

HP Business Service Management

for the Windows operating system

Software Version: 9.00

TransactionVision Deployment

Document Release Date: July 2010

Software Release Date: July 2010



Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

© Copyright 2000 - 2010 Hewlett-Packard Development Company, L.P.

Trademark Notices

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

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

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

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

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

Java™ is a US trademark of Sun Microsystems, Inc.

Microsoft®, Windows®, Windows NT®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

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

UNIX® is a registered trademark of The Open Group.

Acknowledgements

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

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

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

Documentation Updates

The title page of this document contains the following identifying information:

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

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

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

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

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

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

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

Support

Visit the HP Software Support web site at:

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

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

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

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

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

To find more information about access levels, go to:

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

Table of Contents

Welcome to This Guide	13
How This Guide Is Organized	13
Who Should Read This Guide	14
How Do I Find the Information That I Need?	14
TransactionVision and Transaction Management Documentation ..	15
Additional Online Resources.....	16
Documentation Updates	17
Chapter 1: Introduction to TransactionVision	21
About TransactionVision.....	21
Architecture	22
TransactionVision in the Business Service Management Deployment Environment.....	25
Chapter 2: TransactionVision Deployment Planning	31
Installation Packages	31
Compatibility Matrixes	32
Sizing and Tuning	33
Chapter 3: Reviewing System Requirements.....	35
Supported Processing Server Platforms	37
Supported Database Management Systems.....	38
Supported Messaging Middleware Providers.....	39
Supported WebSphere MQ Agent Platforms.....	40
Supported Java Agent Platforms.....	42
Supported CICS Agent Platforms	45
Supported BEA Tuxedo Agent Platforms.....	46
Supported NonStop TMF Agent Platforms.....	46
Supported .NET Agent Platforms	46
Supported Browser Configurations	46
LDAP Support	47
Java Support.....	47
Flash Player Support	47
Localization and I18N Support	47

Chapter 4: Preparing to Install the TransactionVision Processing Server	51
About the TransactionVision Processing Server	51
Overview of the Processing Server Installation and Configuration ...	52
Chapter 5: Installing the Processing Server	55
Installing the Processing Server on Windows.....	55
Installing the Processing Server on UNIX	56
Uninstalling the Processing Server From a Windows Host	58
Uninstalling the Processing Server From a UNIX Host.....	60
Chapter 6: Configuring Databases	63
About Configuring Databases	63
Supported Databases	64
Controlling Database Access	64
Setting DB2 Variables	65
Setting Oracle Variables	65
DBMS Performance Tuning.....	66
DBMS Disk Space Requirements	69
Configuring Databases for Unicode Data	70
Using Table Partitioning	71
Chapter 7: Post-Install Tasks for the TransactionVision Processing Server	75
Deploy the Required Applications to the BSM Server	75
About the TransactionVision Licenses	76
Set Up Security	77
Add the Processing Server to the TransactionVision Deployment Environment	78
Chapter 8: Configuring Analyzer Logging	79
Log Files	79
Circular Logging.....	80
Using Windows and UNIX System Logs.....	82
Enabling SMTP Logging	84
Enabling SNMP Logging.....	85
Enabling JMS Logging	85
Chapter 9: Preparing to Install TransactionVision Agents	91
Applications That Can Be Monitored	92
About WebSphere MQ (WMQ) Agents	93
About Java Agents	94
About .NET Agent.....	95
About BEA Tuxedo Agent.....	96
About NonStop TMF Agent.....	96

Chapter 10: Installing WebSphere MQ and User Event Agents on Windows	97
Starting the Installation Program on Windows	97
Initial Installation.....	98
Upgrade Installation.....	99
Modifying the Installation	101
Uninstalling Agents.....	102
Chapter 11: Installing WebSphere MQ and User Event Agents on UNIX Platforms	105
Installing Agents.....	105
Uninstalling Agents.....	108
Chapter 12: Installing Agents on i5/OS	111
Starting the Installation Program on i5/OS	111
Chapter 13: Installing and Configuring the Java Agent	113
About the Java Agent.....	114
Java Agent Install and Configuration Workflow	114
Task 1: Determine if the Predefined Java Agent Will Collect the Desired Events	115
Task 2: Install the Java Agent for Your Platform.....	116
Task 3: Run the JRE Instrumenter for each Target JRE	128
Task 4: Set Up Messaging Queues	137
Task 5: Manually Configure the Application Servers to Enable the Instrumentation.....	138
Task 6: Enable the Java Agent in Applications to Be Monitored	139
Task 7: Set up Communication Links that Use the Agent.....	140
Optional Task: Configuring Custom User Events.....	140
Optional Task: Silent Installation of the Java Agent.....	141
Uninstalling the Java Agent	142
Chapter 14: Installing and Configuring Agents on z/OS	143
About Agents in the z/OS Environment	143
Base Component Installation Summary	144
Base Component Installation Procedure	145
Basic Configuration Steps for the SLD Agents: CICS, WMQ Batch, WMQ IMS, and WMQ CICS	152
Basic Configuration Steps for the SLM Agent: WMQ IMS Bridge.....	156

Chapter 15: z/OS Components–Operation and Customization	159
Component Overview	160
Component Hierarchy	162
Architectural Overview.....	163
Component Configurations, Single Agent and Multi-Agent.....	164
Controlling the TransactionVision Manager	166
Inquiring about Event Collection	172
Controlling Agents (AgentEvent Collectors)	173
Data Space Buffer Queue Considerations.....	177
Stub Program Usage by the MQ-Batch, IMS, MQ-IMS Bridge Agents	179
Using the WebSphere MQ-IMS Bridge Agent	184
Guidelines for Agent Operation.....	191
Chapter 16: Installing and Configuring Agents on	
BEA Tuxedo	193
Preparing for the Installation	193
Running the Installation	194
Rebinding the Tuxedo Agent	195
Uninstalling Agents.....	195
Configuring BEA Tuxedo Agents	196
Chapter 17: Installing and Configuring the .NET Agent.....	201
About .NET Agent Installer	202
Installing the .NET Agent.....	202
Configuring the .NET Agent	208
Restarting IIS.....	215
Determining the Version of the .NET Agent	216
Uninstalling the .NET Agent.....	216
SSL Configuration for .NET Agents	216
Chapter 18: Installing and Configuring agents on NonStop TMF ...	217
About the NonStop TMF Agent.....	218
Preparing for the Installation	218
Installing the NonStop TMF Agent	219
Startup/Shutdown	220
Configuring the NonStop TMF Agent.....	221
Uninstalling the NonStop TMF Agent	222
Chapter 19: Configuring WebSphere MQ Agents.....	223
Configuring the WebSphere MQ Agent Library	223
Configuring the WebSphere MQ API Exit Agent.....	229
WebSphere MQ Agents and FASTPATH_BINDING	237
Using Agents with WebSphere MQ Samples	238
WebSphere MQ Client Application Monitoring.....	238

Chapter 20: Configuring the Proxy Agent	245
About the Proxy Agent.....	245
Application Requirements.....	246
Enabling the Proxy Agent	246
Configuring the Proxy Definition File	246
Configuring the User Interface	249
Chapter 21: Configuring Agent Logging	251
Log Files	251
Circular Logging	252
Trace Logging	254
Configuring Separate Log Files for Multiple Agent Instances.....	255
Using Windows and UNIX System Logs	256
Chapter 22: Configuring Security	261
About Security in TransactionVision	261
Managing User Permissions	262
Securing with Light Weight Single Sign On (LW-SSO)	267
Basic Authentication Configuration	269
Securing the TransactionVision Database.....	269
Index	283

Table of Contents

Welcome to This Guide

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

This chapter includes:

- ▶ How This Guide Is Organized on page 13
- ▶ Who Should Read This Guide on page 14
- ▶ How Do I Find the Information That I Need? on page 14
- ▶ TransactionVision and Transaction Management Documentation on page 15
- ▶ Additional Online Resources on page 16
- ▶ Documentation Updates on page 17

How This Guide Is Organized

This guide contains the following chapters/parts:

Part I Introduction to TransactionVision

Introduces TransactionVision and provides an overview of the TransactionVision 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.

Who Should Read This Guide

This guide is intended for the following users of TransactionVision:

- ▶ Application developers or configurators
- ▶ System or instance administrators
- ▶ Database administrators

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

How Do I Find the Information That I Need?

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

You can access the Documentation Library by doing the following:

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

TransactionVision and Transaction Management Documentation

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

The TransactionVision and Transaction Management documentation includes:

- ▶ The *TransactionVision Deployment Guide* describes the installation and configuration of the TransactionVision Processing Servers and agents in the Business Service Management deployment environment. This guide is available as a PDF in the Documentation Library.
- ▶ *Using Transaction Management* describes how to set up and configure TransactionVision to track transactions and how to view and customize reports and topologies of business transactions. It also describes how to define business transaction CIs. This guide is available as the Transaction Management Portal or as a PDF in the Business Service Management Online Documentation Library.
- ▶ The *TransactionVision 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 Planning Guide* contains important information for sizing and planning new installations. This guide is available by download from the HP Software Product Manuals site. See "Documentation Updates" on page 4.

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

Readme. Provides a list of version limitations and last-minute updates. From the HP Business Service Management DVD root directory or download package root directory, double-click **readme<version>.html**. You can also access the most updated readme file from the HP Software Support Web site.

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

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

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

Additional Online Resources

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

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

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.

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>).

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>).

Welcome to This Guide

Part I

Introduction to TransactionVision

1

Introduction to TransactionVision

This chapter includes:

- ▶ About TransactionVision on page 21
- ▶ Architecture on page 22
- ▶ TransactionVision in the Business Service Management Deployment Environment on page 25

About TransactionVision

HP TransactionVision is the transaction tracking solution that non-intrusively records individual electronic events generated by a transaction flowing through a computer network.

More importantly, TransactionVision's patented "Transaction Constructor" algorithm assembles those events into a single coherent business transaction.

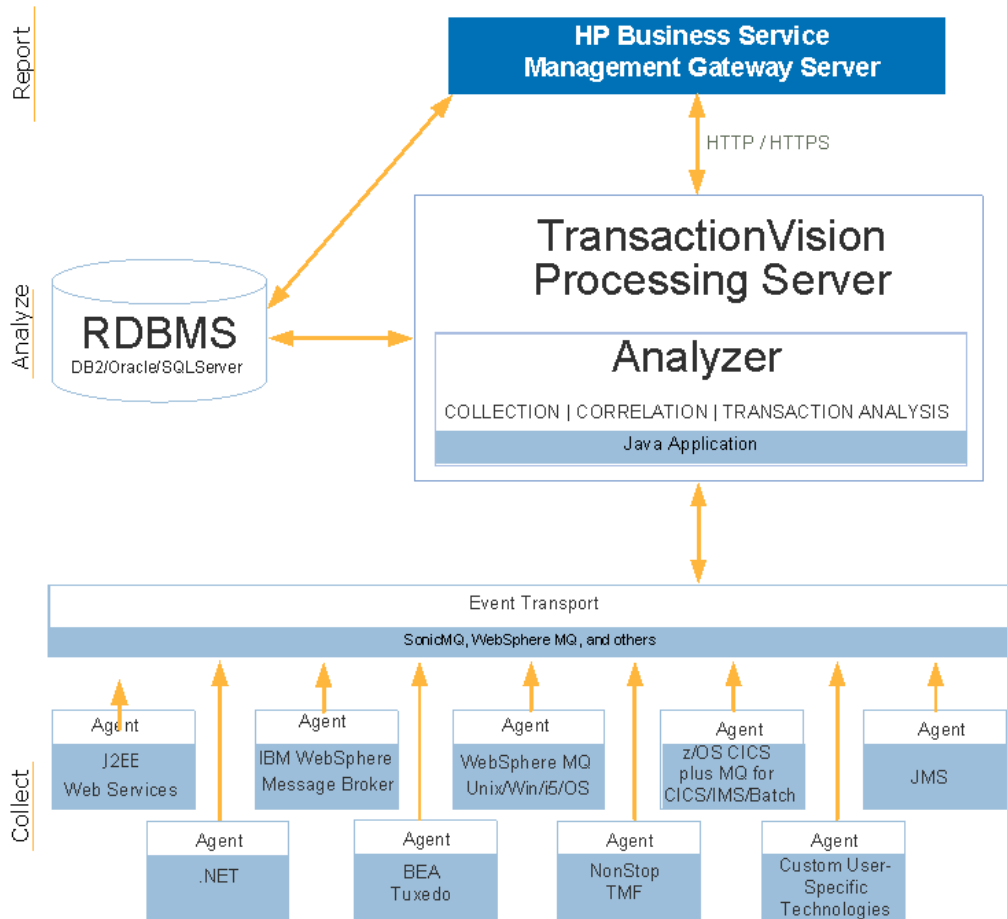
Key capabilities of TransactionVision include:

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

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

Architecture

The following diagram shows the key components of TransactionVision.



Each component is described in the sections that follow.

TransactionVision Processing Server

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

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

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

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

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

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

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

See Part II for information about the installation and configuration of the TransactionVision Processing Servers.

Analyzers

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

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

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

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

Agents

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

Each agent monitors calls made by supporting technologies on that system and compares them against filter conditions. If the call matches the filter conditions, the agent collects entry information about the call, then passes the call on to the appropriate library for processing. When the call returns, the agent collects exit information about the call. It then combines the entry and exit information into a TransactionVision event, which it forwards to the Analyzer by placing it on a designated event queue.

See Part III 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 63 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:

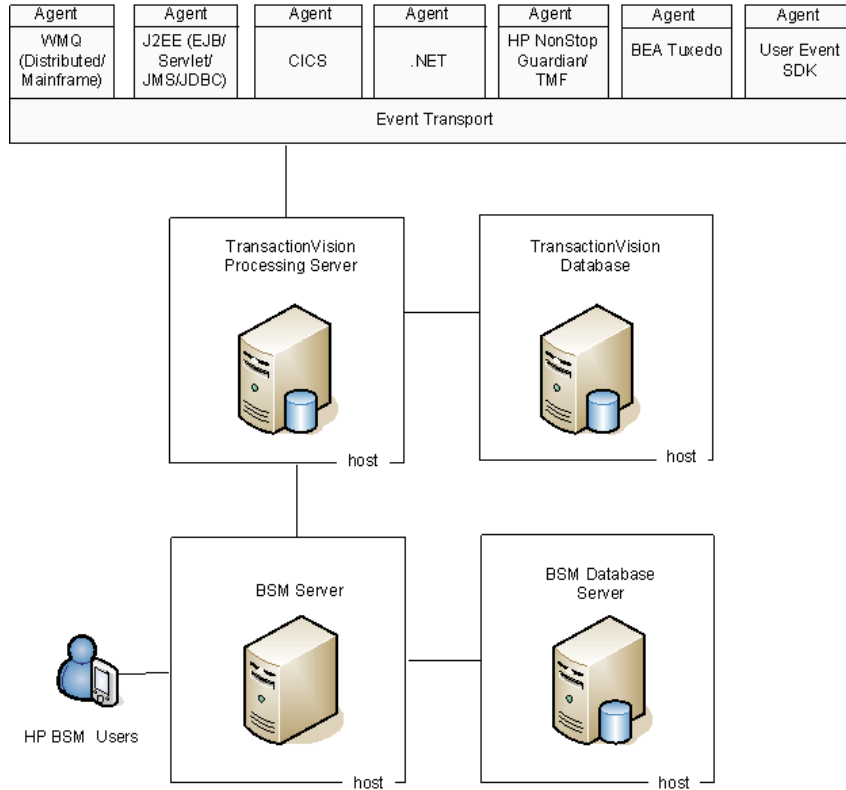
- One-Machine Deployment
- Two-Machine Deployment

