks.jar;lwcrypto.jar;bcprov-jdk14-127.jar;appOrProbeCL_jdk15.jar;C:/
Program Files/IBM/WebSphere MQ/java/lib/com.ibm.mq.jar;C:/Program
Files/IBM/WebSphere MQ/java/lib/com.ibm.mqjms.jar
```

For example, for WebSphere MQ7.0 on 64-bit Windows, your **appSensorLoadPath** property may look similar to the following with highlighted changes:

```
appSensorLoadPath=appCL.jar;appOrProbeCL.jar;platformImpl_appCL.jar;cal
lbacks.jar;lwcrypto.jar;bcprov-jdk14-127.jar;appOrProbeCL_jdk15.jar;C:/
Program Files (x86)/IBM/WebSphere MQ/java/lib/com.ibm.mq.jar;C:/Program
Files (x86)/IBM/WebSphere MQ/java/lib/com.ibm.mq.jmqi.jar;C:/Program Files
(x86)/IBM/WebSphere MQ/java/lib/com.ibm.mqjms.jar
```

Note: You must use a forward slash (/) instead of a back slash (\) and a semicolon (;) instead of a colon (:) in this property, no matter what platform is used.

➤ For JBoss 5.1 or higher, you need to create a file called **jboss-classloading.xml** with the following content and add it to the **WEB-INF** directory of your web applications (for example, **myapp.war/WEB-INF/jboss-classloading.xml**). This also applies to the web applications embedded in an enterprise application (EAR).

```
<classloading xmlns="urn:jboss:classloading:1.0"
    parent-first="true"
    import-all="true">
</classloading>
```

Enabling Java Agent to Monitor Jetty

Java Agent provides a generic mechanism to monitor Jetty or other Servlet containers that are not officially supported. This generic mechanism is disabled by default. Therefore, if you want to monitor the Servlet activities on Jetty, you need to enable it by modifying the following property in the **<JavaAgent_Home>/TransactionVisionAgent/config/sensor/ generic.properties** file:

```
enable=true
```

If you get errors such as the following error, there is a workaround:

java.lang.NoClassDefFoundError: javax/servlet/http/HttpServletRequest

To workaround this issue, modify the **appSensorLoadPath** property in the **<JavaAgent_Home>/DiagnosticsAgent/etc/TV.properties** file. Following is an example with highlighted changes:

```
appSensorLoadPath=appCL.jar;appOrProbeCL.jar;platformImpl_appCL.jar;callbacks.ja
r;lwcrypto.jar;bcprov-jdk14-127.jar;appOrProbeCL_jdk15.jar;C:/jetty-6.1.26/lib/
servlet-api-2.5-20081211.jar
```

Note: This generic Servlet monitoring mechanism does not capture the Servlet payload. Therefore, you get the HTTP headers and Servlet parameters and other information, but you do not get the Servlet request or response data. If you want to capture the Servlet payload or other special customer data, you can use the advanced Java Agent features described in the *HP TransactionVision Advanced Customization Guide* PDF.

Enabling Java Agent to Monitor Multiple JVMs Running on the Same Machine

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 "Configure Monitoring of Multiple Java Processes on an Application Server" in the *HP Diagnostics Java Agent Guide*. This pdf is available with the agent installation <agent_install_dir>\DiagnosticsAgent\docs\Diagnostics_Java_Agent_Guide .pdf.

Setting 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:

➤ "WebSphere MQ" on page 141

➤ "Progress SonicMQ" on page 142

WebSphere MQ

You create the queues on your queue managers using the 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.

Special Configuration for Java Agents that use WebSphere MQ JMS on the IBM System zSeries

Java agents that use WebSphere MQ JMS on the IBM System zSeries platform require the following manual configuration.

1 Add the following property to the **<JavaAgent_Home>/ DiagnosticsAgent/etc/TV.properties** file.

```
#force to use server binding when using WMQ JMS as transport
tvagent.sysprop.switch.use.server.binding=true
```

2 Replace the three WMQ JAR files in **<JavaAgent_Home>/ TransactionVisionAgent/integrations/OOBMQ/HP** with the ones from **<WAS_HOME/>installedConnectors/wmq.jmsra.rar**.

3 Modify the **probeTransportLoadPath** property in **TV.properties** to use the full path to each WMQ JAR file. For example:

```
probeTransportLoadPath=jms.jar;/usr/lpp/zWebSphere/V7R0/installedConnectors/
wmq.jmsra.rar/com.ibm.mqjms.jar;/usr/lpp/zWebSphere/V7R0/installedConnectors/
wmq.jmsra.rar/com.ibm.msg.client.jms.jar;/usr/lpp/zWebSphere/V7R0/installedConnectors/
wmq.jmsra.rar/com.ibm.msg.client.jms.internal.jar
```

4 Add the <WMQ home>/java/lib to the JVM custom property **java.library.path**. You can get the current **java.library.path** property value from the probe.log file. For example:

```
/wasv7config/xscell/xsnode1/AppServer/java64/lib/s390x/default:/wasv7config/xscell/xsnode1/
AppServer/java64/lib/s390x:/wasv7config/xscell/xsnode1/AppServer/java64/bin:/wasv7config/
xscell/xsnode1/AppServer/java64/bin/j9vm:/wasv7config/xscell/xsnode1/AppServer/java64/lib/
s390/j9vm:/wasv7config/xscell/xsnode1/AppServer/java64/lib/s390:/wasv7config/xscell/xsnode1/
AppServer/java64/lib/s390x/j9vm:/wasv7config/xscell/xsnode1/AppServer/lib:/wasv7config/xscell/
xsnode1/AppServer/java64/jre/bin:/wasv7config/xscell/xsnode1/AppServer/java64/jre/lib/s390/
j9vm:/wasv7config/xscell/xsnode1/AppServer/java64/jre/lib/s390x/j9vm:/wasv7config/xscell/
xsnode1/AppServer/java64/jre/lib/s390x:/VERSYSB/usr/lpp/mqm/V7R0M1/java/lib
```

Progress SonicMQ

If you are using the built-in 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.

Configuring Custom User Events (Optional)

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.

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 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 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_9.20_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_9.20_zos.tar
```

This command creates the unpacked directory, **javaAgent**.

4 Ensure that you have a Java executable on your path, and then 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 "Running the Java Agent Setup Module" on page 128. For example, (or as appropriate for your shell):

And then:

```
<probe_install_dir>/bin/setupModule.sh
```

5 Configure the Application Server to allow the agent to monitor the application, see "Preparing the Application Server for Monitoring" on page 135

6 If the agent is configured to work with the other Diagnostics components, verify the agent installation works with these other components

7 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 administrator 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 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 "Running the Java Agent Setup Module" on page 128.

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

5 Configure the Application Server to allow the agent to monitor the application, see "Preparing the Application Server for Monitoring" on page 135

6 If the agent is configured to work with the other Diagnostics components, verify the agent installation works with these other components.

Silent Installation of the Java Agent (Optional)

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.

Troubleshooting Java Agent

This section provides troubleshooting tips for the TransactionVision Java Agent. For more indepth troubleshooting tips, see "Troubleshooting HP Diagnostics" in the *HP Diagnostics Installation and Configuration Guide*.

This section includes:

➤ "Java Agent Support Collector" on page 149

➤ "Troubleshooting for Monitoring WebSphere MQ 7.0 JMS" on page 149

➤ "Troubleshooting for Monitoring TIBCO ActiveMatrix BusinessWorks and Service Bus" on page 150

Java Agent Support Collector

The **runSupportSnapshot** utility creates a .zip file containing the entire set of files relevant to troubleshooting one or more instances of the Java agent in a Diagnostics or TransactionVision deployment environment.

The .zip file contains the following:

➤ Files from the <Diagnostics_probe_install_dir>\etc directory

➤ Files from the <Diagnostics_probe_install_dir>\log directory

➤ Files from the <TransactionVision_sensor_install_dir>\config directory

➤ Files from the <TransactionVision_sensor_install_dir>\logs directory

➤ Property Scanner report, which compares two agent directories and reports differences between property files, points files, and XML files (TransactionVision Agents only).

➤ Probe instance information, including property settings. For agents running in 1.5 JVMs, environment variables, stack dumps, and class loader information is also included.

To run runSupportSnapshot:

1 Navigate to **<Diagnostics_probe_install_dir>\contrib\JASMUtilities\Snapins**.

2 Run .**runSupportSnapshot.cmd -console** on Windows, or **./runSupportSnapshot.sh -console** on UNIX or Linux.

A .zip file is created.

Troubleshooting for Monitoring WebSphere MQ 7.0 JMS

Java Agent requires the **com.ibm.mq.jar** file to be in an application's class path. If your application that uses the WebSphere MQ 7.0 client does not have this file in its class path or **WEB-INF/lib** directory, you may get an error such as **NoClassDefFoundError: com/ibm/mq/MQEnvironment** in the **probe.log**. In this scenario, use the following workaround:

Add the full path to **com.ibm.mq.jar** to the **appSensorLoadPath** property in the **<JavaAgent_Home>/DiagnosticsAgent/etc/TV.properties** file.

For example, for WebSphere MQ 7.0 on 32-bit Windows, your **appSensorLoadPath** property may look similar to the following with highlighted changes:

```
appSensorLoadPath=appCL.jar;appOrProbeCL.jar;platformImpl_appCL.jar;cal
lbacks.jar;lwcrypto.jar;bcprov-jdk14-127.jar;appOrProbeCL_jdk15.jar;C:/
Program Files/IBM/WebSphere MQ/java/lib/com.ibm.mq.jar;C:/Program
Files/IBM/WebSphere MQ/java/lib/com.ibm.jmqi.jar;C:/Program Files/
IBM/WebSphere MQ/java/lib/com.ibm.mqjms.jar
```

Note: You must use a forward slash (/) instead of a back slash (\) and a semicolon (;) instead of a colon (:) in this property, no matter what platform is used.

Troubleshooting for Monitoring TIBCO ActiveMatrix BusinessWorks and Service Bus

When you use the Java Agent to monitor TIBCO ActiveMatrix BusinessWorks or Service Bus, you may encounter errors such as the following error in some TIBCO components:

java.lang.NoClassDefFoundError: com/bristol/tvision/appCL/sensor/....

You can work around this issue by setting the **tvagent.sysprop.switch.use.extloader** property in the **<JavaAgent_Home>/DiagnosticsAgent/etc/TV.properties** file to **true** as follows:

```
# Use the extension class loader
tvagent.sysprop.switch.use.extloader=true
```

After making the above change, if you use Tomcat in your environment you may get another error such as the following:

java.lang.ClassNotFoundException: javax.servlet.http.HttpServletReque

You can work around this issue by appending the path to the **servlet-api.jar** file to the **appSensorLoadPath** property in the **<JavaAgent_Home>/ DiagnosticsAgent/etc/TV.properties** file. Following is an example with highlighted changes:

```
appSensorLoadPath=appCL.jar;appOrProbeCL.jar;platformImpl_appCL.jar;cal
lbacks.jar;lwcrypto.jar;bcprov-jdk14-127.jar;appOrProbeCL_jdk15.jar;C:/
apache-tomcat-7.0.8/lib/servlet-api.jar
```

For instructions on instrumenting TIBCO ActiveMatrix Business Works and Service Bus, see "Configuring TIBCO ActiveMatrix/Business Works" in the *HP Diagnostics Java Agent Guide*. This pdf is available with the agent installation
<agent_install_dir>\DiagnosticsAgent\docs\Diagnostics_Java_Agent_Guide
.pdf.

12

Installing and Configuring Agents on z/OS

This chapter includes:

z/OS Agent Installation Summary

This chapter provides information and instructions on installing and configuring TransactionVision Agents for z/OS.

Installation of the agents is a three-step process consisting of the following:

- Transfer of z/OS agent install libraries to z/OS system using FTP.
- SMPE installation of TransactionVision z/OS product libraries on z/OS systems.
- Post install agent configuration of desired agents to be used.

All z/OS agents are installed during a single SMPE installation process.

The z/OS agent types provided consist of the following:

- CICS Agent - Provides ability to collect CICS application activity for various CICS API requests executed by CICS transactions.
- CICS WMQ Agent - Provides ability to collect CICS application WMQ activity for WMQ API requests executed by CICS transactions.
- Batch WMQ Agent - Provides ability to collect z/OS batch job WMQ application activity executed by batch jobs.
- IMS WMQ Agent - Provides ability to collect IMS application WMQ activity for WMQ API requests executed by IMS transactions and IMS batch jobs.
- IMS WMQ Bridge Agent - Provides ability to collect IMS WMQ activity for WMQ requests executed when using the WMQ MQ-IMS Bridge.

Transfer z/OS Agent Distribution Files to z/OS System

Perform the following steps to transfer z/OS Agent distribution files to a z/OS system:

1 Locate the **@README** file located within the file folder created from extracting the z/OS Agent installation ZIP file obtained from the installation media.

2 Follow the instructions contained within the **@README** file to transfer and decompress the z/OS Agent SMPE installation datasets on the z/OS system.

Install z/OS Agents Using SMPE

Perform the following steps to install z/OS Agents using SMPE:

1 Locate the **@README** member within the SMPINST dataset which was created on your z/OS system during the preceding FTP process.

2 Follow the instructions contained within the **@README** member to SMPE install the z/OS Agents on the z/OS system. Necessary installation jobs are located within the SMPINST dataset.

Post Install Configuration for All Agent Types

Perform the following steps that are required and applicable to all agent types:

1. APF authorize the SSLDAUTH load library dataset which was created during the SMPE installation process.
2. Locate member TVISION in the SSLDPROC installation dataset and do the following:
   - **a** Follow the instructions contained within the member TVISION.
   - **b** Copy the procedure to an available PROCLIB dataset within your PROCLIB concatenation.

After completing these steps, proceed to complete the post install for the individual agent types that you are intending to enable and use.

Post Install Configuration for CICS Agent

The following steps must be completed for each CICS region for which you are planning to use the CICS agent.

Perform the following required configuration steps when using the CICS Agent:

1. Add CICS resource definitions (transactions and programs) required for the TransactionVision CICS agent:
   - **a** Locate the member name CICSCSD in the SSLDINST installation dataset.
   - **b** Follow the instructions contained within the member.

2 Add the SSLDLOAD installation dataset name to your CICS region DFHRPL load library concatenation. The SSLDLOAD dataset contains CICS load modules used by the CICS agent.

Note: Step 2 is also required for the WMQ-CICS agent. If you are planning to use and have already configured the WMQ-CICS Agent, then this step has already been completed.