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

One-Machine Deployment

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

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

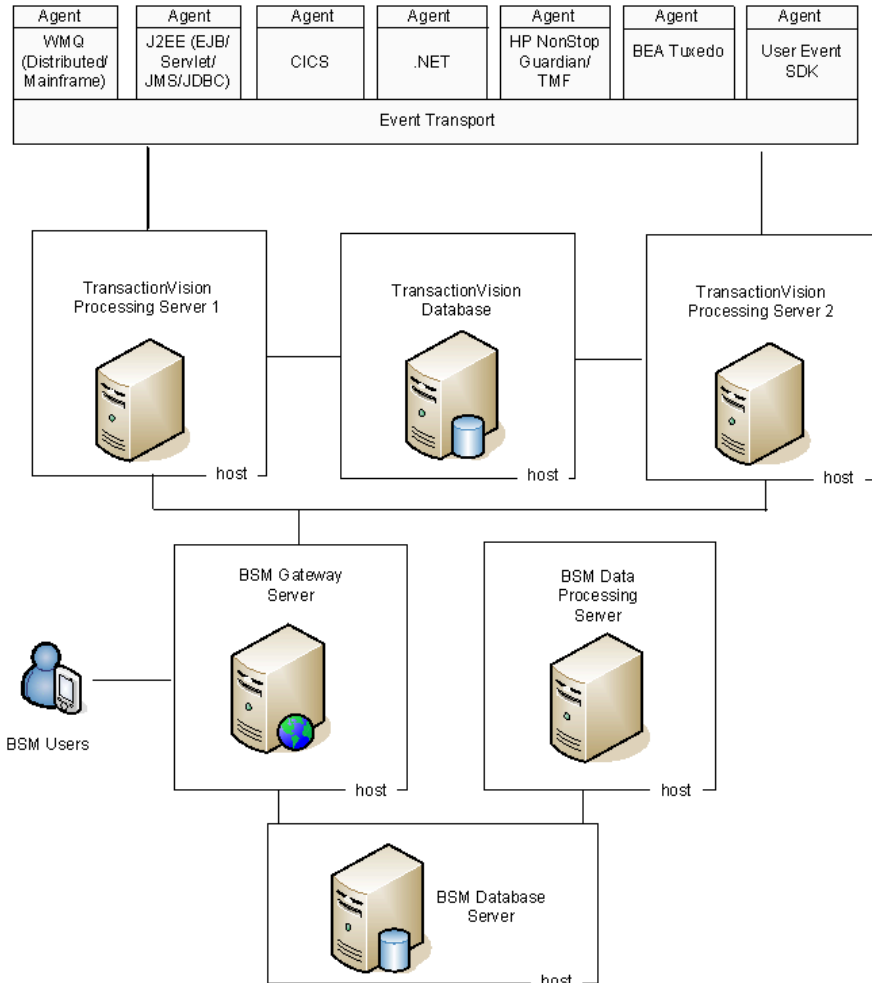


Note: Each TransactionVision Processing Server contains an embedded database that can be used in place of the TransactionVision database for POC or other testing purposes.

Two-Machine Deployment

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

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



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

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

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

A Closer Look at the TransactionVision Processing Server

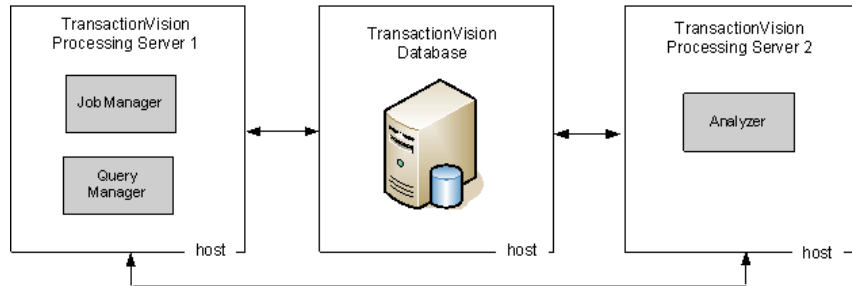
When the deployment environment includes multiple Processing Servers, you have several options in how the Analyzer instances, Job Manager, and Query Manger components are placed.

The guidelines for the deployment environment are:

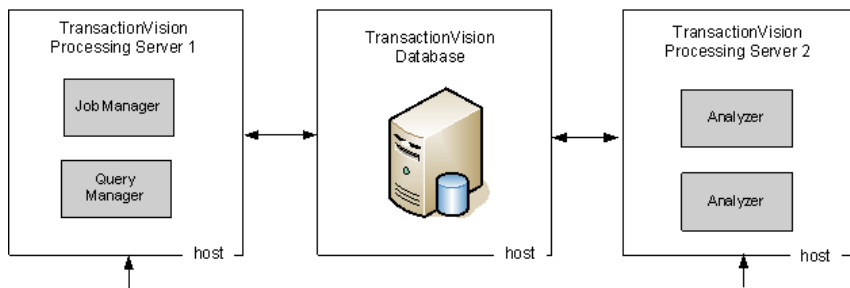
- ▶ The TransactionVision deployment environment must have at least one Processing Server.
- ▶ The TransactionVision deployment environment must have at least one Analyzer, one Job Manager, and one Query Engine. Each of these runs in the context of a Processing Server.
- ▶ The TransactionVision deployment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

Some possible configurations are shown below.

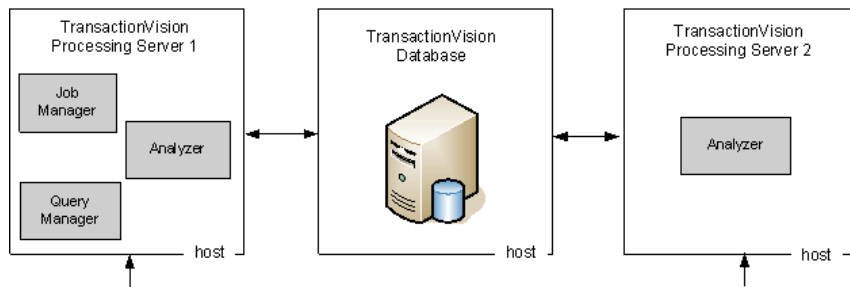
- One Processing Server dedicated to the Job and Query Managers and one Processing Server dedicated to the Analyzer:



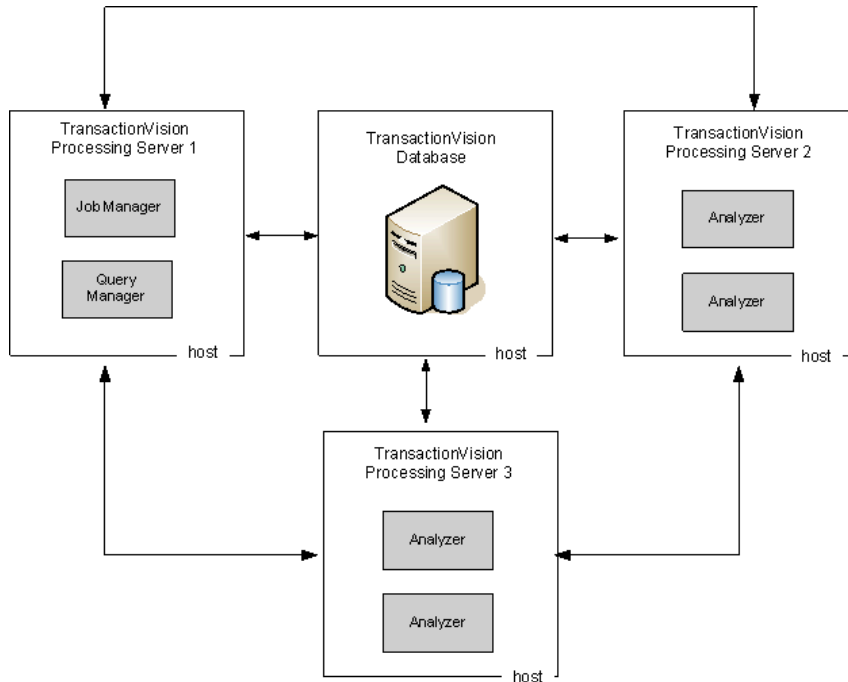
- One Processing Server dedicated to the Job and Query Managers, one Processing Server dedicated to two instances of the Analyzer



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



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



2

TransactionVision Deployment Planning

This chapter includes:

- ▶ Installation Packages on page 31
- ▶ Compatibility Matrixes on page 32
- ▶ Sizing and Tuning on page 33

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

The following table describes the compatible versions of BSM and the TransactionVision Processing Server.

HP Business Service Management	TransactionVision Processing Server
9.00	9.00

The following table describes the compatible versions of the TransactionVision Processing Server and the TransactionVision agents.

TransactionVision Agent	Latest Version	Compatible TransactionVision Analyzer/ Processing Server
NonStop TMF Agent	8.00	8.0x, 9.00
BEA Tuxedo Agent	9.00	9.00
WebSphere MQ Agent	9.00	9.00
Agent for CIS, WMQ Batch, and WMQ IMS on z/OS	9.00	9.00
HP Diagnostics/ TransactionVision Java Agent	9.00	9.00
HP Diagnostics/ TransactionVision .NET Agent	9.00	9.00

Note: All TransactionVision 8.0x agents are compatible with 9.0x TransactionVision Processing Servers.

Sizing and Tuning

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

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

3

Reviewing System Requirements

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

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

This chapter includes:

- ▶ Supported Processing Server Platforms on page 37
- ▶ Supported Database Management Systems on page 38
- ▶ Supported Messaging Middleware Providers on page 39
- ▶ Supported WebSphere MQ Agent Platforms on page 40
- ▶ Supported Java Agent Platforms on page 42
- ▶ Supported CICS Agent Platforms on page 45
- ▶ Supported BEA Tuxedo Agent Platforms on page 46
- ▶ Supported NonStop TMF Agent Platforms on page 46
- ▶ Supported .NET Agent Platforms on page 46
- ▶ Supported Browser Configurations on page 46
- ▶ LDAP Support on page 47
- ▶ Java Support on page 47

- ▶ Flash Player Support on page 47
- ▶ Localization and I18N Support on page 47

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

Supported Processing Server Platforms

Operating Environment	WebSphere MQ	TIBCO EMS	SonicMQ
<p>Highly recommended for enterprise deployment:</p> <ul style="list-style-type: none"> ▶ Windows Server 2008 Enterprise Edition, Service Pack 2 ▶ Windows Server 2008 Enterprise x64 Edition, Service Pack 2 <p>Also supported:</p> <ul style="list-style-type: none"> ▶ Windows Server 2008 R2 Enterprise x64 Edition ▶ Windows Server 2003 Enterprise x64 Edition ▶ Windows Server 2003 32-Bit Enterprise Edition, Service Pack 1 or later 	6.0, 7.0	5.1.12	7.6.2
<p>Highly recommended for enterprise deployment:</p> <ul style="list-style-type: none"> ▶ RedHat Enterprise Linux 5.2 x86 64-bit <p>Also supported:</p> <ul style="list-style-type: none"> ▶ RedHat Enterprise Linux 5.2 x86 32-bit 	6.0, 7.0	5.1.12	7.6.2

Note: The TransactionVisionProcessing Server must be at the same version as HP Business Service Management platform to which it will be deployed.

Supported Database Management Systems

The following database management systems are supported by TransactionVision. These database server configurations can be accessed remotely via the DataDirect Connect ODBC drivers included with TransactionVision. You do not need to install vendor-specific database client software on the Analyzer host

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

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

Supported Messaging Middleware Providers

Agent	WebSphere MQ	Transaction - Vision SonicMQ ¹	SonicMQ ¹	HTTP	WebLogic JMS and TIBCO EMS
Java Agent (EJB, Servlet, JMS and JDBC)	✓	✓	✓		✓ ²
.NET Agent	✓	✓	✓		
WebSphere MQ Agent	✓				
WMQ-IMS Agent	✓				
WMQ-CICS Agent	✓				
CICS Agent	✓				
BEA Tuxedo Agent				✓	
NonStop TMF Agent				✓	

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

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

Supported WebSphere MQ Agent Platforms

Platform	Operating Environment	WebSphere MQ	Supports WMQ API Exit Agent
Microsoft Windows	Windows Server 2003 x86 and x64, 32 and 64-bit	6.0, 7.0 ^{1, 6}	Yes
	Windows Server 2008 x86 and x64, 32 and 64-bit Windows Server 2008 R2 x86 and x64, 32 and 64-bit	7.0 ^{1, 6}	Yes
Sun Solaris ⁴	Solaris 9, 10 SPARC	6.0, 7.0 ^{1, 6} (32-bit) 6.0, 7.0 ^{1, 6} (64-bit)	Yes
Hewlett-Packard HP-UX ⁴	HP-UX 11i v3 PA-RISC & Itanium	6.0, 7.0 ^{1, 6} (32-bit) 6.0, 7.0 ¹ (64-bit)	Yes
IBM AIX ⁴	AIX 5L 5.3, 6.1 POWER	6.0, 7.0 ^{1, 6} (32-bit) 6.0, 7.0 ^{1, 6} (64-bit)	Yes
RedHat Linux ⁴	Enterprise Linux 5.2 x86 32 and 64-bit	6.0, 7.0 ^{1, 6} (32-bit), 6.0, 7.0 ^{1, 6} (64-bit)	Yes
IBM i5/OS ²	i5/OS V5R4 iSeries	6.0, 7.0 ^{1, 6}	Yes
IBM z/OS ^{3, 5}	z/OS 1.8, 1.9, 1.10 zSeries z/OS 1.8, 1.9, 1.10 Batch z/OS 1.8, 1.9, 1.10 CICSTS 2.x, 3.x z/OS 1.8, 1.9, 1.10 RRS z/OS 1.8, 1.9, 1.10 IMS 7.x, 8.x, 9.x	6.0, 7.0 ¹	N/A

¹ TransactionVision does support WMQ applications running against WMQ 7. However, events will not be generated from new WMQ 7 APIs

added in that release. Applications that only use WMQ 6 APIs are fully supported whether they are run against WMQ 6 or 7.

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

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

⁴ Important! When using multithreaded applications with WebSphere MQ on UNIX systems, ensure that the applications have sufficient stack size for the threads. IBM recommends using a stack size of at least 256KB when multithreaded applications are making MQI calls. When using the TransactionVision WebSphere MQ Agent, even more stack size may be required; the recommended stack size is at least 512KB. For more information, see Chapter 7, "Connecting to and disconnecting from a queue manager," in the *WebSphere MQ Application Programming Guide*.

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

⁶ For WebSphere MQ 7.0, Fix Pack 7.0.0.1 is required in order to use the WebSphere MQ API Exit Agent and the Analyzer.

Supported Java Agent Platforms

Servlet Technology

Platform	Operating Environment	Application Servers
Microsoft Windows	<ul style="list-style-type: none"> ▶ Windows 2003 Server x86 and x64 (x86-64), 32 and 64-bit ▶ Windows 2008 Server Service Pack 1 ▶ Windows 2008 Server SP2, 32- and 64-bit 	IBM WebSphere Application Server 6.0 ¹ , 6.1 ^{2,3} , 7 BEA WebLogic Application Server 9.2.3, 10
Sun Solaris	Solaris 9, 10 SPARC	IBM WebSphere Application Server 6.0 ¹ , 6.1 ^{2,3} , 7 BEA WebLogic Application Server 9.2.3, 10
IBM AIX	AIX 5L 5.3, 6.1 POWER	IBM WebSphere Application Server 6.0 ¹ , 6.1 ^{2,3} , 7 BEA WebLogic Application Server 9.2.3, 10
RedHat Linux	RedHat Enterprise Linux 5.2 x86 32 and 64-bit	IBM WebSphere Application Server 6.0 ¹ , 6.1 ^{2,3} , 7 BEA WebLogic Application Server 9.2.3, 10

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

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

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

EJB Technology

Platform	Operating Environment	Application Servers
Microsoft Windows	<ul style="list-style-type: none"> ▶ Windows 2003 Server x86 and x64 (x86-64), 32 and 64-bit ▶ Windows 2008 SP1 ▶ Windows 2008 Server SP2, 32 and 64-bit 	IBM WebSphere Application Server 6.0 ¹ , 6.1 FP9+ ^{2,3} , 7 BEA WebLogic Application Server 9.2.3, 10
Sun Solaris	Solaris 9, 10 SPARC	IBM WebSphere Application Server 6.0 ¹ , 6.1 FP9+ ^{2,3} , 7 BEA WebLogic Application Server 9.2.3, 10
IBM AIX	AIX 5L 5.3, 6.1 POWER	IBM WebSphere Application Server 6.0 ¹ , 6.1 FP9+ ^{2,3} , 7 BEA WebLogic Application Server 9.2.3, 10
RedHat Linux	RedHat Enterprise Linux 5.2 x86 32 and 64-bit	IBM WebSphere Application Server 6.0 ¹ , 6.1 FP9+ ^{2,3} , 7 BEA WebLogic Application Server 9.2.3, 10

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

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

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

JMS Technology

Platform	Operating Environment	JMS Service Provider
Microsoft Windows	<ul style="list-style-type: none"> ▶ Windows 2003 Server x86 and x64 (x86-64), 32 and 64-bit ▶ Windows 2008 SP1 ▶ Windows 2008 Server SP2, 32 and 64-bit 	WebSphere MQ 6.0 ¹ , 7 TIBCO EMS 4.2.0, 4.4.2 SonicMQ 7.6.2, 7 WebLogic JMS 8.1.6, 9.2.3, 10
Sun Solaris	Solaris 9, 10 SPARC	WebSphere MQ 6.0 (32-bit and 64-bit) ¹ , 7 TIBCO EMS 4.2.0, 4.4.2 SonicMQ 7.6.2 WebLogic JMS 8.1.6, 9.2.3, 10
IBM AIX	AIX 5L 5.3, 6.1 POWER	WebSphere MQ 6.0 (32-bit and 64-bit) ¹ , 7 TIBCO EMS 4.2.0, 4.4.2 SonicMQ 7.6.2 WebLogic JMS 8.1.6, 9.2.3, 10
RedHat Linux	RedHat Enterprise Linux 5.2 x86 32 and 64-bit	WebSphere MQ 6.0(32-bit and 64-bit) ¹ , 7 TIBCO EMS 4.2.0, 4.4.2 SonicMQ 7.6.2 WebLogic JMS 8.1.6, 9.2.3, 10

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

JDBC Technology

Platform	Database Version	Operating Environment
Oracle	9.2, 10g, RAC 10g	Windows 2003 Server 32 and 64-bit Windows 2008 Server SP2, 32 and 64-bit Solaris 9, 10 AIX 5L 5.3, 6.1 RedHat Enterprise Linux x86 5.2 32 and 64-bit
DB2	9.1	Windows 2003 Server 32 and 64-bit Windows 2008 Server SP2, 32 and 64-bit Solaris 9, 10 AIX 5L 5.3, 6.1 RedHat Enterprise Linux 5.2 x86 32 and 64-bit

Supported CICS Agent Platforms

Platform	Operating Environment	WebSphere MQ
z/OS	z/OS 1.8, 1.9, 1.10 CICS TS 2.x, 3.x zSeries ¹	6.0, 7.0

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

Supported BEA Tuxedo Agent Platforms

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

Supported NonStop TMF Agent Platforms

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

Supported .NET Agent Platforms

Platform	Operating Environment	.NET
Microsoft Windows	<ul style="list-style-type: none"> ▶ Windows Server 2003 x86 and x64, 32 and 64-bit ▶ Windows Server 2008, SP2 (32 & 64-bit) ▶ Windows Server 2008 R2 (32 & 64-bit) 	1.1, 2.0, 3.0, 3.5, 4.0

Supported Browser Configurations

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

LDAP Support

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

Java Support

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

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

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

Flash Player Support

Some TransactionVision reports and topologies require Adobe Flash Player. See the *HP Business Service Management Deployment Guide* PDF for version information.

Localization and I18N Support

TransactionVision 9.0 is not localized.

Part II

Processing Server Installation

4

Preparing to Install the TransactionVision Processing Server

This chapter includes:

- About the TransactionVision Processing Server on page 51
- Overview of the Processing Server Installation and Configuration on page 52

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 Unix.

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

Though normally managed through the Administration user interface, you can also initiate service shutdown or get status information. See the AnalyzerManager utility description in *Using Transaction Management*.

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

Overview of the Processing Server Installation and Configuration

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

- 1** Locate the host on which the Processing Server will be installed. This host will need access to the TransactionVision database, which can optionally be on the same host.

In most deployment environment 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 37.
- 3** Install the Processing Server.

For installing on a host running a Windows operating system, see "Installing the Processing Server" on page 55.

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

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

Note: Be sure that the Messaging Middleware product that you install is supported by the agent types in your deployment environment. See "Supported Messaging Middleware Providers" on page 39.

5 Set up the database.

See the "Configuring Databases" chapter.

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

5

Installing the Processing Server

This chapter includes:

- Installing the Processing Server on Windows on page 55
- Installing the Processing Server on UNIX on page 56
- Uninstalling the Processing Server From a Windows Host on page 58
- Uninstalling the Processing Server From a UNIX Host on page 60

Installing the Processing Server on Windows

To install a new Processing Server 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 do not need to be shut down.
- 3** From Windows Explorer, double-click **HPTVProcServer_<version>_win.exe**. The InstallShield Welcome screen appears.
- 4** Click **Next** and wait until the TransactionVision Setup Welcome screen appears.
- 5** 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.

- 6 On the Setup Welcome screen, click **Next** to display the TransactionVision license agreement.
- 7 Click **Yes** to accept the license agreement. The User Information screen appears.
- 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 UNIX

The following table shows the installation file names for the TransactionVision Processing Server package for each UNIX platform.

Platform	File Name
AIX	HPTVProcServer_<version>_aix.tgz
Linux	HPTVProcServer_<version>_linux.tgz
Solaris	HPTVProcServer_<version>_sol.tgz

The Processing Server is installed to the following directory on Solaris and Linux:

/opt/HP/TransactionVision. On AIX, the Processing Server is installed to: **/usr/lpp/HP/TransactionVision**.

To install a new Processing Server on UNIX platforms, perform the following steps.

- 1** Change to the directory location of the TransactionVision installation files (either a DVD device or download directory). NOTE: On Solaris, you must instead copy the installation files from the DVD device to a temporary directory on your host's hard drive.
- 2** Unzip and untar the packages for your platform; see "Installing the Processing Server on UNIX" on page 56. For example:

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

- 3** Log in as superuser:

```
su
```

- 4** Set the JAVA_HOME variable to a JDK or JRE installation directory.
- 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
2. TransactionVision WebSphere MQ Agent
3. TransactionVision User Event Agent

99. All of above
q. Quit install

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

- 6** Type 1 and press **Return**.

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

```
Installation of <TVANLZR> was successful.
```

The TransactionVision component menu is displayed.

- 7** Type **q** and press **Return** to quit the installation process.
- 8** Run `<TVISION_HOME/bin/run_topaz start`

Uninstalling the Processing Server From a Windows Host

To uninstall the Processing Server from a Windows host, perform the following steps:

- 1** From the Start menu, choose **Control Panel**.
- 2** Double-click **Add/Remove Programs**.
- 3** Select the HP TransactionVision package you want to uninstall and click **Change/Remove**. The maintenance menu screen appears.
- 4** Prior to uninstalling the Processing Server, the HP BAC Windows Service and all associated TransactionVision processes must be stopped. 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** 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**.

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

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

Uninstalling the Processing Server From a UNIX Host

To uninstall TransactionVision components, perform the following steps:

- 1 Log in as superuser:

```
su
```

- 2 Enter the following command:

```
./tvininstall_<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 Processing Server
2. TransactionVision WebSphere MQ Agent
3. TransactionVision User Event Agent

99. All of above
q. Quit install

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

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

- 3 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 Return. Otherwise, type **1** and press Return to uninstall only the Processing Server, or **99** and press Return to uninstall all TransactionVision components.
- 4 The installation script uninstalls the specified components, then displays the menu again.
- 5 Enter **1** and press **Return**.

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

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

- 6 Type **q** and press **Return** to quit the installation.

6

Configuring Databases

This chapter includes:

- ▶ About Configuring Databases on page 63
- ▶ Supported Databases on page 64
- ▶ Controlling Database Access on page 64
- ▶ Setting DB2 Variables on page 65
- ▶ Setting Oracle Variables on page 65
- ▶ DBMS Performance Tuning on page 66
- ▶ DBMS Disk Space Requirements on page 69
- ▶ Configuring Databases for Unicode Data on page 70
- ▶ Using Table Partitioning on page 71

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. See "How to Create a Processing Server" in *Using Transaction Management*.

Supported Databases

The following databases and associated platforms are supported by TransactionVision.

- DB2 9.2
- Oracle 10g
- Oracle RAC 10g
- Oracle RAC 11g
- Microsoft SQL Server 2005
- Microsoft SQL Server 2008

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.

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

```
db2 connect to tvision
db2 get db cfg for tvision
db2 update db cfg for tvision using APP_CTL_HEAP_SZ 1024
db2 update db cfg for tvision using 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.

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

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

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

- ▶ Run the insert test with one thread and with record size of 1KB - this will gauge the raw event insert performance.
- ▶ Run the insert test with multiple threads and with a record size of 1KB. This will test whether the insert can benefit from multiple threads. Usually the thread count is set to two times the number of CPUs.
- ▶ Run the insert test with one thread and with record size of (7 KB + average message size) - this will gauge the analyzed event insert performance.
- ▶ Run the insert test with multiple threads and with record size of (7 KB + average message size). This will test if the insert can benefit from multiple threads. Usually the thread count is set to two to four times the number of CPUs.
- ▶ If the Processing Server host and the DBMS host are different, the above tests should be run on the Processing Server host. However at least one test should be run on the DBMS host to see if there is any communication/DB client configuration related issues.

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

- ▶ Disk I/O usage for all involved physical disks (tablespaces and log files), especially I/O busy percentage.
- ▶ CPU usage, including wait time percentage.

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

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

Updating Database Statistics

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

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

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

DBMS Disk Space Requirements

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

$$(\text{Average message size} + 7\text{K Byte}) \times \text{Event Rate} \times \text{Event Retention Time}$$

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 72
- ▶ "TransactionVision Database Properties" on page 72

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	<ul style="list-style-type: none"> ▶ 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.

Creating tables with range partitioning

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

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

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

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

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

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

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

Adding and dropping of partitions

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

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

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

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

Example to add two new partitions:

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

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

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

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

Custom XDM tables

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

Example:

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

The partition utilities (CreateSqlScript and PartitionUtil) 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 Options for Analyzers in *Using Transaction Management*.

7

Post-Install Tasks for the TransactionVision Processing Server

This chapter includes:

- ▶ Deploy the Required Applications to the BSM Server on page 75
- ▶ About the TransactionVision Licenses on page 76
- ▶ Set Up Security on page 77
- ▶ Add the Processing Server to the TransactionVision Deployment Environment on page 78

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

Autopass licensing for TransactionVision is enabled and managed by BSM. See *Platform Administration*.

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

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

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

Capacity is calculated once per hour.

Business Service Management License Management page (Admin > Platform > Setup and Maintenance > License Management) reports:

- ▶ whether the TV license is valid.
- ▶ the TV consumed license count (application instance count).
- ▶ the purchased TV license capacity (maximum number of application instances).

You install the TransactionVision license from either the Business Service Management 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.

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

Set Up Security

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

You should set up specific users and roles before accessing the Processing Server configuration. See "Managing User Permissions" on page 262.

Add the Processing Server to the TransactionVision Deployment Environment

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

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

8

Configuring Analyzer Logging

This chapter includes:

- Log Files on page 79
- Circular Logging on page 80
- Using Windows and UNIX System Logs on page 82
- Enabling SMTP Logging on page 84
- Enabling SNMP Logging on page 85
- Enabling JMS Logging on page 85

Log Files

By default, all TransactionVision components log error and warning messages to the appropriate log files.

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

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

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

You can view the content of each Analyzer's log files through the Transaction Management Administration pages or through the <TVISION_HOME>/logs directory.

Circular Logging

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

This section includes:

- ▶ "Log File Naming" on page 80
- ▶ "Maximum Log File Size" on page 81
- ▶ "Maximum Number of Backup Log Files" on page 81
- ▶ "Changing from Circular to Linear Logging" on page 82

Log File Naming

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

- ▶ analyzer.log.2 is removed if it exists.
- ▶ analyzer.log.1 is renamed analyzer.log.2.
- ▶ analyzer.log is renamed analyzer.log.1.
- ▶ A new analyzer.log is created.

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

The <TVISION_HOME>/config/logging/*.Logging.xml files specify the type of logging used, the maximum log file size, and the number of backup log files for each component.

For example, **Sensor.Logging.xml** specifies the configuration for the servlet and JMS Agent. This file contains entries similar to the following:

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

Maximum Log File Size

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

Maximum Number of Backup Log Files

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

Changing from Circular to Linear Logging

To use linear logging rather than circular logging, 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.

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 the following:

- ▶ "Windows Event Appender" on page 82
- ▶ "UNIX Event Appender" on page 83

Windows Event Appender

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

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

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

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

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

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

UNIX Event Appender

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

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

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

Enabling SMTP Logging

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

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

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

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

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

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

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

Enabling SNMP Logging

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

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

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

Enabling JMS Logging

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

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

Choose whether you are configuring using:

- ▶ JNDI
- or
- ▶ Direct WMQ JMS

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

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

- ▶ **ConnectionRetryDelay** is the time before a retry is made if connection fails.
- ▶ **ConnectionRetryTimeout** is the time it will wait before an unresponsive connection times out.
- ▶ **QueueName** is the name of the JNDI object (if JNDI is used), or the actual name of the queue (in the case of WMQ JMS).
- ▶ **Username** and **Password** are optional settings if authentication to the JMS provider is required.