---

3 Locate the member TVISIONC in the SSLDPROC installation dataset and do the following:

a Follow the instructions contained within the member TVISIONC.

b Copy the procedure to an available PROCLIB dataset within your PROCLIB concatenation.

After completing the post install configuration of all desired agent types, proceed to "Post Install Define WMQ Queues" on page 160.

Post Install Configuration for WMQ-CICS Agent

The TVision WMQ-CICS agent requires use of the CICS-WMQ Crossing Exit. The crossing exit program name is CSQCAPX. The current CICS-WMQ architecture (provided by IBM) only supports a single concurrent exploiter of the crossing exit.

The crossing exit implementation is unlike the CICS implementation of CICS Global Exits (and CICS Task Related User exits) which enable multiple interested parties to be concurrently active at a single global exit point.

If you currently have another vendor product or internally developed application that is activating and using the CICS-WMQ Crossing exit contact your HP TransactionVision Marketing representative or HP Software Support.

The following steps must be completed for each CICS region for which you are planning to use the WMQ-CICS agent.

Perform the following required configuration steps when using the WMQ-CICS agent:

1. Add CICS resource definitions (transactions and programs) required for TVision WMQ-CICS agent:
   - **a** Locate member name CWMQCSD in the SSLDINST installation dataset.
   - **b** Follow the instructions contained within the member.
2. Add the SSLDLOAD installation dataset name to your CICS region DFHRPL load library concatenation. The SSLDLOAD dataset contains CICS load modules used by the WMQ-CICS agent.

   Note: Step 2 is also required for the CICS agent. If you are planning to use and have already configured the CICS Agent then this step has already been completed.

   ---

3. Locate member TVISIONQ in the SSLDPROC installation dataset and do the following:
   - **a** Follow the instructions contained within the member TVISIONQ.
   - **b** Copy the procedure to an available PROCLIB dataset within your PROCLIB concatenation.

After completing the post install configuration of all desired agent types, proceed to "Post Install Define WMQ Queues" on page 160.

Post Install Configuration for WMQ-Batch Agent

Perform the following required configuration steps when using the WMQ-BatchAgent:

1. Locate member TVISIONM in the SSLDPROC installation dataset and follow the instructions contained within the member.
2. Copy the procedure to an available PROCLIB dataset within your PROCLIB concatenation.

Note: These same steps are also required for the WMQ-IMS agent. If you are planning to use and have already configured the WMQ-IMS Agent then these steps have already been completed.

After completing the post install configuration of all desired agent types, proceed to "Post Install Define WMQ Queues" on page 160.

Post Install Configuration for WMQ-IMS Agent

Perform the following required configuration steps when using the WMQ-IMS agent.

1. Locate member TVISIONM in the SSLDPROC installation dataset and follow the instructions contained within the member.
2. Copy the procedure to an available PROCLIB dataset within your PROCLIB concatenation.

Note: These same steps are also required for the WMQ-Batch agent. If you are planning to use and have already configured the WMQ-Batch Agent then these steps have already been completed.

After completing the post install configuration of all desired agent types, proceed to "Post Install Define WMQ Queues" on page 160.

Post Install Configuration for WMQ MQ-IMS Bridge Agent

The TransactionVision WMQ MQ-IMS Bridge Agent uses the IMS OTMA Input/Output Edit exit routine (**DFSYIOE0**) to collect event activity.

To enable and use this agent you must enable use of the OTMA Input/Output Edit exit routine in the desired IMS subsystems.

If you currently have another vendor product or internally developed application that is activating and using the IMS OTMA Input/Output Edit exit routine, contact your HP TransactionVision Marketing representative or HP Software Support.

Perform the following required configuration steps when using the WMQ MQ-IMS Bridge Agent.

1. Locate member TVISIONB in the SSLDPROC installation dataset and do the following:
   - **a** Follow the instructions contained within the member TVISIONB.
   - **b** Copy the procedure to an available PROCLIB dataset within your PROCLIB concatenation.
2. Locate member TVISIOND in the SSLDPROC installation dataset and do the following:
   - **a** Follow the instructions contained within the member TVISIOND.
   - **b** Copy the procedure to an available PROCLIB dataset within your PROCLIB concatenation.
3. The SSLDAUTH installation dataset contains the IMS OTMA Input/Output Edit exit load module named **DFSYIOE0**. The SSLDAUTH load library must be added to all applicable IMS control region STEPLIBs, or the **DFSYIOE0** module contained in SSLDLOAD must be copied to an existing dataset in the IMS control region STEPLIB dataset concatenation.

After completing the post install configuration of all desired agent types, proceed to "Post Install Define WMQ Queues" on page 160.

Post Install Define WMQ Queues

All z/OS Agent types use WMQ queues as the transport of event activity collected by the agents. The queues names in use are defined within a TransactionVision communication link which is assigned to a TransactionVision Analyzer on a distributed platform. The TransactionVision Analyzer and the z/OS Agents use WMQ queues to communicate with one another.

Following are the two WMQ queues used by the z/OS Agents:

- **configuration queue.** The configuration queue default name is **TVISION.CONFIGURATION.QUEUE**. All z/OS Agents obtain collection configuration information by reading the configuration queue. The configuration queue name and the z/OS WMQ Queue Manager name it resides on is passed to the applicable z/OS Agent started task using a JCL parameter.

  Information derived from the configuration queue includes the event queue name to be used for event transport, collection filtering information, and additional runtime related parameter data.

- **event queue.** The event queue default name is **TVISION.EVENT.QUEUE**. The event queue is used by the TransactionVision agent manager started tasks to transport collected event activity to the TransactionVision Analyzer component residing on a distributed system. The Analyzer reads event data from the event queue and the z/OS Agent writes event data to the event queue.

The actual WMQ definition of the configuration and event queues are dependent upon how you choose to define and implement queue access. The queues could be defined as z/OS local queues and the TransactionVision Analyzer then accesses the queues using a WMQ Client connection. The queues could also be defined as z/OS remote queues in which case the actual queues reside on a distributed system allowing direct, or indirect access from the TransactionVision Analyzer.

It is important that you communicate with distributed staff members and understand how you intend to physically implement the queues. Once that is understood, then define the queues on the applicable z/OS WMQ subsystems.

For assistance in defining the z/OS WMQ queues, locate member DEFQUEUE in the SSLDINST dataset and follow the instructions provided.

Post Install z/OS Security Requirements

Following are the z/OS security authorizations required by TransactionVision for z/OS Agents:

- All TransactionVision started tasks installed for use require READ ACCESS to the SSLDAUTH dataset.
- All TransactionVision started tasks installed (excluding the TVISION member procedure) require an RACF OMVS segment defined to the authorization ID in use by the started task.This requirement exists solely due to use of Unix System Service calls to obtain IP addresses and host names. Use of UID=0 can simplify this security requirement.
- All TransactionVision started tasks installed (excluding the TVISION member procedure) require RACF WMQ MQCONN (MQ Connect) authorization to the WMQ subsystem owning the WMQ configuration and event queues in use by the applicable z/OS Agent.
- All TransactionVision started tasks installed (excluding the TVISION member procedure) require RACF WMQ READ and UPDATE access to the configuration and event queues in use by the applicable z/OS Agent.

After completing the post install configuration of all desired agent types, proceed to "Post Install WLM DisPatching Priority" on page 162.

Post Install WLM DisPatching Priority

The z/OS Workload Manager service class assigned to TransactionVision started tasks should enable a dispatching priority equal to or higher than the dispatching priority assigned to the CICS region, IMS region, or batch jobs which are the target for TransactionVision agent collection.

This is critical in enabling TransactionVision z/OS agent started tasks the ability to keep up with event workload being generated by the monitored entity.

13

z/OS Components–Operation and Customization

This chapter includes:

Component Overview

TransactionVision z/OS Agents are comprised of common components used by all z/OS agent types and agent-specific components that are unique to the agent type being deployed.

The following sections provide an overview of the three primary components used by the z/OS agent types:

- "TransactionVision Manager" on page 164
- "Agent Managers" on page 164
- "Agent Event Collectors" on page 165

The following discussion does not apply to the z/OS WMQ MQ-IMS Bridge agent. It's implementation is discussed in a subsequent section.

TransactionVision Manager

The TransactionVision Manager (TVM) component manages the execution of agents. It runs as a started task (procedure TVISION) and provides an execution environment that is used by the various z/OS agent types provided.

Agent Managers (discussed in the next section) are started and stopped by issuing z/OS modify commands to the TVM.

TVM does not perform any collection activities. The initialization of TVM can be added to your IPL procedures to automate its startup.

Agent Managers

Agent Managers are started and stopped by modify commands to the TransactionVision Manager (TVM) started task. The start of an Agent Manager results in the automatic creation of as additional address space used in support of collection activity for an agent manager type being started.

The started task procedure used to create the Agent Manager address space is dependent upon the agent manager type being started.

Agent Manager types and the default started task name that is started and stopped by using the TVM start/stop modify commands are as follows:

Agent Manager Type	Default Task Name
CICS	- TVISIONC
MQCICS	- TVISIONQ
MQIMS	- TVISIONM
MQBATCH	- TVISIONM (same as MQIMS)

An Agent Manager communicates with an Agent Event Collector (discussed in the next section) and a TransactionVision Analyzer which resides and executes on a distributed platform.

The Agent Manager receives event data from an Agent Event Collector, performs various processing requirements, and then sends the event data to the TransactionVision Analyzer using z/OS WMQ as the transport mechanism.

An Agent Manager also obtains configuration information (filtering, other runtime options) from the TransactionVision Analyzer using z/OS WMQ and uses such options to control processing flow.

Agent Event Collectors

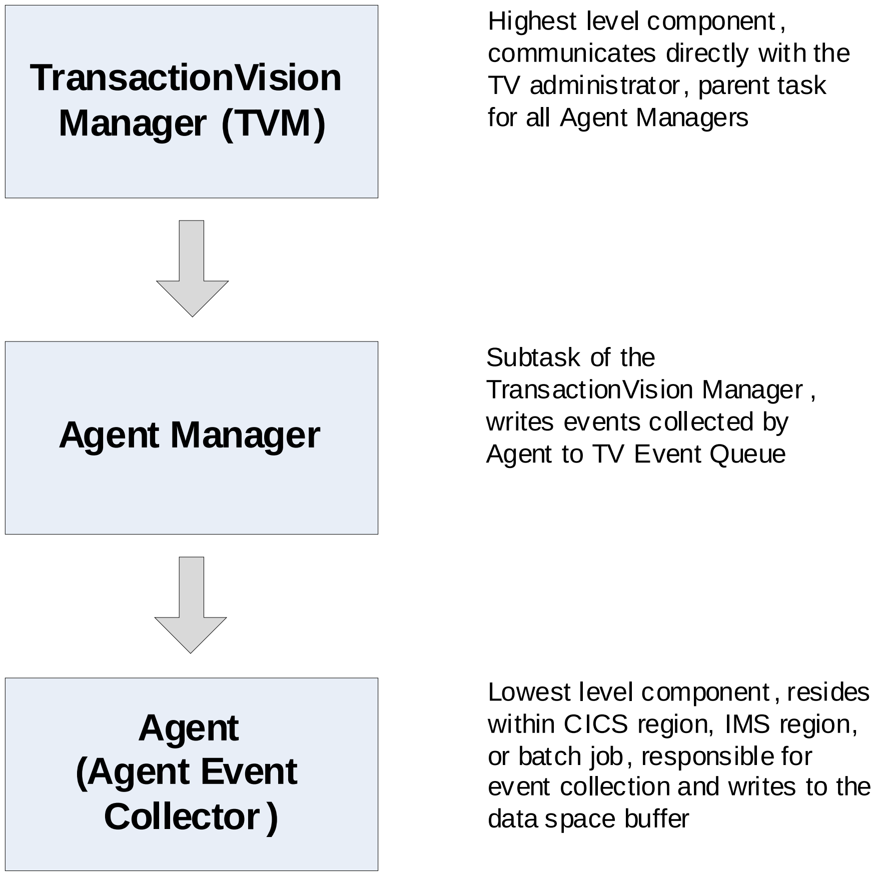
Agent Event Collectors collect the event data for a specific agent type. It is the component that interfaces with application execution to capture the applicable event data for the agent type. Event data is collected, buffered, and made available to an Agent Manager address space for additional processing as discussed in the preceding section.

The implementation of Agent Event Collectors for the various agent types is as follows:

Agent Type	Implementation
CICS Agent	Executes in CICS region and uses CICS services (global exits, task related user exit) to collect required event data. Its status is controlled via CICS transactions and CICS PLTPI processing. For collection to be active a CICS Agent Manager and Agent Event Collector must be started.
MQCICS Agent	Executes in CICS region and uses CICS services (CICS-WMQ crossing exit) to collect required event data. Its status is controlled via CICS transactions and CICS PLTPI processing. For collection to be active a MQCICS Agent Manager and Agent Event Collector must be started.
MQIMS Agent	IMS application must be relinked with an application z/OS WMQ application stub provided by TransactionVision for z/OS. Replacement of the application stub routine is what triggers the ability to capture IMS application WMQ event activity. Status of event collection is controlled by starting and stopping the MQIMS Agent Manager for the applicable IMS id.
MQBATCH Agent	Batch application must be relinked with an application z/OS WMQ application stub provided by TransactionVision for z/OS. Replacement of the application stub routine is what triggers the ability to capture batch job application WMQ event activity. Status of event collection is controlled by starting and stopping the MQBATCH Agent Manager.