For JNDI Settings

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

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

WMQ JMS Settings

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

```
<appender class="com.bristol.tvision.appender.JMSAppender"
name="JMS_APPENDER">
  <!--connection retry interval in milliseconds -->
  <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

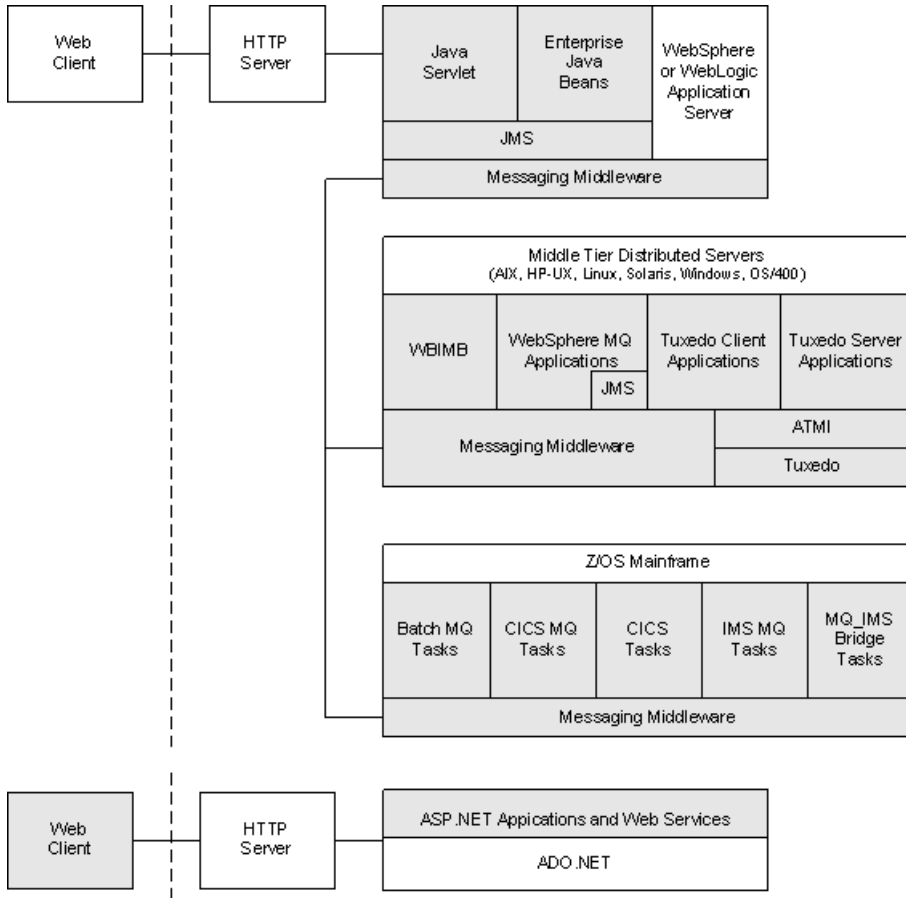
Preparing to Install TransactionVision Agents

This chapter includes:

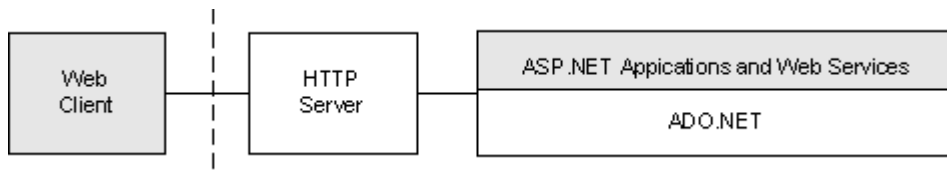
- Applications That Can Be Monitored on page 92
- About WebSphere MQ (WMQ) Agents on page 93
- About Java Agents on page 94
- About .NET Agent on page 95
- About BEA Tuxedo Agent on page 96
- About NonStop TMF Agent on page 96

Applications That Can Be Monitored

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

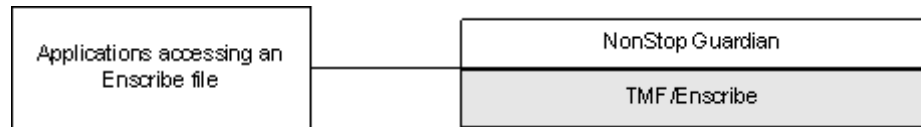


ASP.NET applications can also be monitored:



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

NonStop TMF applications can also be monitored:



About WebSphere MQ (WMQ) Agents

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

Distributed (Non-Mainframe) Platforms

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

- ▶ The **WebSphere MQ Library Agent** intercepts a WebSphere MQ API call by the shared library (or DLL) interception method on distributed platforms. This involves placing the TransactionVision Agent libraries before the WebSphere MQ libraries in the application library path. This method is useful if you need to track MQ APIs for specific applications.
- ▶ The **WebSphere MQ API Exit Agent** uses the WebSphere MQ API exit support. This agent is registered as an exit to the queue manager and invoked when any program connecting to the queue manager invokes a WebSphere MQ API. This method is recommended to collect MQ events from all applications on a queue manager and in particular the listener and the channel agents.

Both of these agents report the same information from an MQ API call. They differ primarily in the mechanism by which they intercept MQ API calls, their usage, and the amount of data they collect from the system.

z/OS Based Agents

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

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

About Java Agents

- ▶ The **Servlet Agent** tracks servlet methods in a J2EE application server. This agent tracks HTTP calls such as HTTP_POST, HTTP_GET, HTTP_PUT, etc., which result in method calls into the J2EE container. The Servlet agent tracks these method invocations by instrumenting the servlet to collect events at the entry and exit of each call.
- ▶ The **JMS Agent** tracks WebSphere MQ Java Message Service or TIBCO EMS events from standalone Java applications as well as from J2EE application servers. This agent tracks JMS interface methods such as send, receive, etc. These methods are tracked by instrumenting the JMS library to collect events at the entry and exit of each call.

- ▶ The **EJB Agent** tracks transactions through business logic within a J2EE application server. This agent tracks all public business methods in an entity bean, session bean or message driven bean. In addition to the business methods, this agent tracks the `ejbCreate`, `ejbPostCreate`, `ejbRemove`, `ejbLoad`, `ejbStore` and `onMessage` methods. These methods are instrumented by the agent to collect events at the entry and exit of each call.
- ▶ The **JDBC Agent** allows users to collect and analyze API and timing information on SQL calls and transactions made to a relational database through the JDBC API.

The capabilities of the TransactionVision Java Agents (JMS, Servlet, EJB and JDBC) and the Diagnostics Java Probe are combined into a single component, HP Diagnostics/TransactionVision Java Agent. The Java Agent instruments and captures events from applications and sends the information to a Diagnostics Server and/or to a TransactionVision Analyzer. In this release the Java Agent can be configured to serve as a Java Probe in a Diagnostics environment or as a Java Agent in a TransactionVision environment. For combined environments, the agent can simultaneously serve as both the Probe and the agent. See Chapter 13, "Installing and Configuring the Java Agent" for details.

About .NET Agent

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

The .NET agent also tracks services implemented with WCF similar to how Webservices are tracked in an ASP.NET environment.

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

About BEA Tuxedo Agent

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

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

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

About NonStop TMF Agent

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

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

10

Installing WebSphere MQ and User Event Agents on Windows

This chapter includes:

- ▶ Starting the Installation Program on Windows on page 97
- ▶ Initial Installation on page 98
- ▶ Upgrade Installation on page 99
- ▶ Modifying the Installation on page 101
- ▶ Uninstalling Agents on page 102

Starting the Installation Program on Windows

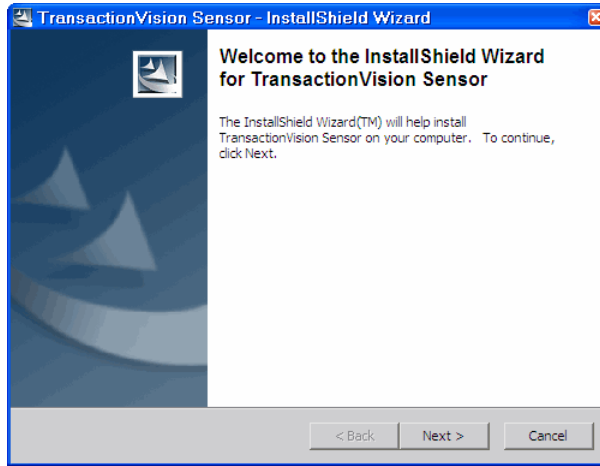
The TransactionVision Agents for WebSphere MQ and User Event are installed as a single package on Windows. This chapter provides instructions for installing these agents.

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

To start the installation program, perform the following steps:

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

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



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

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

Initial Installation

To install a new instance of the agent, perform the following steps:

- 1 On the Setup Welcome screen, click **Next** to display the TransactionVision license agreement.
- 2 Click **Yes** to accept the license agreement. The User Information screen appears.
- 3 Enter your name and company name, then click **Next**. The Destination Location screen appears.

- 4 To install the agents for WebSphere MQ and User Event, select **Complete** and click **Next**. To install only some agents, select **Custom**, click **Next**, select the desired agents, and click **Next**.

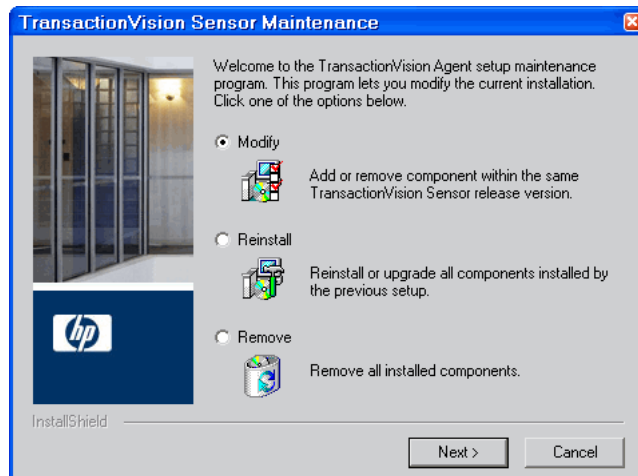
The selected agents are installed in the specified location. The Setup Complete page appears.

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

Upgrade Installation

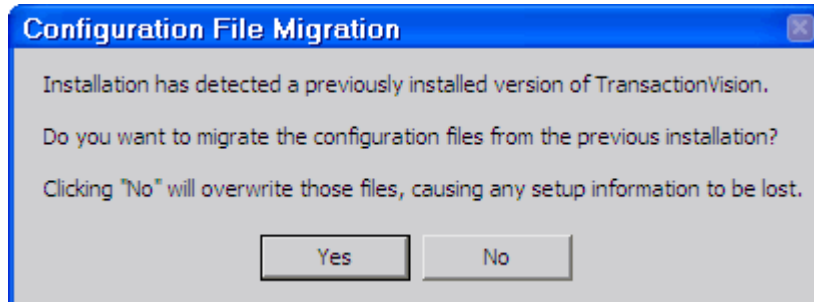
To upgrade the agent on this host, perform the following steps

- 1 For an upgrade installation, double-click **HPTVWMQAgent_<version>.win.exe** and click **Next** on the InstallShield Welcome screen to display the agent setup maintenance menu:



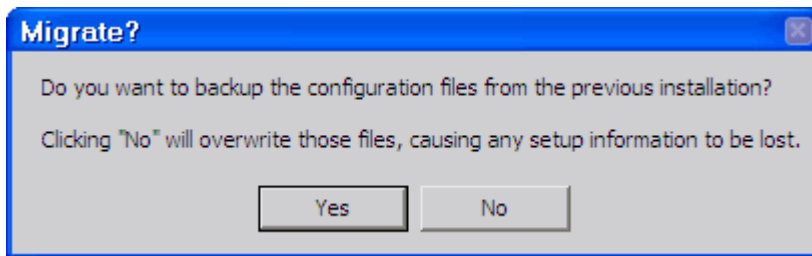
- 2 Select one of the following to upgrade the TransactionVision installation (to modify the installation, see "Modifying the Installation" on page 101):
 - If you want to install TransactionVision agents with different settings from the previous installation, select **Remove** and click **Next** to uninstall the previous installation, then begin the installation procedure again.

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



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

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



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

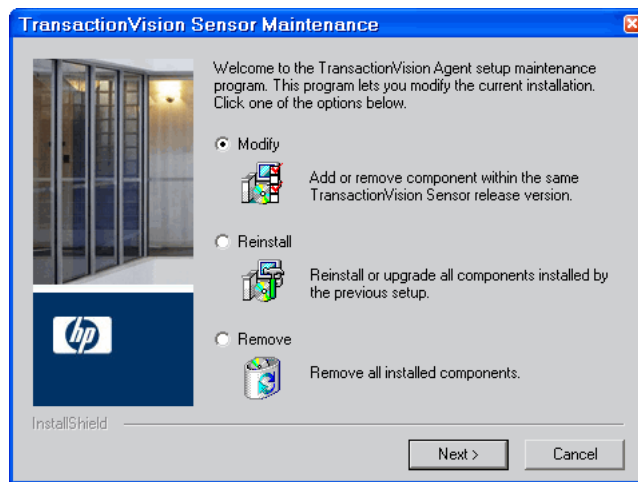
- 4 The installation wizard installs the new version of TransactionVision and displays the Setup Complete screen.
- 5 Click **Finish** to complete the installation.

Modifying the Installation

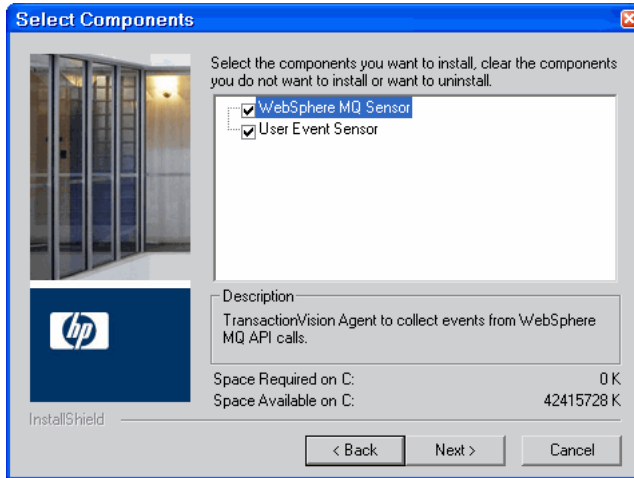
After you install TransactionVision agents on a host, you may want to modify your installation. For example, suppose you initially install the WebSphere MQ Agent on a host, then later decide to install the User Event Agent.

To modify the agent installation perform the following steps:

- 1 Double-click **HPTVWMQAgent_<version>.win.exe** and click **Next** on the InstallShield Welcome screen to display the TransactionVision agent Maintenance screen:



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



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

Uninstalling Agents

To uninstall TransactionVision components, perform the following steps:

- 1 From the Start menu, choose **Settings > Control Panel**.
- 2 Double-click **Add/Remove Programs**.
- 3 Select the HP TransactionVision agent package and click **Change/Remove**. The maintenance menu screen appears.
- 4 Select **Remove** and click **Next** to remove TransactionVision components.
- 5 Click **OK** to confirm that you want to uninstall the specified package. The specified package is uninstalled.

The following types of files are not deleted:

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

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

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

The Uninstallation Complete screen appears.

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

11

Installing WebSphere MQ and User Event Agents on UNIX Platforms

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

Note that TransactionVision provides unique packages for each agent.

This chapter includes:

- Installing Agents on page 105
- Uninstalling Agents on page 108

Installing Agents

This section includes the following:

- "Installation Files" on page 106
- "Installation Steps" on page 106
- "Rebinding the WebSphere MQ Agent on AIX" on page 108

Installation Files

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

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

Installation Steps

To start the installation program, perform the following steps

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

Note: On Solaris and HP-UX, you must copy the installation files from the DVD device to a temporary directory on your host's hard drive.

- 2 Log in as superuser:

```
su
```

- 3 Unzip and untar the packages for your platform; see "Installation Files" on page 106. For example:

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

- 4 Enter the following command to begin the installation procedure:

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

The following menu is displayed:

The following TransactionVision packages are available for installation:

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

99. All of above

q. Quit install

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

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

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

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

3,4

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

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

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

Rebinding the 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 in order for monitored applications to run correctly. For more information about this script, see "Command-Line Utilities" in *Using Transaction Management*.

Uninstalling Agents

To uninstall TransactionVision components, perform the following steps:

- 1 Log in as superuser:

```
su
```

- 2 Enter the following command:

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

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

The following TransactionVision packages are installed on the system:

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

99. All of above
q. Quit uninstall

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

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

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

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

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

12

Installing Agents on i5/OS

This chapter includes:

- Starting the Installation Program on i5/OS on page 111

Starting the Installation Program on i5/OS

To start the Agent installation program on the i5/OS platform, perform the following steps:

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

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

- 2 On an i5/OS machine, either find an existing library to use or create a new a library to copy the installation file to (for example, TVTMP).

- 3 On a PC, FTP the agent installation file **HPTVWMAgent_<version>_i5os.savf** from the CD_ROM and rename it to **agent<version>.savf** to the library created in step 1 on your i5/OS machine. Be sure to set binary mode transfer as follows:

```
ftp> bin
ftp> cd /qsys.lib/tvtmp.lib
ftp> put HPTVWMAgent_<version>_i5os.savf sensor<version>.savf
```

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

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

5 Verify the installation with the following command:

DSPSFWRSC

6 To use the C Sensor, bind your programs to TVSENSOR/LIBMQM.

7 If a new temporary library was created in step 2, it may now be safely deleted.

13

Installing and Configuring the Java Agent

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

This chapter includes:

- ▶ About the Java Agent on page 114
- ▶ Java Agent Install and Configuration Workflow on page 114
- ▶ Task 1: Determine if the Predefined Java Agent Will Collect the Desired Events on page 115
- ▶ Task 2: Install the Java Agent for Your Platform on page 116
- ▶ Task 3: Run the JRE Instrumenter for each Target JRE on page 128
- ▶ Task 4: Set Up Messaging Queues on page 137
- ▶ Task 5: Manually Configure the Application Servers to Enable the Instrumentation on page 138
- ▶ Task 6: Enable the Java Agent in Applications to Be Monitored on page 139
- ▶ Task 7: Set up Communication Links that Use the Agent on page 140
- ▶ Optional Task: Configuring Custom User Events on page 140
- ▶ Optional Task: Silent Installation of the Java Agent on page 141
- ▶ Uninstalling the Java Agent on page 142

About the Java Agent

The Java Agent combines the capabilities of the TransactionVision Java Agent and the Diagnostics Java Probe into a single component. The Java Agent can be configured to serve as a Java Probe in a Diagnostics environment or as a Java Agent in a TransactionVision environment and for combined environments, the agent can simultaneously serve as both the Probe and t"Task 6: Enable the Java Agent in Applications to Be Monitored" on page 139.

The Java Agent is the agent for these technologies: JMS, Servlet, JDBC and EJB.

Java Agent Install and Configuration Workflow

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

- ▶ "Task 1: Determine if the Predefined Java Agent Will Collect the Desired Events" on page 115
- ▶ "Task 2: Install the Java Agent for Your Platform" on page 116
- ▶ "Task 3: Run the JRE Instrumenter for each Target JRE" on page 128
- ▶ "Task 4: Set Up Messaging Queues" on page 137
- ▶ "Task 5: Manually Configure the Application Servers to Enable the Instrumentation" on page 138
- ▶ "Task 7: Set up Communication Links that Use the Agent" on page 140

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

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

- ▶ Servlet methods in a J2EE application server. For instance, HTTP calls such as HTTP_POST, HTTP_GET, HTTP_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 and WMQ from standalone Java applications as well as from J2EE application servers. JMS interface methods such as send and receive. These methods are tracked by instrumenting the JMS library to collect events at the entry and exit of each call.
- ▶ EJBs. Transactions through business logic within a J2EE application server. All public business methods in an entity bean, session bean or message driven bean. In addition to the business methods, the Java Agent tracks the ejbCreate, ejbPostCreate, ejbRemove, ejbLoad, ejbStore and onMessage methods. These methods are instrumented by the Agent to collect events at the entry and exit of each call.
- ▶ JDBC. SQL calls and transactions made to a relational database through the JDBC API.

Other calls can be tracked by using custom code to create user events. See "Optional Task: Configuring Custom User Events" on page 140.

Task 2: Install the Java Agent for Your Platform

Installation Files

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

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

Installing and Configuring the Java Agent on Windows

The following steps describe how to install the Java Agent on a Windows host. These instructions also apply when you are installing the Java Agent on a UNIX machine using the graphical installer.

Launching the Installer on Windows

You can launch the Java Agent installer from the HP Software web site, from the Diagnostics or TransactionVision product disk, or from the Downloads page in BSM.

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

To launch the installer from the HP Software Web site:

- 1 Go to the HP Software Web site's Download Center HP- BTO Software.
- 2 Enter TransactionVision in the Keyword field and "Trial Software" in the "Refine Search by Resource Type" field, and click Search.

3 Continue with "Running the Installation on Windows" on page 117.

To launch the installer from BSM:

- 1** Select **Admin > Platform** from the top menu in BSM and click the **Setup and Maintenance** tab.
- 2** On the Downloads page, click the appropriate link to download the Java Agent installer for Windows.
- 3** Continue with "Running the Installation on Windows" on page 117.

To launch the installer from the BSM product installation disk:

- 1** Run the **setup.exe** file in the root directory of the installation disk. The Diagnostics Setup program begins and displays the installation menu page.
- 2** From the installation menu page, select **Diagnostics/TransactionVision Agent for Java** to launch the installer.

Running the Installation on Windows

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

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

To install the Java Agent on a Windows machine:

- 1** Accept the end user license agreement.
Read the agreement and select **I accept the terms of the license agreement**.
Click **Next** to proceed.
- 2** Specify the location where you want to install the agent.

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

Directory names must contain English characters only.

Click **Next** to proceed.

3 Review the summary information.

The installation directory and size requirement are listed.

Click **Next** to proceed.

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

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

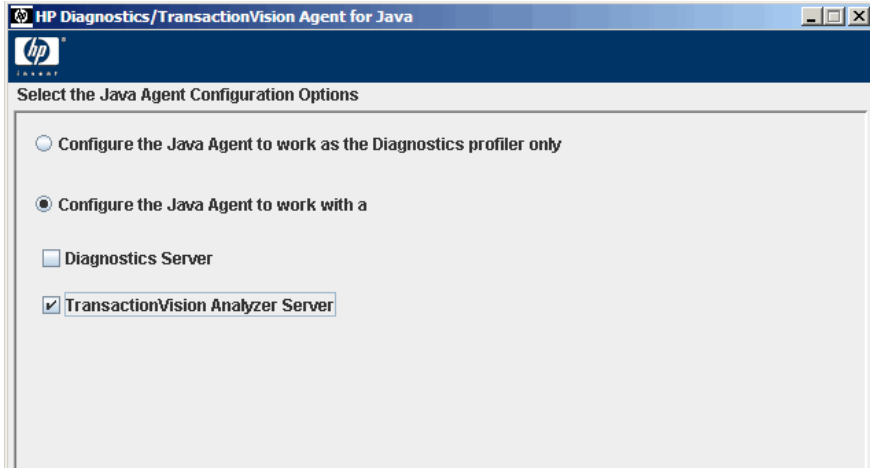
Configuring the Java Agent on Windows

You can configure the Java Agent to work as a TransactionVision Java Agent, a Diagnostics Java Probe, or both. To perform this configuration, you use the Java Agent Setup Module.

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

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

1 Specify the TransactionVision Analyzer Server option:



If you are configuring the Java Agent as a Diagnostics Profiler only, see “Configuring the Java Agent as a Profiler Only” in the *HP Diagnostics Installation and Configuration Guide*. If you are configuring the Java Agent to work as a J2EE Probe with a Diagnostics Server, See “Configuring the Probe to Work with a Diagnostics Server” in the *HP Diagnostics Installation and Configuration Guide*.

Click **Next** to proceed.

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



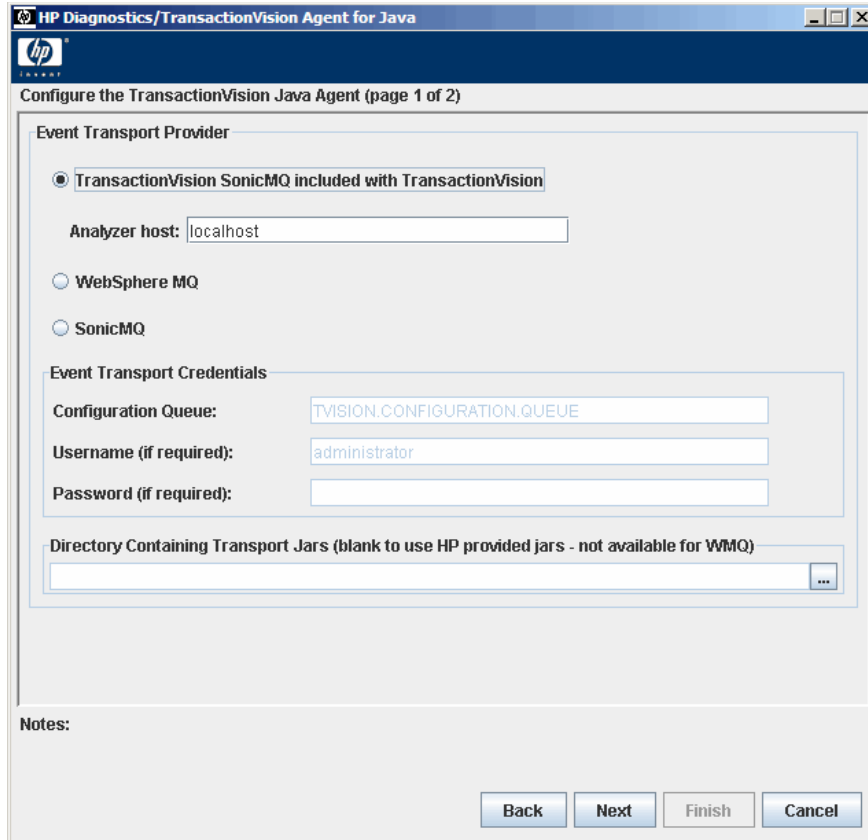
- For the Java Agent name, enter a name that uniquely identifies the agent within TransactionVision. The following characters can be used in the name: -, _, and all alphanumeric characters. The agent name is assigned to be the Java Agent name.

When assigning a name to an agent, choose a name that will help you recognize the application that the agent is monitoring, and the type of Java Agent.

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

Click **Next** to proceed.

3 Choose the event transport provider and specify the credentials.



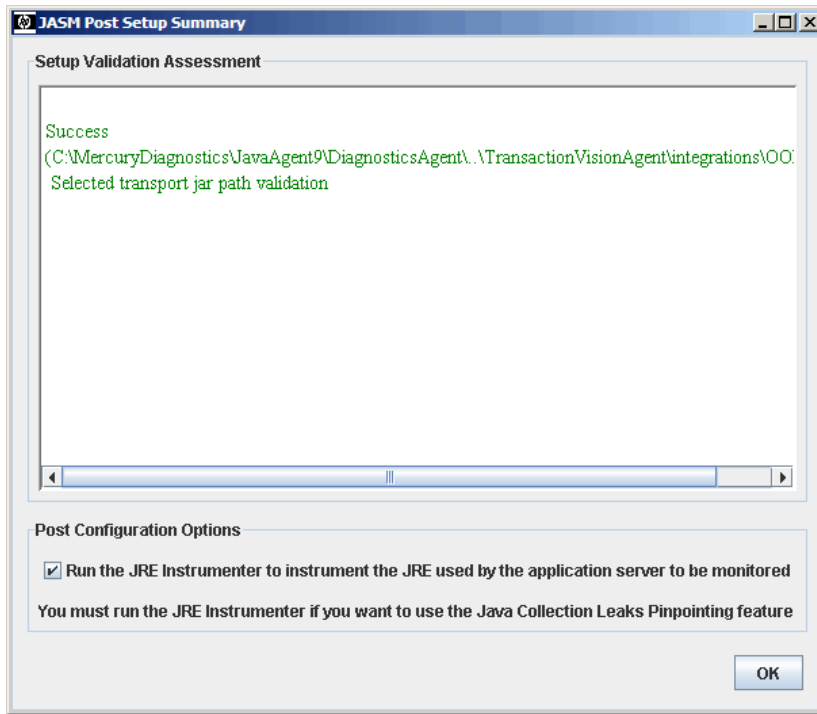
- To use the SonicMQ event transport provider that is included with the TransactionVision, specify the host name of the Processing Server that contains the Analyzer to which this agent will connect.

- If there is a firewall between the Processing Server and Java Agent, you must open port 21104 and 21111 so that the Java Agent can connect to these two ports on the Processing Server host.
- Enter the configuration queue name for your event transport provider. If you are using the built-in SonicMQ event transport provider, the default configuration queue name is `TVISION.CONFIGURATION.QUEUE` however this may have been set to a different value.
- 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.
- 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 any Java files.
- Click **Finish** to continue.

Post Configuration Options

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

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



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

You can choose to run the JRE Instrumenter automatically by checking **Run the JRE Instrumenter to instrument the JRE used by the application server to be monitored**. By default, the JRE Instrumenter option is not selected.

If the application to be monitored is using a JRE version prior to 1.5, you must select this check box or run the JRE Instrumenter manually. For a complete description of the JRE Instrumenter and how to run it manually, see "Task 3: Run the JRE Instrumenter for each Target JRE" on page 128.

Installing and Configuring the Java Agent on UNIX

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

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

Downloading the Installer on UNIX

You can download the Java Agent installer from the HP Software web site, from the Diagnostics or TransactionVision product disk, or from the Downloads page in BSM.

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

To copy the installer from the product installation disk:

- 1 From the <HP TransactionVision Installation Disk>/
TransactionVision_Installers directory, copy the installer
HPDiagTVJavaAgt_<version>_<platform>.bin to the machine where the
TransactionVision Agent is to be installed.
- 2 Continue with "Running the Installation on UNIX" on page 123.

To download the installer from the BSM Downloads page:

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

Running the Installation on UNIX

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

To install the Java Agent on a UNIX machine:

1 Run the installer.

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

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

```
./JavaAgentSetup_<platform>_<version>.bin -console
```

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

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

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

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

```
./JavaAgentSetup_<platform>_<version>.bin
```

The installer displays the same screens that are displayed for the Windows installer, as shown in "Installing and Configuring the Java Agent on Windows" on page 116.

2 Accept the end user license agreement.

The end user software license agreement is displayed.

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

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

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

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

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

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

4 Verify the installation location.

The installation location and the estimated size are listed.

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

The installation may take a few minutes.

The Java Agent Setup Module is launched.

Configuring the Java Agent on UNIX

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

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

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

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

Configuring the Java Agent on UNIX in Graphical Mode

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

1 Set up the graphical interface.

Export your display back to your terminal; `xhost + #` allows you to display the UI on the console:

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

2 Run the Java Agent Setup Module.

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

The Java Agent Setup Module displays the same screens that are displayed for the Windows Java Agent Setup Module, as shown in "Configuring the Java Agent on Windows" on page 118.

Configuring the Java Agent on UNIX in Console Mode

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

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

```

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

```

- ▶ Enter O (capital letter O) to skip the Diagnostics Profile Only option and again to skip the Diagnostics Java Agent working with an HP Diagnostics Server option.
- ▶ If you want to configure the Java Agent to work as both a J2EE Probe with a Diagnostics Server and as a TransactionVision Java Agent, enter X for both options and continue to step 2.

Press **Enter** to continue.

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

```

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

```

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

When assigning a name to an agent, choose a name that will help you recognize the application that the agent is monitoring, and the type of Java Agent.

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

Press **Enter** to continue.

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

- 4 Instrument the JRE.