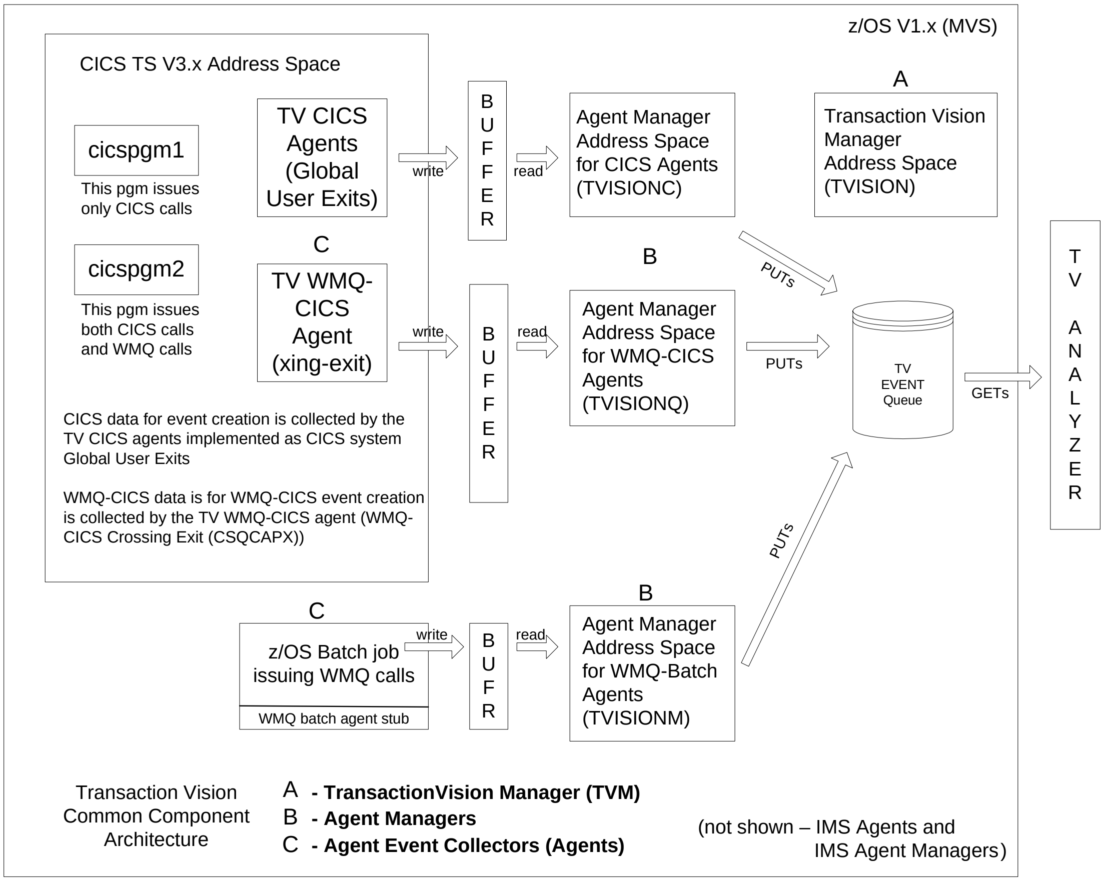
Component Hierarchy

The following diagram shows the component hierarchy.


TransactionVision Manager (TVM)
Highest level component, communicates directly with the TV administrator, parent task for all Agent Managers
Agent Manager
Subtask of the TransactionVision Manager, writes events collected by Agent to TV Event Queue
Agent (Agent Event Collector)
Lowest level component, resides within CICS region, IMS region, or batch job, responsible for event collection and writes to the data space buffer

Architectural Overview

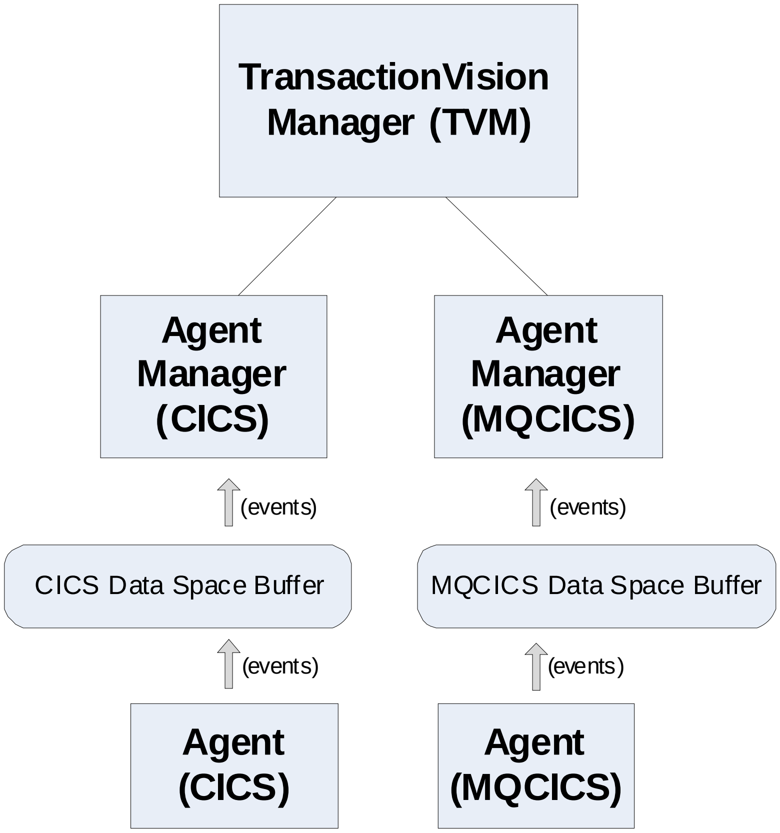
A single CICS or WMQ CICS Agent Manager can service events created by Agents in multiple CICS address spaces. The 'BUFFER' components are actually MVS data spaces. At agent manager startup time, the MAXQBLKS and QBLKSIZE parameters may be used to change the default size of the individual data spaces.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



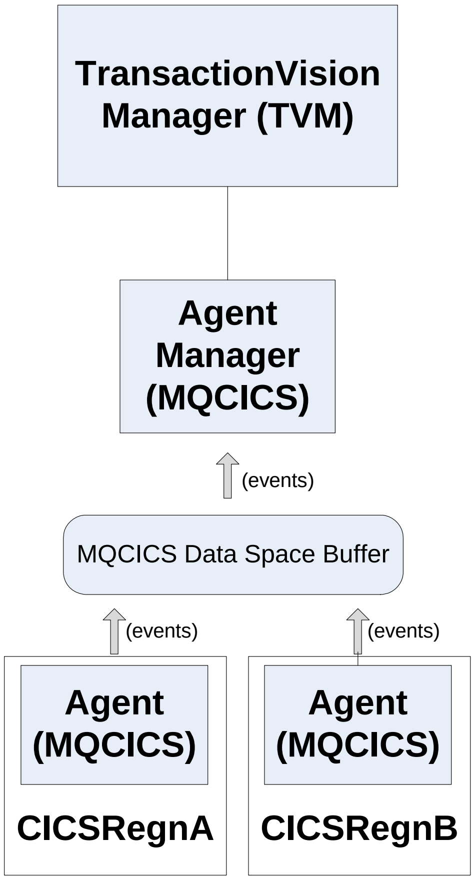
z/OS V1.x (MVS)
CICS TS V3.x Address Space
cicspgm1
This pgm issues only CICS calls
cicspgm2
This pgm issues both CICS calls and WMQ calls
TV CICS Agents (Global User Exits)
C
TV WMQ-CICS Agent (xing-exit)
CICS data for event creation is collected by the TV CICS agents implemented as CICS system Global User Exits
WMQ-CICS data is for WMQ-CICS event creation is collected by the TV WMQ-CICS agent (WMQ-CICS Crossing Exit (CSQCAPX))
write
BUFFER
read
Agent Manager Address Space for CICS Agents (TVISIONC)
A
Transaction Vision Manager Address Space (TVISION)
B
Agent Manager Address Space for WMQ-CICS Agents (TVISIONQ)
PUTs
PUTs
TV EVENT Queue
GETs
TV ANALYZER
PUTs
C
z/OS Batch job issuing WMQ calls
WMQ batch agent stub
write
BUFR
read
B
Agent Manager Address Space for WMQ-Batch Agents (TVISIONM)
Transaction Vision Common Component Architecture
A - TransactionVision Manager (TVM)
B - Agent Managers
C - Agent Event Collectors (Agents)
(not shown – IMS Agents and IMS Agent Managers)

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.



By definition, the MQCICS agents show above reside in different CICS regions

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 173

➤ "Stopping an Agent Manager" on page 175

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. The sysid value specified is used to link an Agent Manager started task with an Agent Event Collector. This link between the sysid values is what informs the Agent Event Collector which Agent Manager should be used to process event data.

When a CICS or MQCICS Agent Event Collector is started (using applicable CICS transactions, or CICS PLTPI) it is also assigned a sysid. The sysid assigned to a CICS or MQCICS Agent Event Collector by default is the CICS SYSID value assigned to the CICS region via the CICS SIT.

If the CICS SYSID used by a CICS region is not unique among all CICS regions to be monitored by TransactionVision for z/OS, or you choose to have multiple CICS or MQCICS Agent Event Collectors use the same Agent Manager instance, see "Using CICS INITPARM" on page 181 for additional information.

The same sysid value can be used for a CICS Agent Manager and a MQCICS Agent Manager. For example if you want to use both CICS and MQCICS agents for a selected CICS region, simply allow the CICS and MQCICS Agent Event Collectors to default to the CICS SYSID value, then issue modify commands to start the CICS and MQCICS Agent Managers using the CICS SYSID value in use by the CICS region.

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 184 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 184 for more information.

Example 1:

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

The MQCICS Agent manager is started using the default started task name - **TVISIONQ**.

Example 2:

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

The CICS Agent Manager is started using the default started task name - **TVISIONC**.

Example 3:

```
F TVISION,START MQBATCH
```

The MQBATCH Agent Manager is started using the default started task name - **TVISIONM**. The MQBATCH Agent Manager does not use the SYSID operand.

Example 4:

```
F TVISION,START MQIMS IMSID(IMS1)
```

The MQIMS Agent Manager is started using the default started task name - **TVISIONM**.

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. The SYSID specified is the SYSID that was used to start the CICS or MQCICS Agent Manager.

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.

Example 2:

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

The CICS Agent Manager is stopped.

Example 3:

```
F TVISION,STOP MQBATCH
```

The MQBATCH Agent Manager is stopped. The MQBATCH Agent Manager does not use the SYSID operand.

Example 4:

```
F TVISION,STOP MQIMS IMSID(IMS1)
```

The MQIMS Agent Manager is stopped.

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 (Agent Event Collectors)

The z/OS CICS Agent and WMQ-CICS Agent require collection to be started within the desired CICS regions for event capture to occur. Collection can be started and stopped manually using CICS transactions or it can be started automatically at CICS startup using CICS PLTPI processing. You can manually start and stop the CICS transactions from a CICS terminal or by using the z/OS **MODIFY** command.

If you use CICS PLTPI to automatically start collection, you can use the appropriate CICS transaction to stop collection if desired. You can subsequently use the appropriate start transaction to restart collection as well.

If CICS is terminated via shutdown all z/OS Agent components will detect shutdown in progress and the z/OS Agent will terminate itself—there is no need to update CICS PLTSD to terminate active components.

The following sections explain how to use the above methods for starting and stopping collection for the CICS and WMQ-CICS Agent event collectors residing within the CICS region:

- "Starting CICS Agent Event Collectors via CICS Transactions" on page 179
- "Stopping CICS Agent Event Collectors via CICS Transactions" on page 179
- "Starting CICS Agent Event Collectors via CICS PLTPI" on page 179
- "Starting MQCICS Agent Event Collectors via CICS Transaction" on page 180
- "Stopping MQCICS Agent Event Collectors via CICS Transaction" on page 180
- "Starting MQCICS Agent Event Collectors via CICS PLTPI" on page 180
- "Enabling MQBATCH and MQIMS Agent Event Collectors" on page 181
- "Using CICS INITPARM" on page 181
- "MQ CICS Agent Event Collector Status Display" on page 183

Starting CICS Agent Event Collectors via CICS Transactions

```
MODIFY cics_jobname, SLDE
```

where

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

SLDE is the CICS transaction-id which starts all CICS Agent Event Collectors

Stopping CICS Agent Event Collectors via CICS Transactions

```
MODIFY cics_jobname, SLDD
```

where

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

SLDD is the CICS transaction-id which stops all CICS Event Collectors

Starting CICS Agent Event Collectors via CICS PLTPI

To automatically start CICS agent collection when CICS is started, add the PLTPI entry to your existing PLTPI as follows. It must be added some where following your existing DFHDELIM entry within your existing PLTPI in use.

```
DFHPLT TYPE=INITIAL,SUFFIX=XX
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
DFHPLT TYPE=ENTRY,PROGRAM=SLDCENA
DFHPLT TYPE=FINAL
END
```

Starting MQCICS Agent Event Collectors via CICS Transaction

```
MODIFY cics_jobname, SLQE
```

where

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

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

Stopping MQCICS Agent Event Collectors via CICS Transaction

```
MODIFY cics_jobname, SLQD
```

where

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

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

Starting MQCICS Agent Event Collectors via CICS PLTPI

To automatically start MQCICS Agent collection when CICS is started add the PLTPI entry to your existing PLTPI as follows. It must be added some where following your existing DFHDELIM entry within your existing PLTPI in use.

```
DFHPLT TYPE=INITIAL,SUFFIX=XX
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
DFHPLT TYPE=ENTRY,PROGRAM=SLDQENA
DFHPLT TYPE=FINAL
END
```

Enabling MQBATCH and MQIMS Agent Event Collectors

The use of the MQBATCH and MQIMS Agents requires that the applicable applications are re-linked with WMQ application stubs provided by TransactionVision for z/OS. The re-link process replaces the WMQ application stubs with the TransactionVision for z/OS WMQ stubs.

The MQBATCH and MQIMS stubs provided are the collection mechanism used to interface with the application in order to capture applicable WMQ event activity. Unlike the CICS and MQCICS Agents there is no starting or stopping of Agent Event Collectors for MQBATCH and MQIMS Agents.

The TransactionVision MQBATCH and MQIMS stubs interface with the applicable Agent Manager. If the application has been re-linked and contains the TransactionVision for z/OS stubs but the applicable Agent Manager is not active, event capture will be bypassed. If the Agent Manager is active event capture will occur. Therefore, the presence of TransactionVision stubs and an active Agent Manager controls the event capture status.

The SSLDINST installation dataset contains members with additional information and sample JCL to assist in relinking your application modules. Refer to member LINKBAT for re-linking MQBATCH applications and member LINKIMS for re-linking MQIMS applications.

Using CICS INITPARM

Use of CICS INITPARM values for CICS and MQCICS Agents enable you to override the SYSID assigned to CICS and MQCICS Agent Event Collectors. CICS INITPARM values can be set in your CICS SIT, which override members in use by your CICS region.

Use of the CICS INITPARM is only required if one of the following is applicable:

➤ CICS SYSID values used by CICS regions in which you intend to use CICS or MQCICS Agents are not unique from a CICS region perspective.

➤ You want to use a single CICS Agent Manager, or a single MQCICS Agent Manager for multiple CICS regions. In other words, you want multiple Agent Manager Collectors to use the same Agent Manager started task. For high volume CICS environments this is not recommended.

When CICS INITPARM values are provided, the CICS and MQCICS Agent Event Collectors will set the SYSID value in use to the CICS INITPARM values. For collection to become fully enabled and functional you must also start the CICS or MQCICS Agent Manager using the SYSID value you specified in the CICS INITPARM.

To set a CICS Agent Event Collector SYSID, add the CICS INITPARM operand as follows:

```
INITPARM=(SLDCMON='XXXX')
```

To set a MQCICS Agent Event Collector SYSID you add the CICS INITPARM operand as follows:

```
INITPARM=(SLDQMON='XXXX')
```

To modify both you would specify:

```
INITPARM=(SLDQMON='XXXX',SLDCMON='XXXX')
```

For example, assume you want to use CICS and MQCICS Agents in three CICS regions and have all three CICS regions use a single CICS Agent Manager and a single MQCICS Agent Manager. To do so you would perform the following steps:

1. Add the following to all three CICS regions:

   ```
   CICS INITPARM=(SLDQMON='AAAA',SLDCMON='AAAA')
   ```

2. Start CICS Agent Manager using:

   ```
   "F TVISION,START CICS SYSID(AAAA)"
   ```

3. Start MQCICS Agent Manager using:

   ```
   "F TVISION,START MQCICS SYSID(AAAA)"
   ```

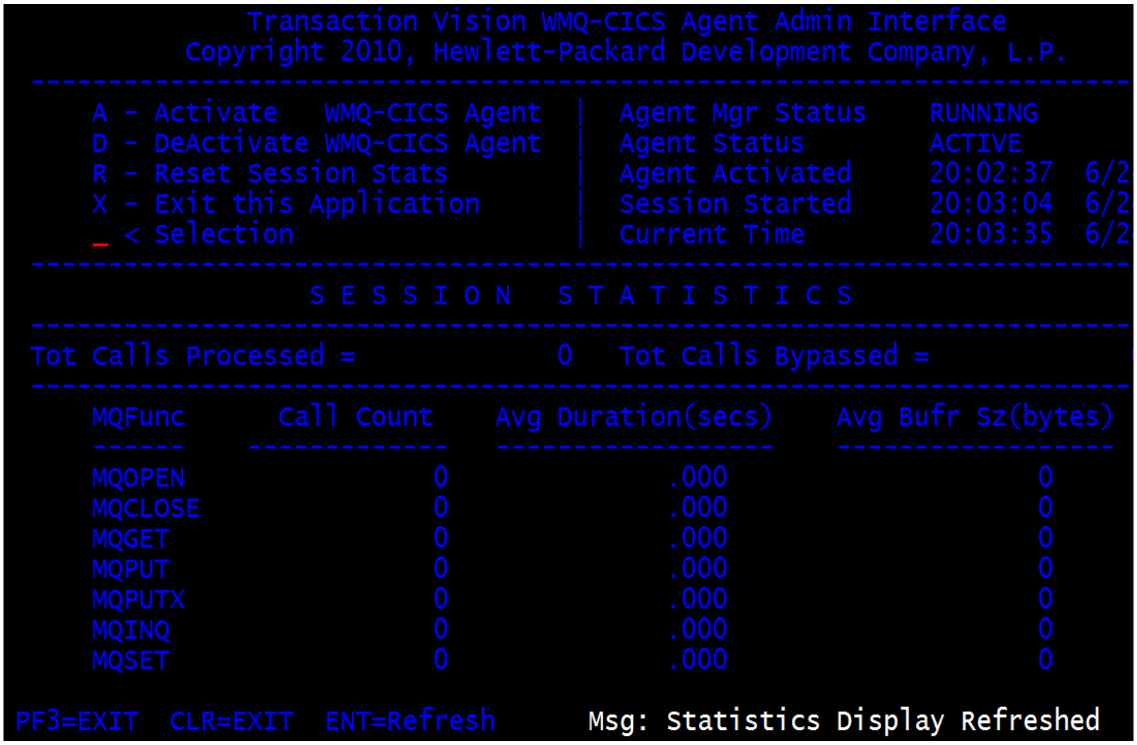
MQ CICS Agent Event Collector Status Display

```
SLQI
```

where

SLQI 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 SLQI 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  6/2
     X - Exit this Application      |  Session Started     20:03:04  6/2
     _ < Selection                  |  Current Time        20:03:35  6/2
 ----------------------------------------------------------------------
                   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
```

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 Agent Data 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.

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.

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 a 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:

- "WebSphere MQ-IMS Bridge Agent Operation" on page 185
- "TVISIONB Buffer Queue" on page 187
- "Event Data" on page 187
- "Data Collection Filters and Queries" on page 190

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 BSM Transaction Management Admin user interface on a connected workstation. For details, see *BSM Application Administration Guide*.

4 Ordinarily, the activity of the Bridge Agent is controlled from the BSM Transaction Management Admin user interface. 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 implicitly disables the agent; the exit routine continues running but does not attempt to send events to the TVISIONB. TVISIOND is automatically 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
```

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
- 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 188
- "IMSBridgeObject.xml" on page 189

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.IMSBridgeEntryModife**r **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" on page 192
- "Enable Data Range Functionality of the Data Collection Filter to Reduce Event Payload Size" on page 192
- "Migrate All TransactionVision WMQ Traffic to a Separate WMQ Queue Manager and Channel Initiator Pair" on page 193

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 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.

14

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. For complete .NET installation and configuration details, see the *HP Diagnostics .NET Agent Guide*. This pdf is available with the agent installation <agent_install_dir>\html\Diagnostics_Dotnet_Agent_Guide.pdf.

This chapter includes:

About .NET Agent Installer

The .NET Agent software is installed on the machine hosting the application you want to monitor. With the .NET Agent you instrument the application domains for monitoring.

The .NET Agent uses points files to provide standard instrumentation to enable you to start monitoring applications. The points files control the workload the agent captures for the application. For information about configuring the points files for TransactionVision, see "Configuring the Points Files" on page 211.

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 203.
  - Default **Asp.Net.points**, **Ado.points** and **WCF.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 files **Asp.Net.points** and **Ado.Points** need to be enabled to generate TransactionVision events.

    To generate .NET Remoting Events you need **Remoting.points** and must setup the application for instrumentation.

    To generate .NET MSMQ Send and Receive Events, you 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*. For details about configuring points files for TransactionVision, see "Configuring the Points Files" on page 211.

- Optional configuration. Modify the agent configuration in the **probe_config.xml** file. See "Configuring the .NET Agent" on page 203.
- Events that TransactionVision can trace with the .NET Agent. For a list of the events, see "About the .NET Agent" on page 92.

For detailed information about installing the .NET Agent for Diagnostics, see the *HP Diagnostics .NET Agent Guide,* this pdf is available with the agent installation <agent_install_dir>\html\Diagnostics_Dotnet_Agent_Guide.pdf

Installing the .NET Agent

The .NET Agent (version 9.2x) requires .NET Framework 2.0 or later. The .NET Framework must be installed on the machine before you run the .NET Agent installation.

Note: If there is a pre-existing installation of the .NET Agent on the host machine, see "Upgrading the .NET Agent" on page 286 for important instructions on how to upgrade the agent systems.

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

This section includes:

- "Preparing for the Installation" on page 196
- "Launching the .NET Agent Installer" on page 197
- "Running the Installation" on page 198

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) which will continue to be supported and updated through patches.

WCF Requirements and Limitations: Monitoring .NET Windows Communication Foundation (WCF) services requires .NET Framework 3.0 SP1 or greater. Only the following bindings are supported:

- BasicHttpBinding
- WSHttpBinding
- NetTcpBinding

If your application uses a binding that is not supported, the .NET probe only creates a generic server request for each WCF method. It will not be a web Service and there will be no XVM correlation.

Launching the .NET Agent Installer

You can install the .NET Agent from the Diagnostics installation disk or copy the executable installation file to another location and run it, or select to install the .NET Agent from the HP Software Download Center.

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

To launch the installer from the product installation product disk or other installation location:

1 Locate the installation package file for your platform:

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

2 Run the installer from the installation product disks.

3 Continue with "Running the Installation" on page 198.

To launch the Installer from the HP Software Download Center:

1 Go to the HP Software web site’s Software Download Center (you will need your HP Passport login).

2 Locate the **TransactionVision** (or Diagnostics) downloads and choose the appropriate link for downloading the .NET Agent software.

3 Follow the download instructions on the web site.

4 Continue with "Running the Installation" on page 198.

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 **End user license agreement**

Accept the end user license agreement.

Read the agreement and select **I accept the terms of the License Agreement**.

Click **Next** to continue.

2 **Specify install location**

Provide the location where you want the Agent installed.

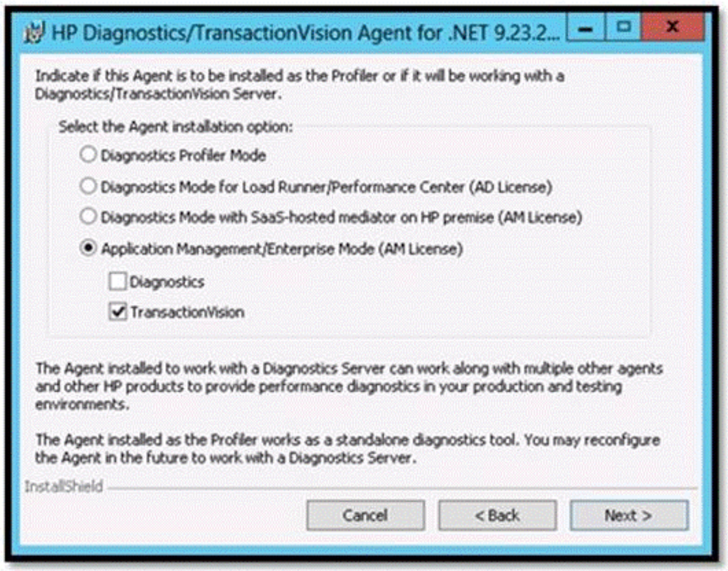
By default, the Agent is installed in **C:\MercuryDiagnostics\.NET Probe**.

Accept the default directory or select a different location either by typing in a different path, or by clicking **Browse** to navigate to the installation directory.

Click **Next** to continue.

3 Select installation options

Select the **Application Management/Enterprise Mode (AM License)** and **TransactionVision** options to install the .NET Agent for use with a TransactionVision Server in an enterprise (or production) environment.


HP Diagnostics/TransactionVision Agent for .NET 9.23.2...

Indicate if this Agent is to be installed as the Profiler or if it will be working with a Diagnostics/TransactionVision Server.

Select the Agent installation option:

- ○ Diagnostics Profiler Mode
- ○ Diagnostics Mode for Load Runner/Performance Center (AD License)
- ○ Diagnostics Mode with SaaS-hosted mediator on HP premise (AM License)
- ● Application Management/Enterprise Mode (AM License)
  - ☐ Diagnostics
  - ☑ TransactionVision

The Agent installed to work with a Diagnostics Server can work along with multiple other agents and other HP products to provide performance diagnostics in your production and testing environments.

The Agent installed as the Profiler works as a standalone diagnostics tool. You may reconfigure the Agent in the future to work with a Diagnostics Server.

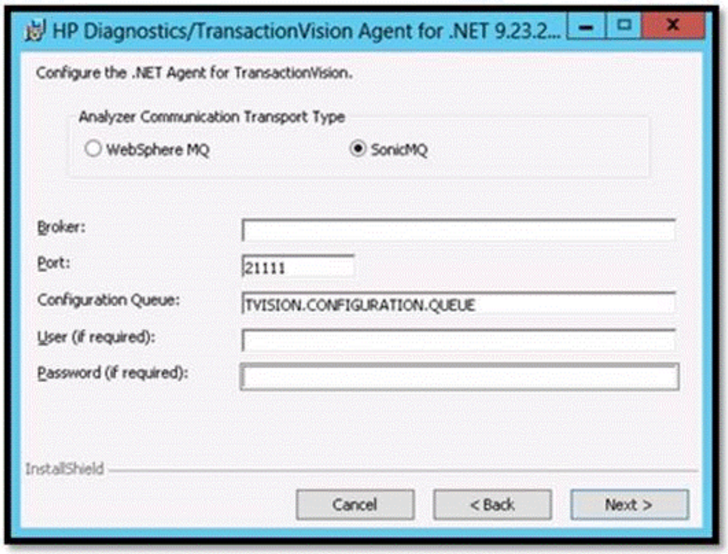
These options set the value of the **probe_config.xml <modes>** element to **tv** mode. The mode value in the **probe_config.xml <modes>** element is also used in determining usage against the license capacity. Agents in **AM** mode are always counted against the AM license capacity.

Click **Next** to continue.

Note: If you are installing the Agent as the Diagnostics Profiler or to work with a Diagnostics Server, see the *HP Diagnostics .NET Agent Guide*.

---

4 Configure the .NET Agent for TransactionVision


HP Diagnostics/TransactionVision Agent for .NET 9.23.2...

Configure the .NET Agent for TransactionVision.

Analyzer Communication Transport Type

○ WebSphere MQ [✓ ● SonicMQ]

Broker:

Port: 21111

Configuration Queue: TVISION.CONFIGURATION.QUEUE

User (if required):

Password (if required):

Choose the Messaging Middleware Provider. Options are: WebSphere MQ and SonicMQ.

SonicMQ is included with the .NET Agent. If you specify this option, the Sonic MQ .NET client (Sonic.Client.dll - Progress SonicMQ .NET Client, version 7.6.0.112) is installed as part of the Agent installation.

A third-party WebSphere MQ installation can be used instead. In this 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.

a 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 "Administration Utilities" in the *BSM Application Administration Guide*.

b 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 "Administration Utilities" in the *BSM Application Administration Guide*.