```

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

```

Enter the instrumentation commands as described in "Running the JRE Instrumenter on a UNIX Machine" on page 134.

- 5 Enter 0 (zero) to complete the setup.

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

- 6 Modify the startup script for the application server so that the probe is started together with the monitored application. See "Task 6: Enable the Java Agent in Applications to Be Monitored" on page 139.
- 7 Verify the Java Agent installation.

Task 3: Run the JRE Instrumenter for each Target JRE

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

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

Notes:

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

This section includes:

- ▶ "JRE Instrumenter Processing" on page 129

- ▶ "Running the JRE Instrumenter Manually" on page 129

JRE Instrumenter Processing

The JRE Instrumenter performs the following functions:

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

Running the JRE Instrumenter Manually

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

This section includes:

- ▶ "Running the JRE Instrumenter on a Windows Machine" on page 129
- ▶ "Running the JRE Instrumenter on a UNIX Machine" on page 134

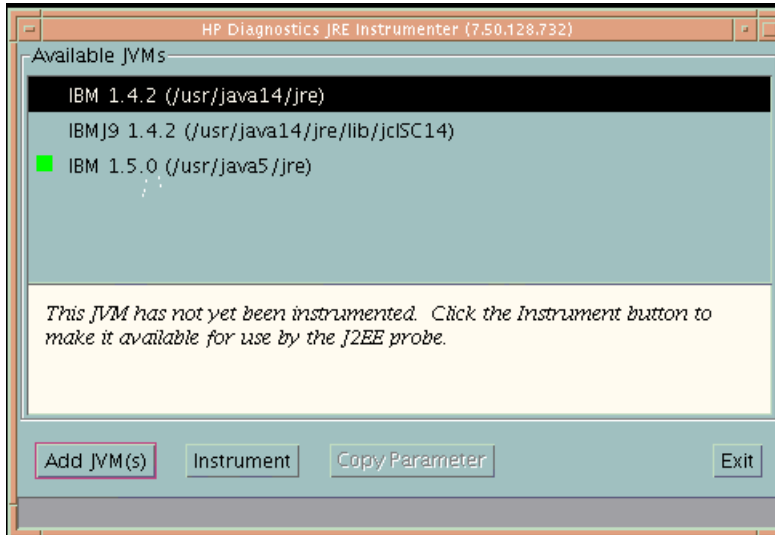
Running the JRE Instrumenter on a Windows Machine

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

Starting the JRE Instrumenter on a Windows Machine

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

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



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

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

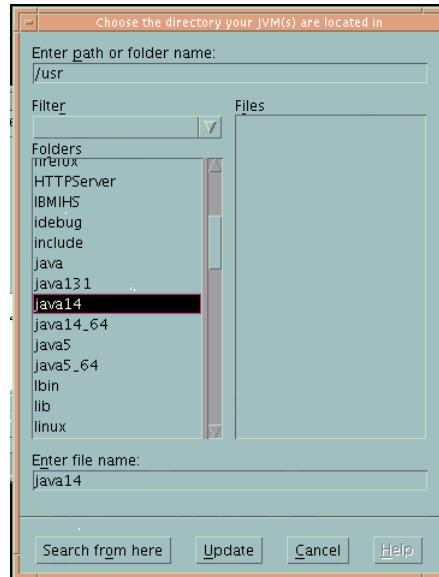
- ▶ If the JVM that you want to instrument is not listed in the Available JVMs list in the dialog, you can add JVMs to the list as described in "To add JVMs to the available JVMs list, perform the following steps:" on page 131.
- ▶ If the JVM that you want to instrument is listed but has not yet been instrumented, you can instrument the JVM as described in "Instrumenting a Selected JVM" on page 133.
- ▶ If the JVM that you want to instrument is listed and has already be instrumented, you can copy the JVM parameter that must be inserted into the start script for the JVM to activate the Probe's monitoring as described in "Including the JVM Parameter in the Application Server's Startup Script" on page 133.

- If you have finished using the JRE Instrumenter you can click Exit to close the JRE Instrumentation Tool.

To add JVMs to the available JVMs list, perform the following steps:

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

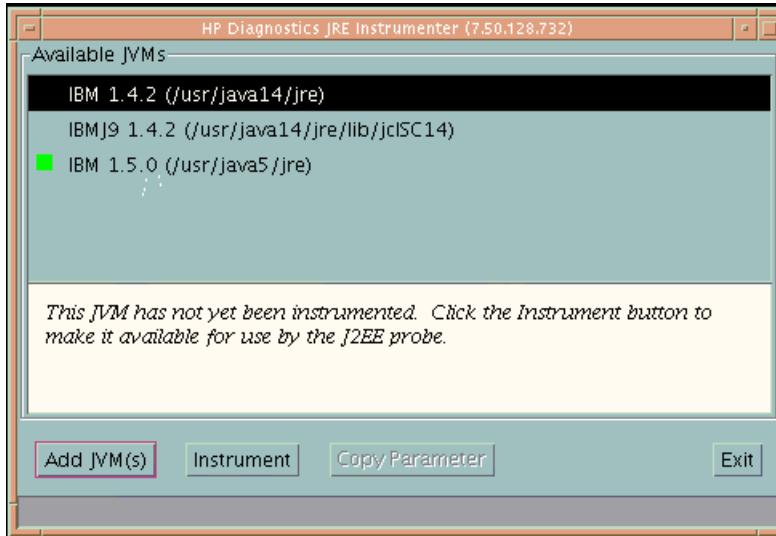
The Instrumenter displays the Choose the Directory dialog box.



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

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

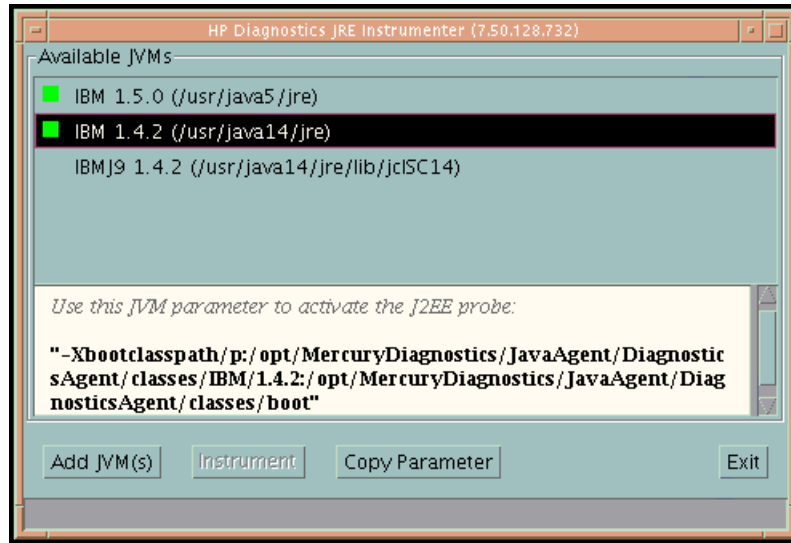
As the tool locates JVMs, it lists them in the **Available JVMs** list.



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

Instrumenting a Selected JVM

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



The JRE Instrumenter displays the JVM parameter in the box below the Available JVMs list, which must be used when the application server is started.

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

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

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

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

Running the JRE Instrumenter on a UNIX Machine

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

The JRE Instrumenter screens that are displayed in a graphical mode are the same as those documented for Windows installer in "Running the JRE Instrumenter on a Windows Machine" on page 129.

Starting the JRE Instrumenter on a UNIX Machine

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

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

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

Instrumenter Function	Description
<code>jreinstrumenter -l</code>	Display the list of known JVMs. See "Displaying the List of Instrumented JVMs" on page 135 for details.
<code>jreinstrumenter -a DIR</code>	Look for JVMs below the DIR directory. See "Adding JVMs to the Available JVMs List" on page 135 for details.

Instrumenter Function	Description
jreinstrumenter -i JVM_DIR	Instrument the JVM in JVM_DIR. See "Instrumenting a Listed JVM" on page 136 for details.
jreinstrumenter -b JVM_DIR	Instrument the JVM in JVM_DIR and put the ClassLoader in <java_agent_install_dir>/ DiagnosticsAgent/classes/boot . See "Instrumenting a Listed JVM" on page 136 for details.

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

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

Displaying the List of Instrumented JVMs

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

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

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

Adding JVMs to the Available JVMs List

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

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

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

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

Instrumenting a Listed JVM

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

- Explicit path to ClassLoader

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

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

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

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

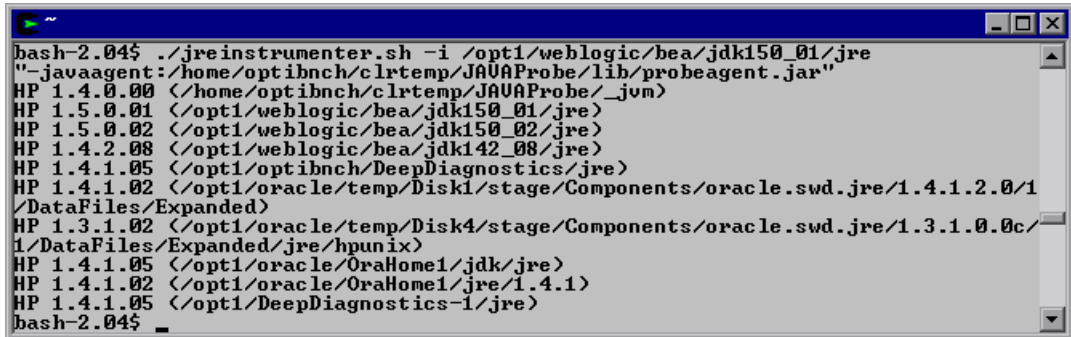
- Generic path to ClassLoader

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

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

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

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



```

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

```

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

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

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

Task 4: Set Up Messaging Queues

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

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

TransactionVision uses three message queues: the configuration queue, the event queue, and the exception queue. The default name of these queues are TVISION.CONFIGURATION.QUEUE, TVISION.EVENT.QUEUE, and TVISION.EXCEPTION.QUEUE, respectively.

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

- ▶ "IBM WebSphere MQ" on page 138
- ▶ "Progress SonicMQ" on page 138

IBM WebSphere MQ

You create the queues on your queue managers using the IBM WebSphere MQ runmqsc utility program or the MQ Explorer graphical user interface that is available on Windows and Linux. 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.

Progress SonicMQ

If you are using the builtin SonicMQ, the queues are created automatically for you.

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

Task 5: Manually Configure the Application Servers to Enable the Instrumentation

See *HP Diagnostics Installation and Configuration Guide*.

Task 6: Enable the Java Agent in Applications to Be Monitored

If the application to be monitored is using a JRE version 1.5 or later, follow these steps to enable the Java agent to collect events from the application:

Enable Java Agent in Applications on Windows

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

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

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

C:\MercuryDiagnostics\JavaAgent.

- To enable Java Agent for application servers, update the application server startup scripts manually. Refer to your application server documentation.

Enable Java Agent in Applications on UNIX

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

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

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

JavaAgent.

- To enable Java Agent for application servers, update the application server startup scripts manually. Refer to your application server documentation.

Task 7: Set up Communication Links that Use the Agent

See "TransactionVision Administration" in *Using Transaction Management*.

Optional Task: Configuring Custom User Events

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

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

For example:

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

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

Optional Task: Silent Installation of the Java Agent

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

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

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

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

To generate a response file for the Java Agent installation:

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

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

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

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

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

► On Windows:

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

► On UNIX:

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

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

To perform a silent installation or configuration:

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

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

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

Followed by:

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

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

Uninstalling the Java Agent

To uninstall the Java Agent:

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

Or run `uninstaller.exe`, which is located in the `<probe_install_dir>_uninstdirectory`.

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

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

14

Installing and Configuring Agents on z/OS

This chapter includes:

- ▶ About Agents in the z/OS Environment on page 143
- ▶ Base Component Installation Summary on page 144
- ▶ Base Component Installation Procedure on page 145
- ▶ Basic Configuration Steps for the SLD Agents: CICS, WMQ Batch, WMQ IMS, and WMQ CICS on page 152
- ▶ Basic Configuration Steps for the SLM Agent: WMQ IMS Bridge on page 156

About Agents in the z/OS Environment

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

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

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

Base Component Installation Summary

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

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

Two options are available:

- ▶ Use the `tvinstall_<ver>_zos_zseries.bat` script found on the installation media to assist with the transfer, where `<ver>` is replaced by a three digit version-release identifier (recommended).
- ▶ Manually FTP the install files.

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

- ▶ Use the (TSORECV) Rexx script created by the `tvinstall_<ver>_zos_zseries.bat` script in Step 1 above to execute the TSO RECEIVE commands (recommended).
- ▶ Manually execute the needed TSO RECEIVE commands.

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

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

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

- 5** Review installation libraries.

Base Component Installation Procedure

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

Two options are available:

- "Option 1: Run the tvsinstall_<ver>_zos_zseries.bat script" on page 146
- "Option 2: Manually FTP the install files:" on page 147

File Name Conventions

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

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

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

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

where:

```
<filequal> = f1 | f2 | f3 | mcs
```

```
<prodcode> = SLD | SLM
```

```
<ver> = current version-release, for example 900
```

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

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

```
tvsldf1_900_zos_zseries.xmit
```

An ftp output dataset name might be:

```
TVISION.SLD900.F1
```

Option 1: Run the `tvinstall_<ver>_zos_zseries.bat` script

- 1** Logon to a workstation running Microsoft Windows that is capable of connecting to the target z/OS system using TCP/IP FTP.
- 2** Start an MS-Windows Command Prompt session (Start > All Programs > Accessories > Command Prompt). Adjust the windows properties, increasing its size and buffering, as necessary.
- 3** Navigate, using the change directory (`cd`) command, to the folder containing installation media.
- 4** Review the information that will be collected by the script by entering:
`tvinstall_<ver>_zos_zseries.bat /h | more`
- 5** Invoke `tvinstall_<ver>_zos_zseries.bat` with no parameters and respond to the prompts.

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

- 6** Carefully review the results of the FTP command output issued by `tvinstall_<ver>_zos_zseries.bat`.
 - ▶ There should be no errors.
 - ▶ Check for the existence of installation RELFILES with the high-level qualifiers specified on the target z/OS system.
 - ▶ Check for the existence of the Rexx script (TSORECV) in the PDS library specified.
 - ▶ If any datasets are missing or FTP errors were detected, they should be investigated, corrected, and the `tvinstall_<ver>_zos_zseries.bat` script re-executed.

Option 2: Manually FTP the install files:

Do not perform this step if you installed the files by using the `tvinstall_<ver>_zos_zseries.bat` script.

- 1** Logon to a workstation running Microsoft Windows that is capable of connecting to the target z/OS system using TCP/IP FTP.
- 2** Start an MS-Windows Command Prompt session (Start > All Programs > Accessories > Command Prompt). Adjust the windows properties, increasing its size and buffering, as necessary.
- 3** Navigate, using the change directory (`cd`) command, to the folder containing installation media.
- 4** Using a z/OS account with permissions sufficient to create datasets with the desired high level qualifier, initiate an FTP session with the target z/OS system, for instance:

```
> ftp hostname
```

- 5** Issue the following FTP sub-commands:

```
a ftp> quote site fixrecfm 80 lrecl=80 recfm=fb blksize=3120
vol=&custvol u=&custunit pri=45 sec=15 tr
```

where `&custvol` is a valid disk volser and `&custunit` is a valid disk device type or esoteric

```
b ftp> bin
```

```
c ftp> put tv<prodcode><filequal>_<ver>_zos_zseries.xmit
'<&hlq>.<prodcode><|ver>.<filequal>'
```

where

```
<prodcode> = slm | sld
```

```
<&hlq> = customer select high level qualifier
```

```
<filequal> = f1 | f2 | f3 | mcs
```

```
<ver> = current version such as 900
```

For example:

```
put tvsldf1_900_zos_zseries.xmit 'TVISION.SLD900.F1'
```

- d** Issue ftp put commands for any remaining product installation files with a filename suffix of fn or mcs.
- e** It is important that ftp command output be carefully examined for errors. If detected, the target dataset on the z/OS system may need to be deleted and the ftp put command re-executed.
- f** ftp> quit

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

Two options are available:

- ▶ Run the (TSORECV) Rexx script created by the tvinstall_<ver>_zos_zseries.bat in step "Transfer files from the installation media to the zSeries platform." on page 144.
- ▶ Manually execute the needed TSO RECEIVE commands

Option 1: Execute the (TSORECV) Rexx script

From a TSO command line or TSO/ISPF command line, execute the (TSORECV) Rexx script created in "Transfer files from the installation media to the zSeries platform." on page 144. This results in the execution of TSO RECEIVE commands for all z/OS target datasets specified.

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

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

Option 2: Manually execute the needed TSO RECEIVE commands

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

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

Step 3: Determine the Best Installation Approach

Two options are available:

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

Assess the deployment environment to help you determine the appropriate strategy.

Step 4: Tailor and Execute the Installation Jobs

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

Option 1: SMP/E installation jobs

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

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

SMP/E installation jobs:

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

Option 2: Non-SMP/E installation jobs

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

Non-SMP/E installation jobs:

Order	Job	Description
1.	SL_INSTL	
2.	SL_IVP	Installation verification

Step 5: Review the Installation Libraries

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

- ▶ Not shown in the above table are the installation SMP/E distribution libraries (DLIBS). These will not be populated until execution of SMP/E Accept processing.
- ▶ 'TVISION' is the default high level qualifier, yours may differ.

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

Depending upon the mainframe agent components licensed at your installation, it may be necessary to repeat steps 3-5 of the base component installation procedure using the second agent product code (SLM or SLD). Most customers are licensed for all agents, so if you have just completed the base installation steps using the SLD product code, repeat the steps using the SLM or vice versa.

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

- 1 Required for the CICS and WMQ CICS Agents - Update your CICS CSD file(s) with the required resource definitions for SLD Agents by customizing and running job SLDCICSD, found in DS#2 - Agent Installation JCL Samples. If your environment consists of multiple CICS regions with separate CSDs or different startup lists, repeat this step for each CICS region to be monitored by the Agent(s).

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

- 2 Required for the CICS and WMQ CICS Agents - Place the CICS and WMQ CICS Agent program load modules in a library within your DFHRPL concatenation. Either copy the modules into an existing library already in your DFHRPL concatenation or add the SSLDLOAD library to the DFHRPL concatenation. If you run multiple CICS regions with separate startup JCL, repeat this step for each CICS region to be monitored by the Agent. The CICS modules are:

CSQCAPX, SLDPCCX
SLDPCM
SLDPCPX
SLDPCSX, SLDPX1
SLDPDSX
SLDPFCX

SLDPICX
 SLDPPCX, SLDPQMD, SLDPQME, SLDPQMX
 SLDPPSX, SLDPSTX
 SLDPTCX
 SLDPTDX
 SLDPTSX, SLDPUXI, SLDPXADM SLDSXAD

- 3** Optional for CICS Agent - To automatically enable the CICS Agent's exit programs at CICS startup time, define SLDPCSX as a second phase PLTPI. Refer to CICS Resource Definition Guide, DFHPLT section. Add the definition to your existing DFHPLTxx. If a new PLT is defined, add references to it in the CICS System Initialization Table (SIT). Refer to CICS System Definition Guide, "Specifying CICS system initialization parameters." Repeat this step for each CICS region to be monitored by the Agent. (Note - if this optional step is not performed, CICS transaction SLDS can be manually executed to enable the Agents.)

Sample SIT entry:

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

Sample definition of PLT with the suffix entry:

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

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

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

Sample INITPARM Entry used by the WMQ-CICS Agent:

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

Corresponding WMQ CICS Agent Manager startup command:

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

- 9** Optional for WMQ CICS Agent - To automatically start the WMQ CICS Agent's infrastructure initialization and monitoring programs at CICS startup time, define SLDPQME as a second phase PLTPI. Refer to the CICS Resource Definition Guide, DFHPLT section. Add the definition to your existing DFHPLTxx. If a new PLT is defined, add references to it in the CICS System Initialization Table (SIT). Refer to CICS System Definition Guide, "Specifying CICS system initialization parameters." Repeat this step for each CICS region requiring a WMQ CICS Agent. (Note - if this optional step is not performed, CICS transaction SLQE can be manually executed to enable the WMQ CICS Agent.)

Sample SIT entry:

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

Sample definition of PLT with the suffix entry:

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

- 10** Required for all z/OS TransactionVision Agents - The userid (account) associated with TransactionVision Agent Manager address spaces must have an OMVS segment defined to it.
- 11** Starting with z/OS version 1.9, the default setting for system parmlib DIAGnn member statement VSM ALLOWUSERKEYCSA has changed to "NO." This parameter disallows User Key CSA storage for security reasons. In order for TransactionVision 9.00 agents to execute successfully under z/OS, the VSM ALLOWUSERKEYCSA parameter value must be set to "YES."

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

Basic Configuration Steps for the SLM Agent: WMQ IMS Bridge

- 1** Required - APF-authorize the TransactionVision library, &hlq.SSLMAUTH. Please refer to MVS Initialization and Tuning Reference, PROGxx or IEAAPFxx system parameters for additional details. (Note - library &hlq.SSLMLOAD should NOT be APF authorized.)
- 2** Required - Create appropriate agent configuration and event queues on the WebSphere MQ queue manager to be used for communication between the Agent Manager components and the TransactionVision Analyzer. Customize and run job SLDCRTQS in DS#2 - Agent Installation JCL. (If you have already performed this step as part of SLD agent configuration, you may bypass it here.)
- 3** Required - Customize the sample TransactionVision Manager startup procedure: TVISION (If you have already performed this step as part of SLD agent configuration, you may bypass it here.)
- 4** Required - Customize the sample TVISIONB startup procedure in thlqual.SSLMPROC and copy it to an appropriate PROCLIB. TVISIONB requires four startup parameters, which may be specified in the procedure or on the START command.
 - ▶ The QMGR parameter specifies the name of the WebSphere MQ queue manager to which TVISIONB must connect to access its configuration and event queues. Note that this queue manager is the one to which the Analyzer connects when establishing a communication link to the agent and not necessarily the queue manager(s) to which the WebSphere MQ-IMS bridge is connected. It must be the same queue manager used when defining the configuration and event queues during installation (see the sample job in thlqual.SSLMINST(SLMCRTQS)).
 - ▶ The MAXQ parameter specifies the maximum amount of storage, in megabytes, that TVISIONB will allocate for its buffer queue. See "The TVISIONB Buffer Queue" on page 187.

- ▶ The EDPROC parameter specifies the name of the procedure to start the TVISIOND address space. Copy the TVISIOND startup procedure to an appropriate PROCLIB
 - ▶ The IMSJOB parameter specifies the jobname of the IMS control region for the IMS system to be monitored.
- 5** Include the thlqual.SSLMAUTH in the STEPLIB concatenation for each IMS control region for which TransactionVision WebSphere MQ-IMS bridge monitoring is required or copy the DFSYIOE0 module to an existing
- ▶ (IMS only) TVISIONB requires one system linkage index (LX), which it reserves the first time it is started after an IPL and thereafter reuses. Refer to the IBM z/OS or z/OS MVS Initialization and Tuning Reference, IEASYSxx (System Parameter List) NSYSLX=nnn parameter.
- 6** Required for all z/OS TransactionVision Agents - The userid (account) associated with TransactionVision Agent Manager address spaces must have an OMVS segment defined to it.
- 7** Starting with z/OS version 1.9, the default setting for system parmlib DIAGnn member statement VSM ALLOWUSERKEYCSA has changed to "NO." This parameter disallows User Key CSA storage for security reasons. In order for TransactionVision 9.00 agents to execute successfully under z/OS, the VSM ALLOWUSERKEYCSA parameter value must be set to "YES."

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

15

z/OS Components–Operation and Customization

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

This chapter includes:

- ▶ Component Overview on page 160
- ▶ Component Hierarchy on page 162
- ▶ Architectural Overview on page 163
- ▶ Component Configurations, Single Agent and Multi-Agent on page 164
- ▶ Controlling the TransactionVision Manager on page 166
- ▶ Inquiring about Event Collection on page 172
- ▶ Controlling Agents (AgentEvent Collectors) on page 173
- ▶ Data Space Buffer Queue Considerations on page 177
- ▶ Stub Program Usage by the MQ-Batch, IMS, MQ-IMS Bridge Agents on page 179
- ▶ Using the WebSphere MQ-IMS Bridge Agent on page 184
- ▶ Guidelines for Agent Operation on page 191

Component Overview

All TransactionVision z/OS Agents are implemented as a combination of common base components: The TransactionVision Manager, one or more technology specific Agent Managers, and Agents (aka Agent Event Collectors). See the diagram in "Component Hierarchy" on page 162 showing the individual components and interactions.

TransactionVision Manager

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

Agent Managers

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

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

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

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

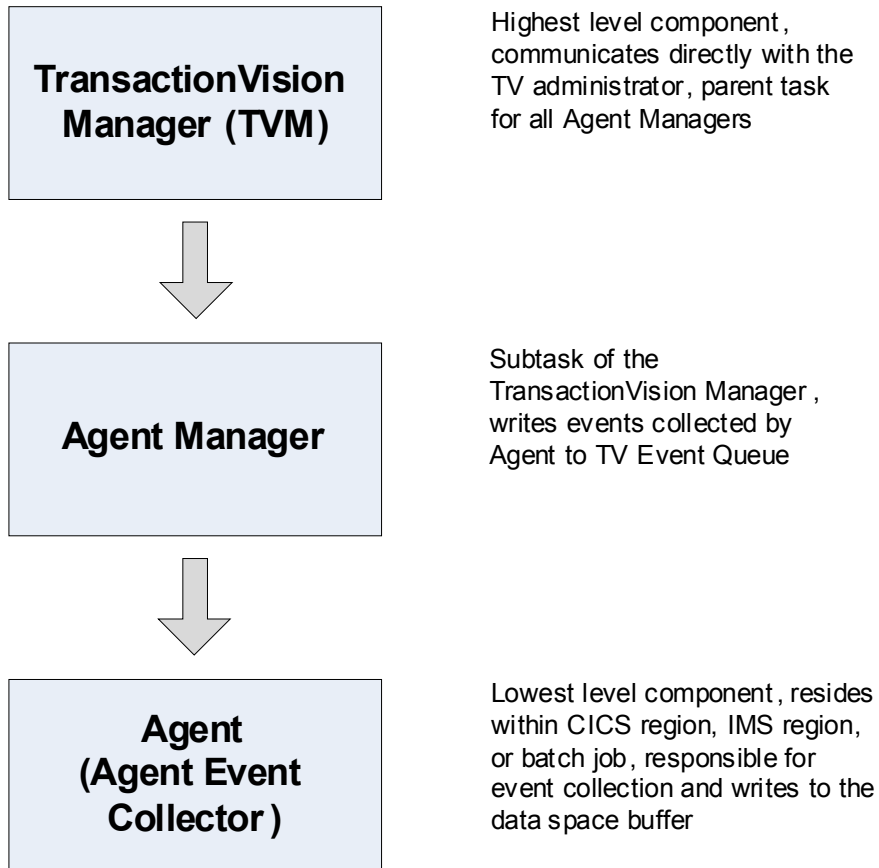
Agents or Agent Event Collectors

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

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

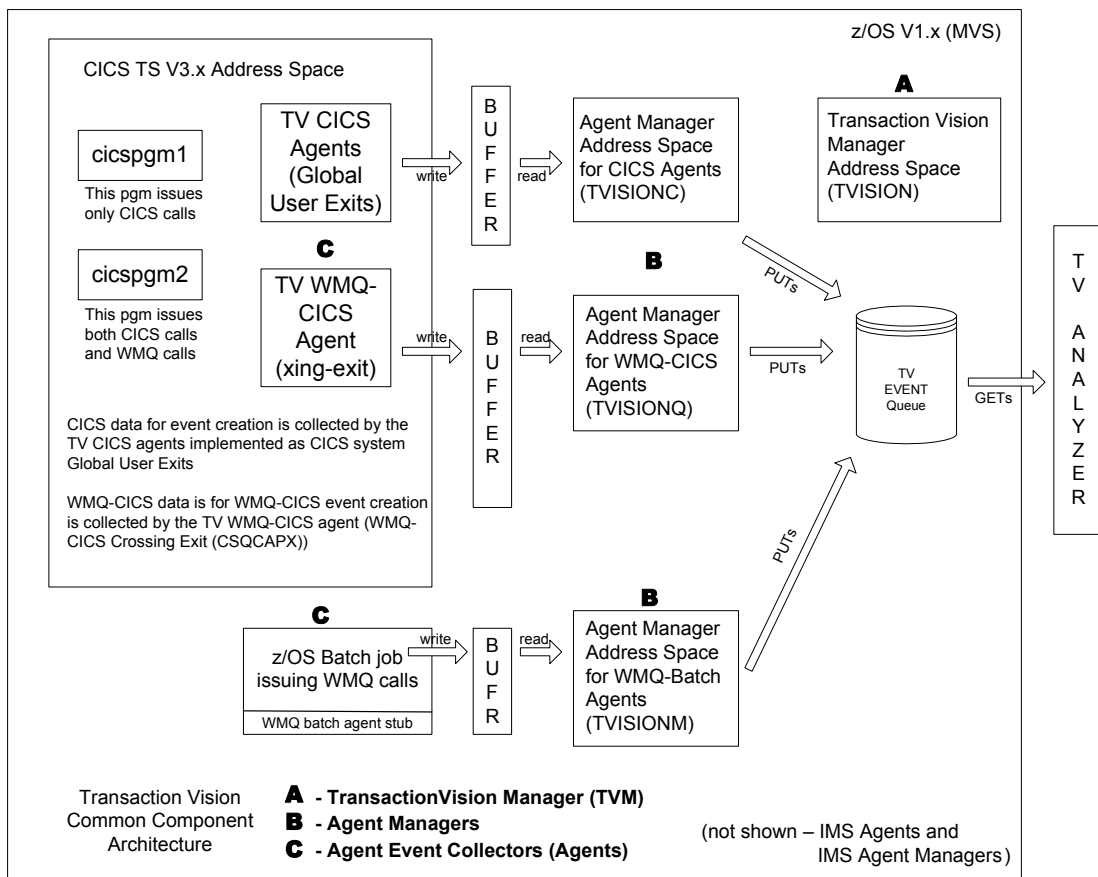
Component Hierarchy

The following diagram shows the component hierarchy.



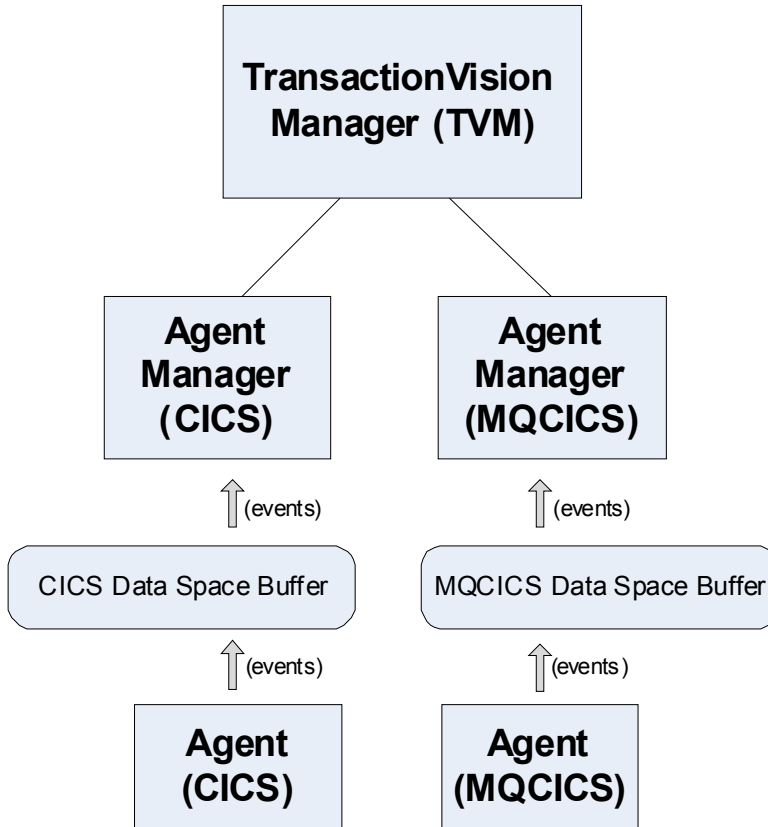
Architectural Overview

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

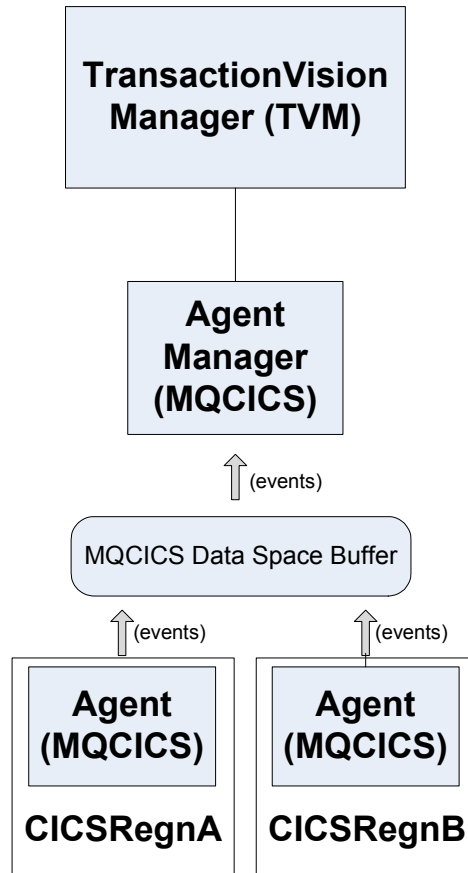


Component Configurations, Single Agent and Multi-Agent

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



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



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

Controlling the TransactionVision Manager

Starting the TransactionVision Manager

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

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.

tvld is a unique system id for an instance of TVM, consisting of four or fewer characters. (default value is 'TV01') Note that tvld 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

Starting an Agent Manager

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

where:

tvm_jobname is the MVS task/job of the TransactionVision Manager

START is the command name and is required.

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

sysid is required when the agenttype is CICS or MQCICS. When agenttype is CICS, then SYSID must specify the system id of the CICS region to be monitored. When agenttype is MQCICS, then SYSID identifies a customer defined agent SYSID. This should be the same value specified in the CICS INITPARM customization as found in Step 7 of section "Basic Configuration Steps for the SLD Agents: CICS, WMQ Batch, WMQ IMS, and WMQ CICS" on page 152.

imsid is required when the agenttype is MQIMS. imsid must specify the IMSID of the IMS system to be monitored.

drvproc is optional. drvproc 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 177 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 177 for more information.

Example 1:

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

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

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

Example 2:

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

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

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

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

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

Stopping an Agent Manager

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

where:

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

STOP is the command name and is required.

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

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

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

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

Example 1:

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

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

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

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

Example 2:

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

The CICS Agent Manager is stopped. The following messages will be displayed:

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

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

Inquiring about Event Collection

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

where

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

INQUIRE is the command name and is required.

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

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

Example:

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

The inquire statistics command will display the following:

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

Controlling Agents (AgentEvent Collectors)

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

Starting CICS AgentEvent Collectors