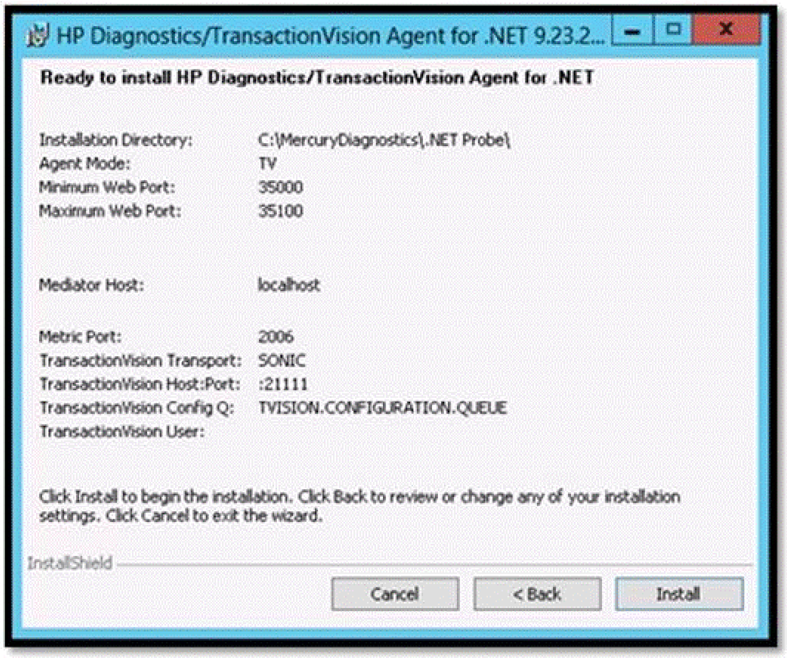
Websphere MQ channel. Channel name for WebSphere MQ queue manager.

Websphere MQ Q Manager. CCSID for WebSphere.

Click **Next** to continue.

5 Pre-install summary

The pre-installation summary dialog opens. Click **Back** to make any changes. Click **Install** to start the .NET Agent installation.



Ready to install HP Diagnostics/TransactionVision Agent for .NET

Installation Directory:	C:\MercuryDiagnostics\.NET Probe\
Agent Mode:	TV
Minimum Web Port:	35000
Maximum Web Port:	35100
Mediator Host:	localhost
Metric Port:	2006
TransactionVision Transport:	SONIC
TransactionVision Host:Port:	:21111
TransactionVision Config Q:	TVISION.CONFIGURATION.QUEUE
TransactionVision User:	

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

After the installation completes, you must restart IIS (see "Determining the Version of the .NET Agent" on page 213). You can perform any custom configuration if desired as described in the section that follows.

6 Post Install Information

On the final installation screen, you can select the **Show the Windows Installer Log** check box to view the log file and check for errors.

Click **Finish** to exit the installer.

7 Restart 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.

Configuring the .NET Agent

The default configuration of the .NET Agent allows you to begin tracing certain .NET events in a monitored application. You can customize the .NET Agent configuration to control what .NET events are traced and sent to the TransactionVision Analyzer.

This section includes the following:

- "Configuring the probe_config.xml File" on page 203
- "Configuring the Points Files" on page 211

Configuring the probe_config.xml File

To override the default configuration, access the **<agent_install_dir>/etc/probe_config.xml** file.

Use the following elements:

- "<tv> element" on page 204
- "<timeskew> element" on page 206

➤ "<transport> element" on page 207

➤ "<gentvhttpeventforwcf> Element" on page 209

➤ "<logging> Element" on page 209

➤ "<modes> Element" on page 210

➤ "<points> Element" on page 210

The <tv>, <timeskew>, <transport>, and <gentvhttpeventforwcf> elements are supported for TransactionVision only. The <logging>, <modes>, and <points> elements are supported by TransactionVision and Diagnostics. The <modes> element is set at installation.

For complete information on the **probe_config** file, see the *HP Diagnostics .NET Agent Guide*, this pdf is available with the agent installation <agent_install_dir>\html\Diagnostics_Dotnet_Agent_Guide.pdf.

<tv> element

Purpose

Configure the .NET Agent for use with TransactionVision.

Attributes

Attributes	Valid Values	Default	Description
eventthreads	number	3	(Read on startup) The number of threads spawned by the Agent to send events to the Analyzer.
eventthreadsleep	number	100	(Dynamic) The time in milliseconds the event thread sleeps after sending a message (event package).

Attributes	**Valid Values**	**Default**	**Description**
eventmemorythreshold	number	250,000,000	(Dynamic) The memory consumed by the internal buffer(Q) after which the .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.
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 "Administration Utilities" in the *BSM Application Administration Guide*.
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"/>
```

<gentvhttpeventforwcf> Element

Purpose

Setting this option enables generation of a TransactionVision event for a WCF service with any binding that uses IIS (http based) hosting. Some WCF services may use a custom or private binding that is not supported as a true web service and in these types of cases TransactionVision web service events would not be generated unless you enable this option.

Attributes

Attributes	Valid Values	Default	Description
enabled	true, false	false	Enables/disables the generation of an http event for a WCF service with any binding that uses IIS (http based) hosting. If enabled, provides TransactionVision web service events.

Elements

hhhh

Number of Occurrences	zero or more
Parent Elements	probeconfig, process, appdomain
Child Elements	None

Example

```
<gentvhttpeventforwcf enabled="true"/>
```

<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 .NET Agent Guide*, this pdf is available with the agent installation <agent_install_dir>\html\Diagnostics_Dotnet_Agent_Guide.pdf.

<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 .NET Agent Guide*, this pdf is available with the agent installation <agent_install_dir>\html\Diagnostics_Dotnet_Agent_Guide.pdf.

<points> Element

Purpose

Specifies the capture points file to use for instrumentation.

Attributes

Attributes	Valid Values	Default	Description
file	string	none	Name of instrumentation capture points file.

Elements

Number of Occurrences	zero or more
Parent Elements	appdomain, process
Child Elements	None

Example

```
<appdomain enabled="false" name="1/ROOT/WCFAccounts" website="Default Web
Site">
    <points file="Default Web Site-WCFAccounts.points" />
    <logging level="" />
</appdomain>
```

Configuring the Points Files

The .NET Agent installer automatically creates a separate capture points file for each IIS deployed **ASP.NET** Application Domain it detects. You must modify the automatically detected and created points file to enable custom instrumentation points for the Application Domain. These capture points files are located in the **<probe_install_dir>\etc\<ApplicationDomain>.points** file. These points files and the default points files are read by the .NET Agent. For complete details about points files, see "Custom Instrumentation for .NET Applications" in the *HP Diagnostics .NET Agent Guide,* this pdf is available with the agent installation <agent_install_dir>\html\Diagnostics_Dotnet_Agent_Guide.pdf

---

Caution: Do not modify any of the default points files because, in an installation upgrade, modifications are lost. Store your application-specific instrumentation points in a custom capture points file.

You enable the points files by adding a reference to them in the **<points>** element in the scope of the appdomain in the **probe_config.xml** file.

The Argument **detail** in the points file contains a value **tv:user_event**, which provides extensive support for TransactionVision tracing by enabling TransactionVision event generation from practically any given method in any .NET application. For more details about the points file arguments and their syntax, see "Coding Points in the Capture Points File" in the *HP Diagnostics .NET Agent Guide.*

tv:user_event generates a TransactionVision event for the methods that match. As part of the TransactionVision event the parameters to the method are collected as the Request Payload and the return value is collected as the Response Payload. The values displayed are the **ToString()** values returned by the parameters or the return value objects. You specify the method on which you want a TransactionVision event generated. It is highly recommended that event generation is specified for one method at a time to avoid too many events and performance degradation in TransactionVision. Avoid using wild card specifications (but they are supported for convenience).

Note: Not all parameters and return values may be collected.

Following is an example of a points file using the **tv:user_event** value:

```
[ILTest]
class       = !ILTest_NameSpace.ILTest_Class
method  = methodWithParams, staticMethodWithParams, ManyParams
detail = tv:user_event
ignoremethod = Main
```

Enabling Correlation of .NET Events

The following .NET correlation rules are available by default on the Transaction Management user interface and can be enabled from the Event Customization Rules page. For details about correlation rules, see "Custom Correlation" in the *BSM Application Administration Guide*.

- .NETMSMQRule
- .NETRemotingRule
- .NETRule
- .NETWCFRule

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. From the Windows Control Panel, select **Add/Remove Programs** and then select **HP Diagnostics/TransactionVision Agent for .NET** to uninstall.
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 255. For more information, see the *HP Business Service Management Hardening Guide* PDF.

15

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

The Proxy Agent can correlate business transactions involving unsensored applications in two different ways.

- 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.
- Both request and reply events from the monitored application share a common key in the event data that can be used to identify the request/reply pair. The XPath to this key must be specified in the proxy definition. If the data exists in different locations of the request/reply events, modifier rules can be used to extract this value into a common location.

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, or based on a common key. The proxy definition file is an XML file that defines the attributes of proxy objects. The proxy XML file can be found in the Analyzer properties configuration in the Transaction Management user interface (select **Admin** > **Transaction Management** > click the **Configuration** tab (left pane) > select <processing_server> > <analyzer> > click the **Configuration** tab (right pane) > **Properties** tab).

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.

This section includes the following:

- ➤ "Example" on page 216
- ➤ "Subelements" on page 217
- ➤ "Optional Attributes for the Proxy Element" on page 218

Example

This example defines a proxy element for the program P3 based on a common key:

```
<ProxySensor>
  <Proxy>
    <Request queue="Q1" queueManager="QM1" xpath="/Event/
CustomCorrelationKey"/>
    <Reply queue="Q2" queueManager="QM2" />
    <Program name="P3" path="/usr/local/bin/P3path" />
    <Host name="P3_host_name" os="SOLARIS" />
  </Proxy>
</ProxySensor>
```

Subelements

Specify the following subelements for each proxy element:

Element	**Required?**	**Attributes**	**Description**
Request	Yes	queue queueManager (XPath optional)	The WebSphere MQ queue and queue manager from which the proxy program gets the event. If XPath is specified, the Proxy Agent will ignore the **msgid/correlid/replyTo** information, and create and correlate the proxy node based on the value found at the XPath location.
Reply	Yes	queue queueManager	The WebSphere MQ queue and queue manager where the proxy program puts the reply event of the same message ID and correlation ID as the request event.
Program	Yes	name path	The name and path of the proxy program.
Host	Yes	name os	The name and operating system of the host where the proxy program runs.
Retrieve	No	queue	Causes the Proxy 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.

Element	Required?	Attributes	Description
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*

16

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, or the z/OS operator console 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.

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" on page 221
- ➤ "Maximum Log File Size" on page 221
- ➤ "Maximum Number of Backup Log Files" on page 222
- ➤ "Changing from Circular to Linear Logging" on page 222

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, or the z/OS operator console 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 *BSM Application Administration Guide*.

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 224
- "UNIX Event Appender" on page 225

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

17

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" on page 229

➤ "BSM Predefined Roles Relevant to Transaction Management" on page 231

➤ "About User Data" on page 232

➤ "Additional Permission Requirements for Some Reports and Topologies" on page 233

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 232. **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 233.

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, and so on. 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 "Processing Servers" in *BSM Application Administration Guide*.

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

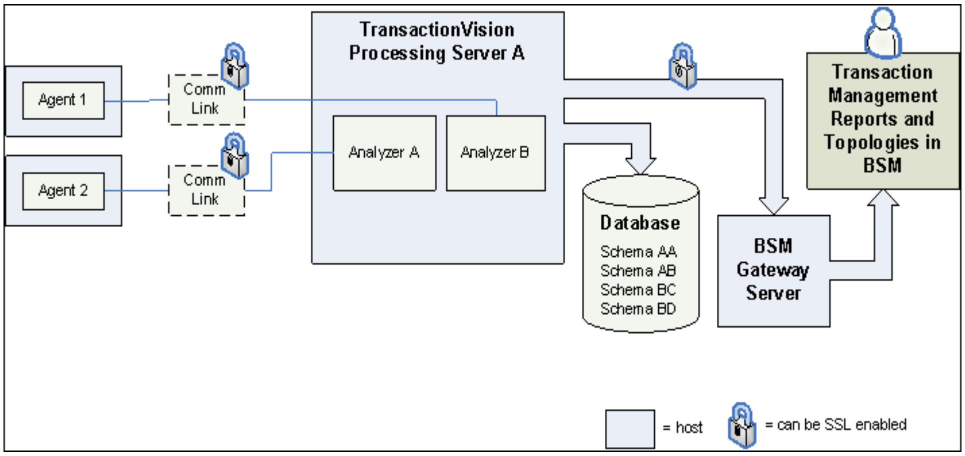
CreateSqlScript -grant -schema SCHEMA

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. The following diagram shows the SSL eligible pathways in an example deployment environment:



The following sections describe how to enable SSL on the TransactionVision Processing Server to the Gateway Server pathway (right side of the diagram), and on the communication link data pathway (left side of the diagram), which is dependent on the type of agent and message middleware provider.

- "Enabling SSL Between a TransactionVision Processing Server and the BSM Gateway Server" on page 238
- "Enabling SSL for the Messaging Middleware Provider" on page 244
- "Enabling SSL for the Agent" on page 253
- "Enabling SSL for the SonicMQ Communication Link" on page 258
- "Enabling SSL for the HTTP and RUM Communication Links" on page 258
- "Enabling SSL Communication for Events Reported to BPI" on page 259
- "Enabling Authentication in TransactionVision SonicMQ" on page 259

Enabling SSL Between a TransactionVision Processing Server and the BSM Gateway Server

To enable SSL between a Processing Server and BSM Gateway Server pathway, perform the tasks that follow:

- "Task 1: Import the Certificate from an SSL Enabled BSM Gateway Server to the TransactionVision Processing Servers" on page 238
- "Task 2: Generate a Certificate" on page 239
- "Task 3: Import the Certificate to the BSM Truststore" on page 240
- "Task 4: Enable SSL on the TransactionVision Processing Servers" on page 240
- "Task 5: Set the BSM Communication Protocol and Port" on page 242
- "Task 6: Synchronize the Processing Server" on page 243

Task 1: Import the Certificate from an SSL Enabled BSM Gateway Server to the TransactionVision Processing Servers

The BSM Gateway Server host must be enabled for SSL. A certificate obtained from the BSM Gateway Server needs to be imported into the cacerts file on each Processing Server host.

The TransactionVision Processing Server cacerts file is located in the following location: **<TVISION_HOME>/jre/lib/security/cacerts**.

Following import of the certificate, the Processing Server components must be restarted.

One way to restart the Processing Server components is to use the nanny utility on the host on which the Processing Server is running:

➤ For Windows run:

```
<TVISION_HOME>\bin\nanny.bat stopAllServices
<TVISION_HOME>\bin\nanny.bat startAllServices
```

➤ For Linux run:

```
<TVISION_HOME>/bin/nanny.sh stopAllServices
<TVISION_HOME>/bin/nanny.sh startAllServices
```

For more information about restarting these components, see "Processing Server" in *BSM Application Administration Guide*.

Task 2: Generate a Certificate

To generate a certificate in the default keystore:

1 On the Processing Server host, generate a certificate with the following command:

```
keytool -genkey -keystore <TVISION_HOME>\jre\lib\security\cacerts -alias tvserverkey
-keyalg RSA
```

Replace <TVISION_HOME> with the absolute path of the TransactionVision Processing Server installation directory. The default installation path on Linux is **/opt/HP/TransactionVision**; on Windows it is **C:\Program Files\HP\TransactionVision**.

TransactionVision requires a JKS keystore type. To import certificates from a PKCS12 keystore into the default TransactionVision keystore, use the following command:

```
keytool -importkeystore -srckeystore C:\mykeystore.p12 -srcstoretype pkcs12
-destkeystore <TVISION_HOME>\jre\lib\security\cacerts
```

The keytool command prompts you for information regarding the creation of the key. Note the following when using this command:

- If you specify a password other than the default **changeit**, be sure to record it as it will be needed to access this key in a later task.
- If you plan to use the keystore with SonicMQ, specify a 1 or 2 character country code. Longer country codes are not supported.
- When keytool prompts for "your first and last name", the fully-qualified Processing Server hostname should be used. For example:

```
What is your first and last name?
[Unknown]: tvhost.my.domain.com
```

2 Export the certificate's public key with the following command:

```
keytool -export -alias tvserverkey -file serverkey.cer -keystore
<TVISION_HOME>\jre\lib\security\cacerts
```

This exports the key to a file called **serverkey.cer**.

Task 3: Import the Certificate to the BSM Truststore

The certificate generated in Task 2: Generate a Certificate, must be incorporated into the BSM truststore.

This task requires the BSM Gateway Server to be restarted.

Task 4: Enable SSL on the TransactionVision Processing Servers

If the deployment environment has multiple Processing Servers, each one must be separately enabled for SSL.

To enable SSL on a Processing Server:

1 Select **Admin** > **Transaction Management** > **Configuration** > **Processing Servers** > <processing server>.

2 Select **Configuration** > **Advanced,** then set the **Enable SSL** check box.

Enter the Keystore Password and Location, and the Key password. These values were provided as result of Task 2: Generate a Certificate.

Note: The keystore location is relative to <TVISION_HOME> unless an absolute path is specified. The default location is <TVISION_HOME>/jre/lib/security/cacerts. Forward slashes can be used regardless of the Processing Server's host operating system.

---

3 Optionally, by default, the SSL port on the Processing Server is used for SSL communication. If you have a port conflict, you can modify the SSL port for the Processing Server. Select **Admin** > **Transaction Management** > **Configuration** > **Processing Servers** > <processing server> > **Configuration** > **General** > **SSL Port** field.

4 Click **Apply**.

When a Processing Server becomes enabled for SSL, any Analyzer, Job Manager or Query Engine running on that Processing Server is also set to run in SSL. By default, the SSL dedicated ports are used for each of them. If you have a port conflict, you can modify the SSL ports.

To modify the SSL port for the Job Manager:

1. Select **Admin** > **Transaction Management** > **Configuration** > **Processing Servers.**
2. On the **Job Manager** tab, locate and select the processing server for which you want to enable the SSL setting.
3. Click the **Edit** button to modify the SSL port as well as any other Job Manager properties.

To modify the SSL port for the Query Engine:

1. Select **Admin** > **Transaction Management** > **Configuration** > **Processing Servers.**
2. On the **Query Engine** tab, locate and select the processing server for which you want to enable the SSL setting.
3. Click the **Edit** button to modify the SSL port as well as any other Query Engine properties.

To modify the SSL port for the Analyzer:

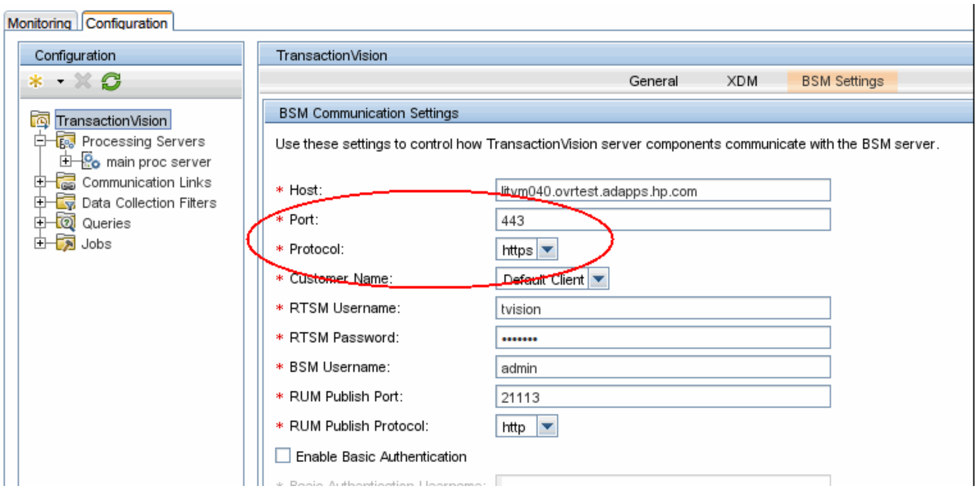
1 Select **Admin** > **Transaction Management** > **Configuration** > **Processing Servers** > <processing server> > <analyzer>.

2 On the **Configuration** > **General** tab, modify the **SSL Port** setting.

Task 5: Set the BSM Communication Protocol and Port

By default, the protocol for the Processing Server to communicate with the BSM Gateway Server is http. To enable SSL, the protocol must be https and the SSL port of 443 must be specified.

To specify these settings, choose **Admin** > **Transaction Management** > **Configuration** > **TransactionVision** (root level node) > **Configuration tab** > **BSM Settings**, and set the **Protocol** to https and **Port** to 443.



Use these settings to control how TransactionVision server components communicate with the BSM server.

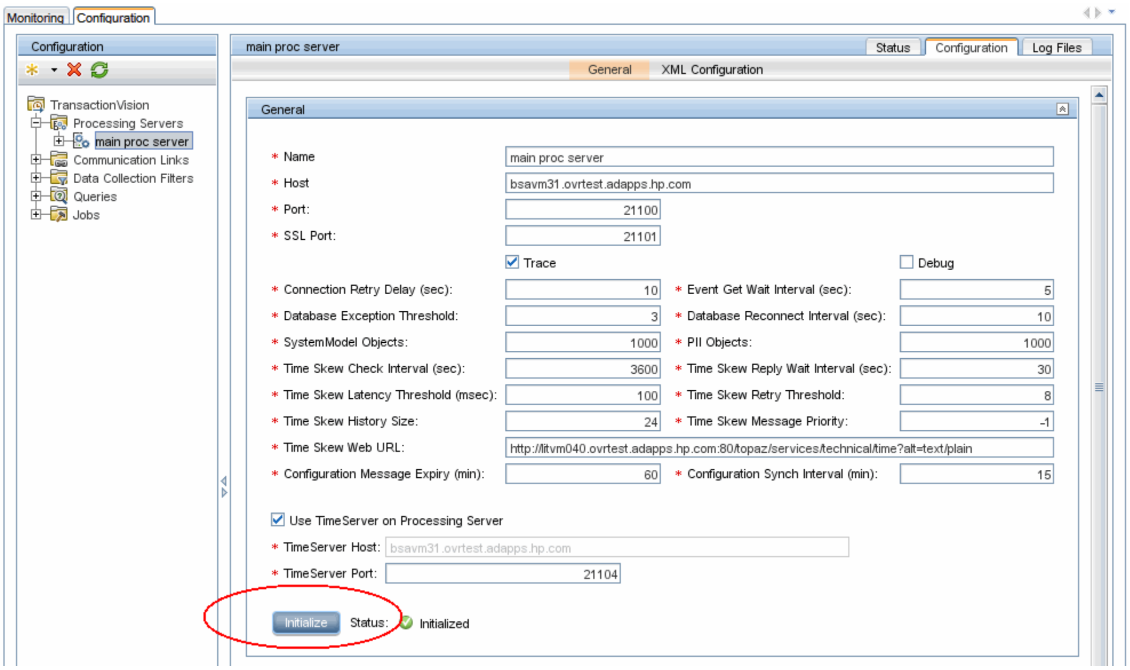
* Host:	litvm040.ovrtest.adapps.hp.com
* Port:	443
* Protocol:	https
* Customer Name:	Default Client
* RTSM Username:	tvision
* RTSM Password:	•••••••
* BSM Username:	admin
* RUM Publish Port:	21113
* RUM Publish Protocol:	http

☐ Enable Basic Authentication

Task 6: Synchronize the Processing Server

To synchronize the Processing Server configuration settings with the changes to the BSM Settings page:

1. Select **Admin** > **Transaction Management** > **Configuration** > **Processing Servers** > <processing server>.
2. On the **Configuration** tab, click the **Initialize** button.



* Name	main proc server		
* Host	bsavm31.ovrtest.adapps.hp.com		
* Port:	21100		
* SSL Port:	21101		
	☑ Trace		☐ Debug
* Connection Retry Delay (sec):	10	* Event Get Wait Interval (sec):	5
* Database Exception Threshold:	3	* Database Reconnect Interval (sec):	10
* SystemModel Objects:	1000	* PII Objects:	1000
* Time Skew Check Interval (sec):	3600	* Time Skew Reply Wait Interval (sec):	30
* Time Skew Latency Threshold (msec):	100	* Time Skew Retry Threshold:	8
* Time Skew History Size:	24	* Time Skew Message Priority:	-1
* Time Skew Web URL:	http://litvm040.ovrtest.adapps.hp.com:80/topaz/services/technical/time?alt=text/plain		
* Configuration Message Expiry (min):	60	* Configuration Synch Interval (min):	15

☑ Use TimeServer on Processing Server

* TimeServer Host: bsavm31.ovrtest.adapps.hp.com
* TimeServer Port: 21104

Status: ✓ Initialized

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.

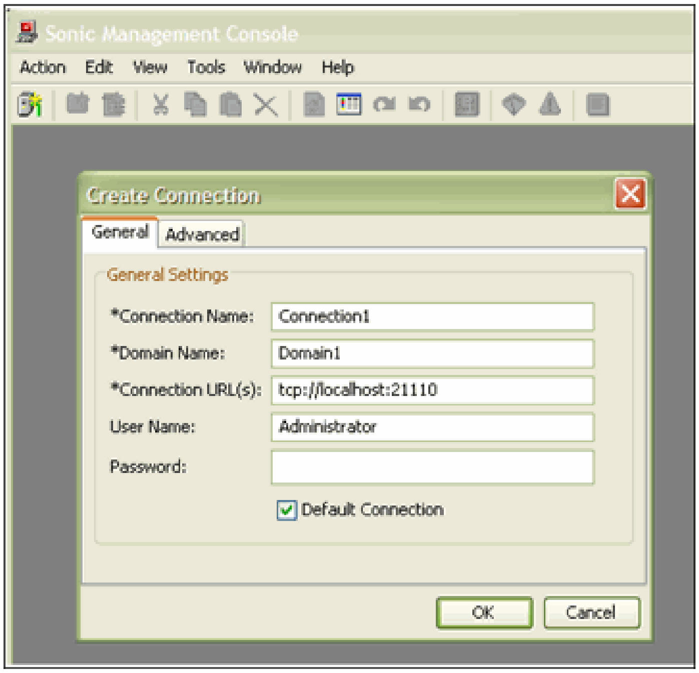
```
keytool -export -alias tvclientcert -keystore client.jks -file CA/tvclientcert.cer
                    Enter keystore password: changeit
                    Certificate stored in file <CA/tvclientcert.cer>
```

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.

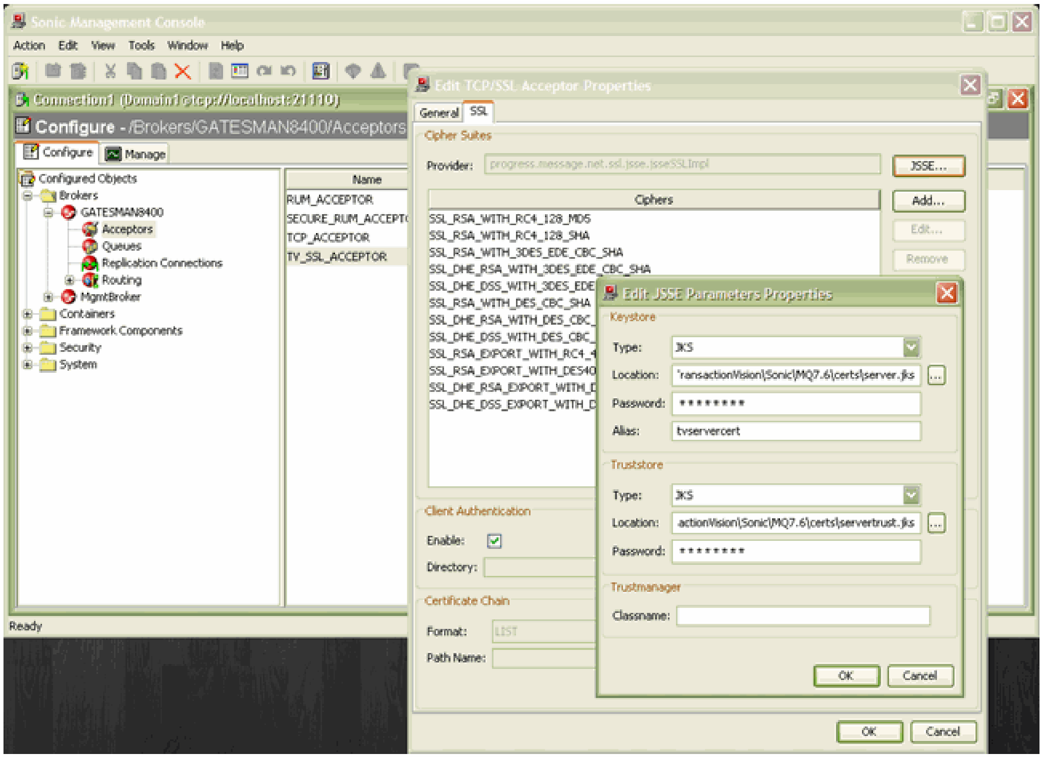


*Connection Name: Connection1
*Domain Name: Domain1
*Connection URL(s): tcp://localhost:21110
User Name: Administrator
Password:
[✓] Default Connection

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.



Provider: progress.message.net.ssl.jsse.jsseSSLImpl

Ciphers:
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE
SSL_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_
SSL_DHE_DSS_WITH_DES_CBC_
SSL_RSA_EXPORT_WITH_RC4_4
SSL_RSA_EXPORT_WITH_DES40
SSL_DHE_RSA_EXPORT_WITH_D
SSL_DHE_DSS_EXPORT_WITH_D

Enable: ✓
Directory:
Format: LIST
Path Name:

Keystore
Type: JKS
Location: 'ransactionVision\Sonic\MQ7.6\certs\server.jks
Password: ********
Alias: tvservercert

Truststore
Type: JKS
Location: actionVision\Sonic\MQ7.6\certs\servertrust.jks
Password: ********

Trustmanager
Classname:

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 244.

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 244. 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 244.

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 253
- "Configure the .NET Agent to Use SSL" on page 255
- "SSL and the WebSphere MQ Agents on Windows or UNIX" on page 257
- "SSL and the WebSphere MQ and CICS Agents on z/OS" on page 258

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 Set the following SSL-related properties:

```
#set the following to true to enable ssl properties. This is required if Sonic SSL transport
is selected
sonicTransport.ssl.prop.enable=true
#path to the clent's key store -- client.jks file
#it can be either absolute path e.g. C:/Sonic/MQ7.6/certs/client.jks
# or relative path from TransactionVisionAgent e.g. config/sensor/client.jks
javax.net.ssl.keyStore=config/sensor/client.jks
#keyStore Password
javax.net.ssl.keyStorePassword=changeit
#Alias name of client's key entry in the key store
sonic.mq.ssl.keyStoreClientAlias=tvclientcert

#path to the Sonic broker's certificate --- clienttrust.jks file
#it can be either absolute path e.g. e.g. C:/Sonic/MQ7.6/certs/clienttrust.jks
# or relative path from TransactionVisionAgent e.g. config/sensor/clienttrust.jks
javax.net.ssl.trustStore=config/sensor/clienttrust.jks
#trust store password
javax.net.ssl.trustStorePassword=changeit
#SSL Provider class name
SSL_PROVIDER_CLASS=progress.message.net.ssl.jsse.jsseSSLImpl
```

c 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 **client.jks** and **clienttrust.jks** files from your processing server installation location to the directory defined by **javax.net.ssl.keyStore** and **javax.net.ssl.trustStore**.

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 .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://brokerhost;
port=21112; user=; password=;
configurationqueue=TVISION.CONFIGURATION.QUEUE"/> Note the broker
has been prefixed with "ssl://".
```

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 244).

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 244).

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

18

Upgrading TransactionVision

This chapter provides instructions for upgrading TransactionVision from versions 8.0x, 9.1x or 9.20 to 9.2x.

Note: Be sure you maintain compatiblity among components when upgrading. See "Compatibility Matrixes" on page 30.

This chapter includes:

Upgrading TransactionVision Overview

There are several upgrade paths for TransactionVision: from 8.0x to 9.2x and from 9.1x or 9.20 to 9.2x. Upgrades from versions of TransactionVision prior to 8.00 are not supported. You must upgrade your TransactionVision system to version 8.0x before proceeding with the upgrade to version 9.2x.

The upgrade procedures and limitations are different for each upgrade path. However, there are some common tasks that need to be completed as well.

- For common upgrade tasks, see "Common Tasks to Complete Before Starting the TransactionVision Upgrade" on page 270.
- For information you need to know before upgrading from 8.0x to 9.2x, see "Notes and Limitations for the 8.0x to 9.2x Upgrade Path" on page 266. For upgrade procedures, see "Upgrading TransactionVision From 8.0x to 9.2x" on page 272.
- For information you need to know before upgrading from 9.1x to 9.2x, see "Notes and Limitations for the 9.1x or 9.20 to 9.2x Upgrade Path" on page 269. For upgrade procedures, see "Upgrading TransactionVision From 9.1x or 9.20 to 9.2x" on page 278.

Note: Be sure you maintain compatiblity between components when upgrading. See "Compatibility Matrixes" on page 30.

See the BSM Release Notes for last-minute information regarding upgrading TransactionVision.

BSM-Side and TransactionVision Processing Server-Side Upgrades

Upgrading TransactionVision consists of two parts: BSM-side upgrade and TransactionVision Processing Server-side upgrade.

The BSM-side upgrade contains the following TransactionVision components, which are upgraded with the BSM upgrade to 9.2x. For BSM upgrading instructions, see the *BSM Upgrade Guide*.

- BSM Management database upgrade, which contains the customer's Processing Server, Analyzer, Communication Links, Filters, Jobs, definitions, and so on, set up by the TransactionVision Administrator.
- Key performance indicators (KPIs) and health indicators (HIs) upgrade, which are the customer's infrastructure configuration items (CIs) collected by TransactionVision and contained in the Run-Time Service Model (RTSM) repository.
- TransactionVision Profile database upgrade, which contains the Business Transaction CI samples, collected and built by TransactionVision.

The TransactionVision Processing Server-side upgrade contains the following components for the TransactionVision upgrade to 9.2x:

- TransactionVision Processing Server
  - For the 8.0x to 9.2x upgrade, the 8.0 Analyzer and UI/Job Server need to be uninstalled and the 9.2x Processing Server needs to be installed.
  - For the 9.1x or 9.20 to 9.2x upgrade, the 9.2x Processing Server needs to be installed over the previous Processing Server with the reinstall/ upgrade option.
  - TransactionVision database upgrade

Upgrading TransactionVision After a BSM Direct Upgrade

TransactionVision can only be upgraded after the BSM direct upgrade to 9.2x is completed and BSM is operating in production mode. Upgrading directly refers to installing the new version on the same servers and database schemas as the original version. TransactionVision does not support a staging upgrade method.

- The TransactionVision system must be shut down before BSM is upgraded by either upgrade method. For more information, see "Common Tasks to Complete Before Starting the TransactionVision Upgrade" on page 270.
- For details on upgrading BSM for either upgrade method, see "Staging vs. Direct Upgrade" in the *BSM Upgrade Guide*.
- For instructions on upgrading TransactionVision, see "Upgrading TransactionVision From 8.0x to 9.2x" on page 272 or "Upgrading TransactionVision From 9.1x or 9.20 to 9.2x" on page 278.

Upgrading Agents

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 280
  - "Upgrading the Java Agent" on page 282
  - "Upgrading the .NET Agent" on page 286
- 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."

Notes and Limitations

The following sections provide information you need to know before upgrading from 8.0x to 9.2x or from 9.1x or 9.20 to 9.2x.

Notes and Limitations for the 8.0x to 9.2x Upgrade Path

Before you start the upgrade process, you need to be aware of the following:

- TransactionVision Analyzers from release 8.0x can be migrated to release 9.2x.
- Transaction classification definitions that had been created for the 8.x TransactionVision deployment can be migrated to 9.2x. This involves exporting the classification definitions from the current 8.x install into a file, and then importing this file into the new 9.2x install.

  To migrate the transaction classification definitions:

  a In the current 8.x install, go to any Analyzer machine and run the following command in the **<TVHOME>/bin** directory:

  ```
  DataUtil.bat|sh -export_txndef
  ```

  This creates a **TransactionDefinition.xml** file in the current directory.

Note: You must perform the rule export before the Processing Server machine is upgraded in the working 8.x environment.

---

b After TransactionVision 9.2x has been fully installed, follow the steps in "Post 8.0x to 9.2x Upgrade Tasks" on page 276 to import all class definitions and create the corresponding transaction CIs in RTSM.

➤ Event modification rules for the **RuleBasedEventModifierBean** in 8.x can be migrated to 9.2x. This involves backing up the 8.x **EventModifierRules.xml** file from **<TVHOME>/config/services** and then importing this file into the new 9.2x install.

Note: The old rule file must be copied/created before the Processing Server machine has been upgraded in the working 8.x environment.

After TransactionVision 9.2x has been fully installed and the user interface is fully operational, follow the steps in "Post 8.0x to 9.2x Upgrade Tasks" on page 276 to migrate the **EventModifierRules.xml** file.

Note: The **Oneof** and **XPathExp** rules are no longer supported in 9.2x. A **Oneof** rule will be converted into a single modifier rule during the import, with a separate modifier for each expression. There is currently no equivalent functionality for **XPathExp** rules in 9.2x, so any such rules will be ignored when importing.

It is still possible to use this functionality by configuring the Analyzer to use the 8.x legacy modifier bean. Contact HP support for details.

➤ Event correlation rules for the **XMLRuleCorrelationBean** in 8.x can be migrated to 9.2x. This involves backing up the 8.x **EventCorrelationDefinition.xml** file from **<TVHOME>/config/services** and then importing this file into the new 9.2x install.

Note: The old rule file must be copied/created before the Processing Server machine has been upgraded in the working 8.x environment.

After TransactionVision 9.2x has been fully installed and the user interface is fully operational, follow the steps in "Post 8.0x to 9.2x Upgrade Tasks" on page 276 to migrate the **EventCorrelationDefinition.xml** file.

➤ Migration of Favorites Filters from 8.0x to 9.2x for Transaction Management reports is not supported. You need to manually recreate the Favorites Filters for Transaction Management reports in 9.2x.

➤ TransactionVision 8.0x portlets in MyBSM are not migrated. MyBSM 9.2x contains newly created Transaction Management components. For more information about the Transaction Management components, see "MyBSM" in *BSM User Guide*.

➤ When you install the TransactionVision 9.2x Processing Server, the 8.0x Analyzer is automatically uninstalled.

➤ TransactionVision UI/Job Servers do not exist in 9.x and do not have configuration settings that apply to 9.x Processing Servers. Therefore, the UI/Job Server does not need to be migrated and needs to be removed from the upgraded environment.

➤ Agents do not contain configurations that can be migrated. Some agents can be upgraded individually and some need to be uninstalled and then installed to the 9.2x version. For details, see Part III, "Agent Installation and Configuration."

- Projects are not migrated — only the event and transaction data stored in the database is migrated. Projects are not supported in 9.2x.
- TransactionVision 9.2x requires transactions to be contained in applications. After the upgrade, Business Transaction CIs are not visible from the **Business Transaction** view. You need to create containment relationships between the transactions and an application. For more information, see "Post 8.0x to 9.2x Upgrade Tasks" on page 276.

Notes and Limitations for the 9.1x or 9.20 to 9.2x Upgrade Path

Before you start the upgrade process, you need to be aware of the following:

- Agents do not contain configurations that can be migrated. Some agents can be upgraded individually (see "Upgrading Agents" on page 266) and some need to be uninstalled and then installed to the 9.2x version. For details, see "Agent Installation and Configuration" on page 86.

Enabling TransactionVision in BSM 9.2x After Upgrading

If TransactionVision was not enabled before upgrading, you need to enable TransactionVision through the BSM License Management and Server Deployment page to be able to access the specified functions in the Transaction Management application from both the Applications and Admin menus. Otherwise, the Transaction Management user interface cannot be accessed.

To access the BSM License Management page, select **Admin** > **Platform** > **Setup and Maintenance** > **License Management**. For details see "License Management Page" in *Platform Administration*.

To access the BSM Server Deployment page, select **Admin** > **Platform** > **Setup and Maintenance** > **Server Deployment**. For details see "Server Deployment Page" in *Platform Administration*.

Common Tasks to Complete Before Starting the TransactionVision Upgrade

Before starting the TransactionVision upgrade (for any upgrade path), make sure that you have completed the following common tasks as appropriate:

- "Check Versions of Prerequisite Software" on page 270
- "Stop the event collection on all analyzers" on page 271
- "Back Up Your System" on page 271
- "Shut down all TransactionVision components" on page 271
- "Upgrade BSM to 9.2x" on page 272

1 Check Versions of Prerequisite Software

The version of TransactionVision you are upgrading to 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 9.2x Processing Server in "Compatibility Matrixes" on page 30.

2 Stop the event collection on all analyzers

Below are the steps for stopping the event collection on all analyzers for TransactionVision 9.x. For TransactionVision 8.0, please refer to the TransactionVision User's Guide.

a) Select **Admin** > **Transaction Management**.

b) Click the **Configuration** tab in the left pane.

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'.
```

3 Back Up 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 the *HP Business Service Manager Database Guide* PDF.
- For your TransactionVision system:
  - Create a copy of the files in **C:\Program Files\HP\TransactionVision**.
  - Copy your TransactionVision database.

4 Shut down all TransactionVision components

Shut down all TransactionVision components by executing the following command on each Processing Server (or Analyzer) machine.

- On Windows use the Windows Service or run: `<TVISION_HOME>\bin\SupervisorStop.bat`
- On Linux run: `<TVISION_HOME>/bin/run_topaz stop`

5 Upgrade BSM to 9.2x

You can only upgrade TransactionVision to a 9.2x version after BSM has been upgraded to that same version. For details, see the *BSM Upgrade Guide*.

You can now continue and upgrade your TransactionVision components as described in "Upgrading TransactionVision From 8.0x to 9.2x" on page 272 or "Upgrading TransactionVision From 9.1x or 9.20 to 9.2x" on page 278.

Note: For last-minute information regarding upgrading or reinstalling TransactionVision, see the TransactionVision Release Notes.

Upgrading TransactionVision From 8.0x to 9.2x

Complete the following steps to upgrade TransactionVision 8.0x to 9.2x only after the BSM upgrade to the same 9.2x version is completed:

1 Before you begin be sure to do the following:

- **a** Read "Notes and Limitations for the 8.0x to 9.2x Upgrade Path" on page 266.
- **b** Complete the tasks described in "Common Tasks to Complete Before Starting the TransactionVision Upgrade" on page 270.

2 Uninstall the Analyzer and UI/Job Server:

On Windows, use the **Add/Remove Programs** utility.

On Linux, do the following:

- **a** Log in as superuser, enter **su**.
- **b** Enter the following command:

```
./tvinstall_<version>_unix.sh -u
```

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

```
The following TransactionVision packages are available for installation:
1. TransactionVision Analyzer
2. TransactionVision UI/Job Server
3. TransactionVision WebSphere MQ Agent


99. All of above
q. Quit install

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

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

During the uninstall of the Analyzer, all TransactionVision processes are automatically stopped.

c Enter **99** and press **Enter** to uninstall all TransactionVision components.

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

d Type **q** and click **Enter** to quit the installation.

3 Install the TransactionVision 9.2x Processing Server as follows:

Run the appropriate TransactionVision Processing Server installer from the HP TransactionVision product installation disk.

On Windows: **HPTVProcServer_<version>_win.exe**

On Linux: **HPTVProcServer_<version>_linux.tgz**

Note: 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.

For details about installing the Processing Server, see "Installing the Processing Server on Windows" on page 48 or "Installing the Processing Server on Linux" on page 50.

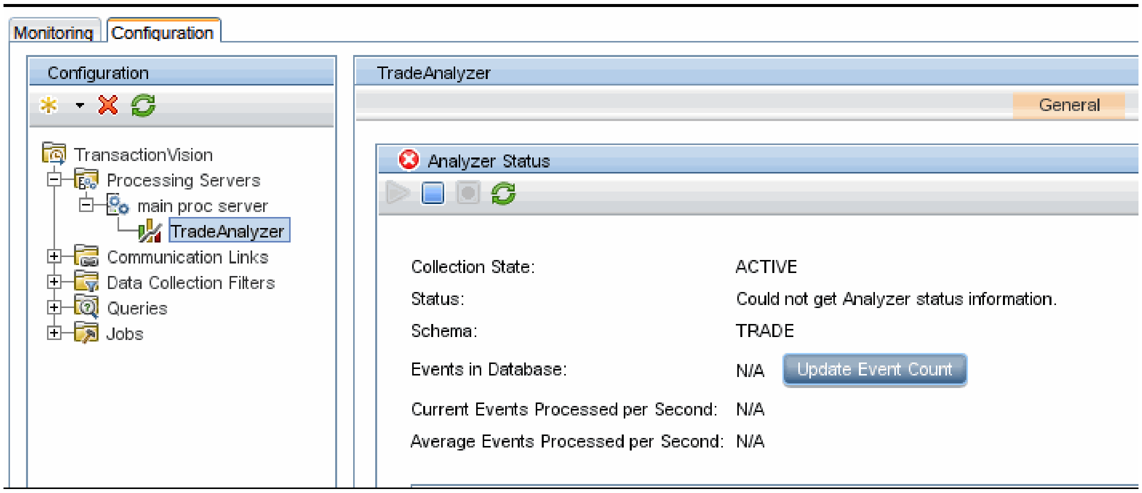
4. Start the Processing Server.
   - On Windows run: <TVISION_HOME>\bin\SupervisorStart.bat
   - On Linux run: <TVISION_HOME>/bin/run_topaz start
5. In the BSM 9.2x user interface add and define a Processing Server. For details see "Creating a Processing Server" under Transaction Management in the *BSM Application Administration Guide*.
6. Make sure the Processing Server Status on the Processing Server page in the user interface displays a green check mark ✓ (select **Admin** > **Transaction Management** > **Configuration tab** > in the left pane, select the Processing Server and in the right pane, look at the Processing Server name bar in the **Status** tab).
7. Be sure you have backed up your databases before you run MigrateDB. Run **MigrateDB** on the Processing Server to migrate all 8.0x project schemas to 9.2x.
   - On Windows run: <TVISION_HOME>\bin\MigrateDB.bat
   - On Linux run: <TVISION_HOME>/bin/MigrateDB.sh

   After the migration is complete, **MigrateDB** creates a file called **ClassIdMapping.xml** in the current directory which will be needed later for the 8.x classification migration.

8 Add and define an Analyzer for each 8.0x project on the 9.2x Processing Server, using the migrated schema of the project. For details, see "Analyzers" in *BSM Application Administration Guide*.

For each migrated schema, add a corresponding Analyzer on the 9.2x Processing Server. A Processing Server can have up to five Analyzers. If a new Processing Server is needed to host the Analyzer, see "How to Create a Processing Server" in *BSM Application Administration Guide*.

For example, in the following figure, an Analyzer named **TradeAnalyzer** was created for the TRADE schema:



Collection State: ACTIVE
Status: Could not get Analyzer status information.
Schema: TRADE
Events in Database: N/A
Current Events Processed per Second: N/A
Average Events Processed per Second: N/A

Caution: If you have more than one project, be aware that you may get duplicated transactions if multiple projects are using the same communication link, such as the default SonicMQ communication link.

The upgrade from TransactionVision 8.0x to 9.2x is complete.

Post 8.0x to 9.2x Upgrade Tasks

1 After the upgrade, Business Transaction CIs are not visible from the **Business Transaction** view. Manually create application CIs and containment relationships for the Business Transaction CIs in RTSM by selecting **Admin** > **RTSM Administration** > **Modeling** > **IT Universe Manager**.

2 Migrate 8.x transaction classification definitions in the BSM 9.2x user interface as follows:

Note: When migrating existing 8.x classification definitions, do not define any new transaction classes until the migration is finished.

---

a Be sure you ran the DataUtil.bat|sh -export_txndef command as described in "Notes and Limitations for the 8.0x to 9.2x Upgrade Path" on page 266. This creates a **TransactionDefinition.xml** file.

b Import the **TransactionDefinition.xml** file.

Select **Admin** > **Transaction Management** > **Monitoring tab** > **Transaction Tracing Rules** (bottom-left pane), click the **Import Transaction Rule Definition XML from a file** button, and select the **TransactionDefinition.xml** file.

This imports all class definitions and creates the corresponding transaction CIs in RTSM.

c Import the **ClassIdMapping.xml** file that was generated by **MigrateDB** in step 8 in "Upgrading TransactionVision From 8.0x to 9.2x" on page 272.

Select **Admin** > **Transaction Management** > **Monitoring tab** > **Transaction Tracing Rules** (bottom-left pane), click the **Import Transaction Rule Definition XML from a file** button, and select the **ClassIdMapping.xml** file.

This ensures that the transaction classes keep their original class IDs they had in 8.x.

d Assign the classification rules to the appropriate Analyzers. For instructions, see "How to Assign a Rule to the Appropriate Analyzers" in *BSM Application Administration Guide*.

3 Migrate 8.x event correlation and modification rules as follows:

a Import the existing **EventModifierRules.xml** rule file.

Select **Admin** > **Transaction Management** > **Monitoring tab** > **Event Customization Rules** (bottom-left pane), click **Import Event Customization Definitions XML from a file** button, and select the **EventModifierRules.xml** rule file.

This imports all modification rules.

b Import the existing **EventCorrelationDefinition.xml** rule file.

Select **Admin** > **Transaction Management** > **Monitoring tab** > **Event Customization Rules** (bottom-left pane), click **Import Event Customization Definitions XML from a file** button, and select the **EventCorrelationDefinition.xml** rule file.

This imports all correlation rules.

c Assign the correlation and modification rules to the appropriate Analyzers. For instructions, see "How to Assign a Rule to the Appropriate Analyzers" in *BSM Application Administration Guide*.

4 You can now upgrade the agents to version 9.2x in the following sections:

➤ "Upgrading the WebSphere MQ Agent on Windows" on page 280

➤ "Upgrading the Java Agent" on page 282

➤ "Upgrading the .NET Agent" on page 286

Upgrading TransactionVision From 9.1x or 9.20 to 9.2x

Complete the following steps to upgrade TransactionVision 9.1x or 9.20 to 9.2x after the BSM upgrade to the same target version as TransactionVision is completed:

1 Before you begin be sure to do the following:

- **a** Read "Notes and Limitations for the 9.1x or 9.20 to 9.2x Upgrade Path" on page 269.
- **b** Complete all tasks described in "Common Tasks to Complete Before Starting the TransactionVision Upgrade" on page 270.

2 Stop the Processing Server:

- ➤ On Windows use the Windows Service or run <TVISION_HOME>\bin\SupervisorStop.bat.
- ➤ On Linux run <TVISION_HOME>/bin/run_topaz stop.

3 Install the TransactionVision 9.2x Processing Server over the 9.1x Processing Server as follows:

Run the appropriate TransactionVision Processing Server installer from the HP TransactionVision product installation disk.

On Windows: **HPTVProcServer_<version>_win.exe**

On Linux: **HPTVProcServer_<version>_linux.tgz**

Note: 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.

For details about installing the Processing Server, see "Installing the Processing Server on Windows" on page 48 or "Installing the Processing Server on Linux" on page 50.

4 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

5 In the BSM 9.2x user interface, initialize the Processing Server:

- **a** Select **Admin** > **Transaction Management**.
- **b** Click the **Configuration** tab in the left pane.
- **c** Expand Processing Servers and select the Processing Server to initialize.
- **d** Click the **Initialize** button.

6 The upgrade from TransactionVision 9.1x or 9.20 to 9.2x is complete.

7 Start the event collection on the Analyzers by clicking the **Start Analyzer** ▶ icon on each Analyzer's Status page.

For more details, see "Analyzers" in *BSM Application Administration Guide*.

Post 9.1x or 9.20 to 9.2x Upgrade Tasks

You can now upgrade the agents to version 9.2x in the following sections:

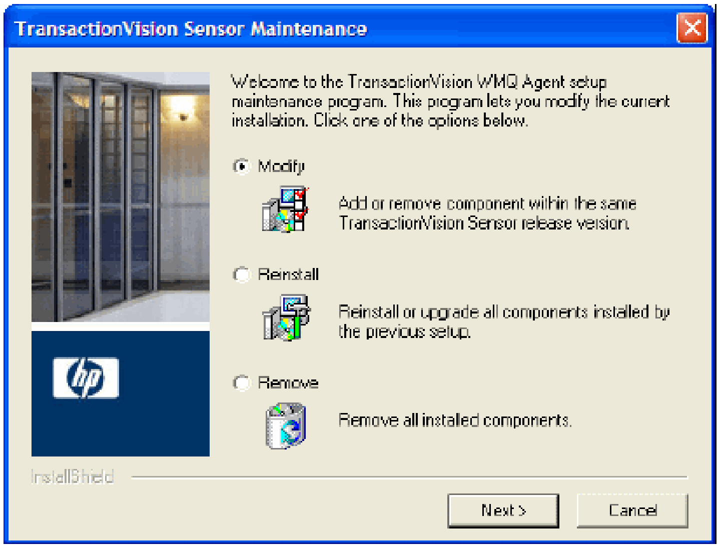
- ➤ "Upgrading the WebSphere MQ Agent on Windows" on page 280
- ➤ "Upgrading the Java Agent" on page 282
- ➤ "Upgrading the .NET Agent" on page 286

Upgrading the WebSphere MQ Agent on Windows

This section contains instructions for upgrading your TransactionVision WebSphere MQ Agent on Windows. For detailed information about the WebSphere MQ Agent, see Chapter 10, "Installing and Configuring the WebSphere MQ Agent."

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:


TransactionVision Sensor Maintenance

Welcome to the TransactionVision WMQ Agent setup maintenance program. This program lets you modify the current installation. Click one of the options below.

[✓ Modify]

Add or remove component within the same TransactionVision Sensor release version.

Reinstall

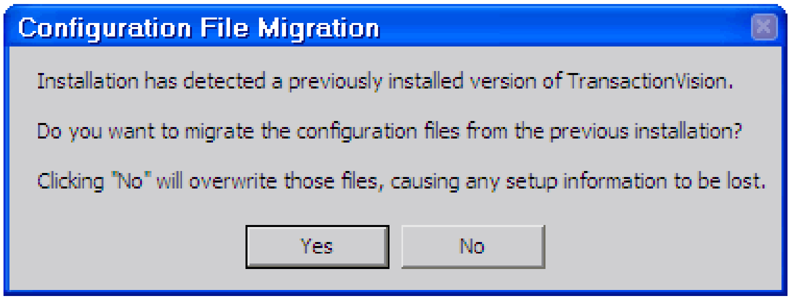
Reinstall or upgrade all components installed by the previous setup.

Remove

Remove all installed components.

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:


Configuration File Migration

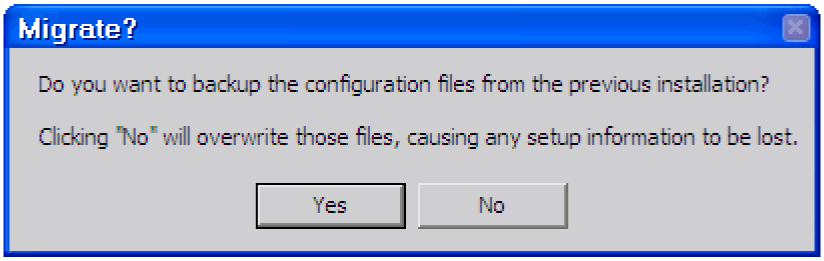
Installation has detected a previously installed version of TransactionVision.

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

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

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.


Migrate?

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

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

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.

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.

Caution: With each new release of Diagnostics you should re-record the Java agent silent install response files prior to performing silent installation on multiple machines.

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, for TransactionVision be sure to do the following. This ensures that the persisted data for your application will match up with the metrics captured by the new agent:

- 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.

2 The installer creates a **<probe_install_dir>\etc** directory in the new installation directory.

Note: If you want to perform a 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 "Silent Installation of the Java Agent (Optional)" on page 146.

---

3 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\Sna
pins>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/TransactionVision 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

Property File	**Configuration Properties To Be Copied to the New Diagnostics/TransactionVision Agent**
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.
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.

4 Prepare your application servers to be monitored using the JRE instrumentation methods described in the *HP Diagnostics Java Agent Guide* chapter on "Preparing Application Servers for Monitoring with the Java Agent." This pdf is available with the agent installation <agent_install_dir>\DiagnosticsAgent\docs\Diagnostics_Java_Agent_Guide.pdf. In particular you need to update the application's startup script or JVM parameters to point to the upgraded agent installation.

5 At an approved time, shut down the applications that were being monitored by the old agent.

6 Restart the applications to allow the new version of the agent to begin monitoring your applications.

7 Clear your browser's cache and restart the browser before you attempt to access the Diagnostics Java Profiler user interface. Failure to do this may result in a size mismatch error message.

8 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.

9 If you configured the Java Agent to work with a Diagnostics Server, you can verify that the upgraded Java Agent is running by checking the version in the System Health view in the Diagnostics UI. The version should be the latest version if the upgrade was successful. To access the System Health view you must open the Diagnostics UI as the Mercury System customer from http://<Diagnostics_Commanding_Server_Name>:2006/query/. Then in the Views pane you can select the System Views view group.

10 When all your applications have been migrated over to the latest version 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.

Note: You must upgrade the Diagnostics Server and/or TransactionVision Processing Server before upgrading the agents that are connected to the servers because servers are typically incompatible with newer versions of the agents.

To upgrade a .NET Agent:

1 Install the new Diagnostics/TransactionVision Agent for .NET (select Upgrade).

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 Agent.

b Restart IIS.

  c Restart the applications that were being monitored by the old .NET Agent.

3 You can verify that the upgraded Diagnostics Agent is running by checking the version in the System Health view in the Diagnostics UI. The version should be the latest version if the upgrade was successful. To access the System Health view you must open the Diagnostics UI as the Mercury System customer from http://<Diagnostics_Commanding_Server_Name>:2006/query/. Then in the Views pane you can select the System Views view group.

Index