```
MODIFY cics_jobname, SLDS
```

where

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

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

Stopping CICS Agent Event Collectors

```
MODIFY cics_jobname, SLDP
```

where

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

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

Enabling the MQCICS AgentEvent Collectors

```
MODIFY cics_jobname, SLQE
```

where

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

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

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

Stopping (Disabling) MQCICS AgentEvent Collectors

```
MODIFY cics_jobname, SLQD
```

where

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

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

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

MQ CICS Agent Event Collector Admin User Interface

```
SLQA
```

where

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

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

```

Transaction Vision WMQ-CICS Agent Admin Interface
Copyright 2010, Hewlett-Packard Development Company, L.P.
-----
A - Activate WMQ-CICS Agent | Agent Mgr Status RUNNING
D - DeActivate WMQ-CICS Agent | Agent Status ACTIVE
R - Reset Session Stats | Agent Activated 20:02:37 6
X - Exit this Application | Session Started 20:03:04 6
_ < Selection | Current Time 20:03:35 6
-----
SESSION STATISTICS
-----
Tot Calls Processed = 0 Tot Calls Bypassed =
-----
MQFunc Call Count Avg Duration(secs) Avg Bufr Sz(bytes)
-----
MQOPEN 0 .000 0
MQCLOSE 0 .000 0
MQGET 0 .000 0
MQPUT 0 .000 0
MQPUTX 0 .000 0
MQINQ 0 .000 0
MQSET 0 .000 0
PF3=EXIT CLR=EXIT ENT=Refresh Msg: Statistics Display Refreshed

```


WMQ CICS Agent State Considerations

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

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

Data Space Buffer Queue Considerations

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

The buffer queue is configured as a number of queue blocks of a certain size. Both of these dimensions can optionally be specified as parameters on the start Agent manager commands. MAXQBLKS specifies the maximum number of queue blocks and QBLKSIZE specifies the size of each queue block in megabytes.

The maximum size of the data space used, in megabytes, is the product of MAXQBLKS and QBLKSIZE and must not exceed 2 GB. Default values in effect for SLD Agents are as follows:

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

The appropriate buffer size varies widely based on several factors: transaction throughput of the monitored application environment, the type and size of events collected, and the throughput capabilities of the Agent Manager. These variables will be discussed in more detail later, but before attempting laborious calculations using what may be mere guesses as your parameters, consider taking a trial and error approach. A generous allocation at startup can compensate for a lack of precision in estimating the event workload while not costing any extra overhead.

At Agent Manager, three queue blocks are allocated. When the TransactionVision application environment components are run, they store events into the first queue block and then to the next, etc. Concurrently, the Agent manager is retrieving events from the buffer queue in the same sequential order that they were stored (FIFO). The buffer queue management functions are invoked every hundredth of a second to check the state of the buffer queue. If the queue block currently receiving events is the last queue block allocated, an additional queue block is allocated provided that the total number allocated doesn't exceed the MAXQBLKS specification. Any queue block that has had all its events retrieved is freed,

reducing the total number of allocated blocks. Therefore, the size of the buffer queue expands and contracts as traffic dictates and, regardless of the maximum allowed, no more space is used than is required—beyond the minimum of three queue blocks. A generous allocation at startup can compensate for a lack of precision in estimating the event workload while not costing any extra overhead.

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

Stub Program Usage by the MQ-Batch, IMS, MQ-IMS Bridge Agents

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

On the z/OS batch and IMS platforms, the SLMLKSTB member of the sample procedure library thlqual.SSLMPROC contains a sample job to bind the agent to an WebSphere MQ application program. The WebSphere MQ batch stub section—CSQBSTUB, CSQBRSI, or CSQBRSTB for Batch or CSQQSTUB for IMS—is replaced in the application load module with the corresponding agent batch or IMS stub—SLMBSTUB, SLMBRSI, SLMBRSTB, or SLMQSTUB. After this bind, the application will invoke the agent instead of WebSphere MQ directly.

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

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

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

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

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

To modify SLMBCNFG, perform the following steps:

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

For more information about the WebSphere MQ-IMS bridge Agent, see "Using the WebSphere MQ-IMS Bridge Agent" on page 184.

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

If the program SLMCNFQ can be executed by the agent via an EXEC CICS LINK, the agent does so, passing SLMCNFQ a CICS comm area large enough to hold a configuration queue name (48 bytes). The comm area initially contains the default configuration queue name (TVISION.CONFIGURATION.QUEUE).

When executed by the agent, SLMCNFQ should write the desired configuration queue name to the comm area passed to it, and then return. Subsequently, the agent will read the comm area to retrieve the correct configuration queue name.

Note: The installation procedure for the z/OS CICS Agent does NOT create an SLMCNFQ program. For instructions on writing this program, see Writing SLMCNFQ, which follows.

If the attempt to execute the SLMCNFQ fails, the z/OS CICS WebSphere MQ Agent tries to load the program SLMBCNFG and gets the configuration queue name. For instructions on generating this program, see "To generate SLMBCNFG, follow these steps: " on page 182.

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

Note: To use a different configuration queue name for each CICS region, see "Using Separate Configuration Queues for Each CICS Region" on page 182.

Writing SLMCNFQ

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

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

To generate SLMBCNFG, follow these steps:

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

Using Separate Configuration Queues for Each CICS Region

If you have multiple CICS regions, you may want to use a different configuration queue name for each CICS region while sharing the Agent library among those CICS regions.

To specify a different queue name for each CICS region, perform the following steps:

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

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

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

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

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

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

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

Add an additional step before the actual CICS execution step:

```

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

```

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

```

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

```

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

```

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

```

Using the WebSphere MQ-IMS Bridge Agent

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

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

This section includes:

- "Agent Setup" on page 185
- "WebSphere MQ-IMS Bridge Agent Operation" on page 186
- "The TVISIONB Buffer Queue" on page 187
- "Event Data" on page 188
- "Data Collection Filters and Queries" on page 190

Agent Setup

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

- 1** Customize the sample TVISIONB startup procedure in `thlqual.SSLMPROC` and copy it to an appropriate PROCLIB. TVISIONB requires four startup parameters, which may be specified in the procedure or on the START command.
 - The QMGR parameter specifies the name of the WebSphere MQ queue manager to which TVISIONB must connect to access its configuration and event queues. Note that this queue manager is the one to which the Analyzer connects when establishing a communication link to the agent and not necessarily the queue manager(s) to which the WebSphere MQ-IMS bridge is connected. It must be the same queue manager used when defining the configuration and event queues during installation (see the sample job in `thlqual.SSLMINST(SLMCRTQS)`).
 - The MAXQ parameter specifies the maximum amount of storage, in megabytes, that TVISIONB will allocate for its buffer queue. Please refer to "The TVISIONB Buffer Queue" on page 187.
 - The EDPROC parameter specifies the name of the procedure to start the TVISIOND address space.
 - The IMSJOB parameter specifies the jobname of the IMS control region for the IMS system to be monitored.

- 2 Include the thlqual.SSLMAUTH in the STEPLIB concatenation for each IMS control region for which TransactionVision WebSphere MQ-IMS bridge monitoring is required or copy the DFSYIOE0 module to an existing qualifying library.

WebSphere MQ-IMS Bridge Agent Operation

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

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

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

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

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

TVISIONB will automatically start TVISIOND.

- 3 Request bridge monitoring from the TransactionVision UI/Job Server on a connected workstation. Please refer to the *TransactionVision User's Guide* for more information.

- 4 Ordinarily, the activity of the bridge agent is controlled from the TransactionVision UI/Job Server. However, you may disable the agent from the system console with the MODIFY command: F TVISIONB,DISABLE MQIMSBGD. 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 MQIMSBGD.
- 5 Stop the TVISIONB address space as follows: P TVISIONB. This will implicitly disable the agent; the exit routine continues running but does not attempt to send events to the TVISIONB. TVISIOND will automatically be stopped.

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

The TVISIONB Buffer Queue

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

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

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

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

Event Data

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

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

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

- Beans.xml
- IMSBridgeObject.xml

Beans.xml

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

```
<Module type="Context" name="EventModifierCtx">
  <!--
    This bean read MQObject definition for IMS bridge
    entry event from $TVISION_HOME/config/service/
    IMSBridgeObject.xml
  -->
  <Module type="Bean" class="com.bristol.tvision.services.analysis.eventmodifier
  .IMSBridgeEntryModifierBean"/>
</Module>
```

IMSBridgeObject.xml

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

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

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

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

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

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

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

Data Collection Filters and Queries

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

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

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

Guidelines for Agent Operation

- ▶ 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.

16

Installing and Configuring Agents on BEA Tuxedo

This chapter includes:

- ▶ Preparing for the Installation on page 193
- ▶ Running the Installation on page 194
- ▶ Rebinding the Tuxedo Agent on page 195
- ▶ Uninstalling Agents on page 195
- ▶ Configuring BEA Tuxedo Agents on page 196

Preparing for the Installation

You start the agent installation from the HP Business Service Management product installation disk or from the Downloads page in BSM.

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

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

Running the Installation

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

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

- 2 Log in as superuser:

```
su
```

- 3 Unzip and untar the packages for your platform; see "Preparing for the Installation" on page 193. For example:

```
gunzip HPTVTuxedoAgent_<version>_hppa.tgz
```

- 4 Enter the following command to begin the installation procedure:

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

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

The following TransactionVision packages are available for installation:

1. TransactionVision Analyzer
2. TransactionVision WebSphere MQ Agent
3. TransactionVision BEA Tuxedo Agent
4. TransactionVision User Event Agent

99. All of above
q. Quit install

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

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

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

```
3,4
```

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

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

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

Rebinding the Tuxedo Agent

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

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

Uninstalling Agents

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

- 1** Log in as superuser:

```
su
```

- 2** Enter the following command:

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

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

The following TransactionVision packages are available for installation:

1. TransactionVision Analyzer
2. TransactionVision WebSphere MQ Agent
3. TransactionVision BEA Tuxedo Agent
4. TransactionVision User Event Agent

99. All of above

q. Quit install

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

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

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

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

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

Configuring BEA Tuxedo Agents

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

The required configuration consists of the following:

- ▶ "Configure the Application Library to Load the Agent Library" on page 197
- ▶ "Set Required Environment Variables" on page 198
- ▶ "Setting the Agent Connection URL" on page 198

An optional configuration is:

► "Filter Sensitive Data" on page 199

Configure the Application Library to Load the Agent Library

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

Add the directory location of the Agent library to an environment variable setting on the computer where the monitored application runs before starting the application.

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

Platform	Environment Variable	Default Directory
Sun Solaris	LD_LIBRARY_PATH	/opt/HP/TransactionVision/lib
HP-UX	SHLIB_PATH	/opt/HP/TransactionVision/lib
IBM AIX	LIBPATH	/usr/lpp/HP/TransactionVision/lib

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

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

Platform	Environment Variable	Default Directory
Sun Solaris	LD_LIBRARY_PATH	/opt/HP/TransactionVision/lib64
HP-UX	SHLIB_PATH	/opt/HP/TransactionVision/lib64
IBM AIX	LIBPATH	/usr/lpp/HP/TransactionVision/lib64

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

On the HP-UX platform, you may have to use the `chatr` command to enable your application to pick up the agent library if the use of the `SHLIB_PATH` environment variable is not enabled or is not the first in the dynamic library search path for your application. You may use the `chatr` command to check the current settings of your application. In any case, make sure that `SHLIB_PATH` is enabled and is before the standard BEA Tuxedo library path. See Troubleshooting in *Using Transaction Management*.

Set Required Environment Variables

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

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

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

Setting the Agent Connection URL

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

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

Filter Sensitive Data

The BEA Tuxedo Agent is configured by default to collect payload (user) data from monitored calls. You may wish to disable this capability if the user data contains sensitive information. To do so, modify the BEA Tuxedo Agent property file (<TVISION_HOME>/config/TuxedoSensor.properties) and set the **collectUserData** option in the **static configuration** section to **no**.

17

Installing and Configuring the .NET Agent

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

The .NET Agent provides a low-overhead capture solution that works with HP Software's BSM applications. The .NET Agent captures events from a .NET application and sends the event metrics to the TransactionVision Analyzer or to the Diagnostics Server, or both.

This chapter includes:

- About .NET Agent Installer on page 202
- Installing the .NET Agent on page 202
- Configuring the .NET Agent on page 208
- Restarting IIS on page 215
- Determining the Version of the .NET Agent on page 216
- Uninstalling the .NET Agent on page 216
- SSL Configuration for .NET Agents on page 216

About .NET Agent Installer

During the .NET Agent installation the following setup and configuration is done for you:

- ▶ Discovery of ASP.NET applications. The installer attempts to automatically detect the ASP.NET applications on the system where the agent is installed.
- ▶ Default agent configuration.
 - ▶ The installer configures the agent to capture basic ASP.NET/ADO/WCF workload for each of the ASP.NET application detected. The agent configuration is controlled using the **probe_config.xml** file. See "Configuring the .NET Agent" on page 208.
 - ▶ Standard .NET application instrumentation. Default aspnet.points and adonet.points files are installed and enabled providing standard instrumentation to allow you to start monitoring ASP.NET applications. The points files control the workload that the agent will capture for the application.

The default points ASP.NET.points and Ado.Points need to be enabled to generate TransactionVision events. These are enabled by default.

To generate .NET Remoting Events you will also need Remoting.points and will have to setup the application for instrumentation.

- ▶ Optional configuration. Modify the agent configuration in the probe_config.xml file. See "Configuring the .NET Agent" on page 208.

Installing the .NET Agent

This section includes:

- ▶ "Preparing for the Installation" on page 203
- ▶ "Starting the Installation" on page 203
- ▶ "Running the Installation" on page 204
- ▶ "Installing the Agent to Work in a TransactionVision Environment" on page 206

Preparing for the Installation

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

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

Platform	All Platforms.
Memory	60 MB Additional RAM
Free Hard Disk Space	200 MB Additional Space
.NET Framework	1.1 or later

Caution: The .NET Agent includes a SOAP Extension Handler. Installing the .NET Agent may cause existing Web Applications that are using SOAP to restart.

Starting the Installation

You start the installer from the HP Business Service Management product installation disk or from the Downloads page in Business Availability Center.

To start the installation from the product installation disk:

1 Locate the installation package file for your platform:

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

2 Run the file in the root directory of the installation disk.

Continue with "Running the Installation" on page 204.

To start the installation from the Download Center - HP - BTO Software page:

- 1** Enter **TransactionVision** in the Keyword field, and **Trial software** in the Refine Search by resource type box.
- 2** Click the appropriate link to download the .NET Agent installer for your platform.

Continue with "Running the Installation" on page 204.

Running the Installation

After you have launched the installer, you are ready to begin the main installation procedure.

To install the .NET Agent on a Windows machine:

- 1** Accept the end user license agreement.

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

Click **Next** to proceed.

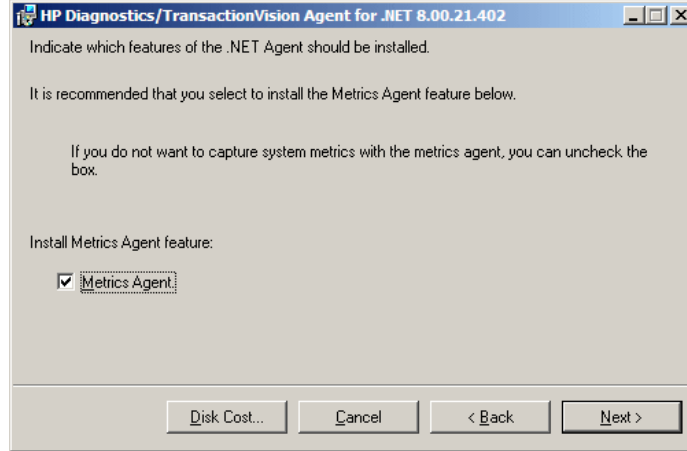
- 2** Provide the location where you want the Agent installed.

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

Accept the default directory or select a different location either by typing in the path to the installation directory into the **Installation HP Diagnostics/TransactionVision Agent for .NET to** box, or by clicking **Browse** to navigate to the installation directory.

Click **Next** to continue.

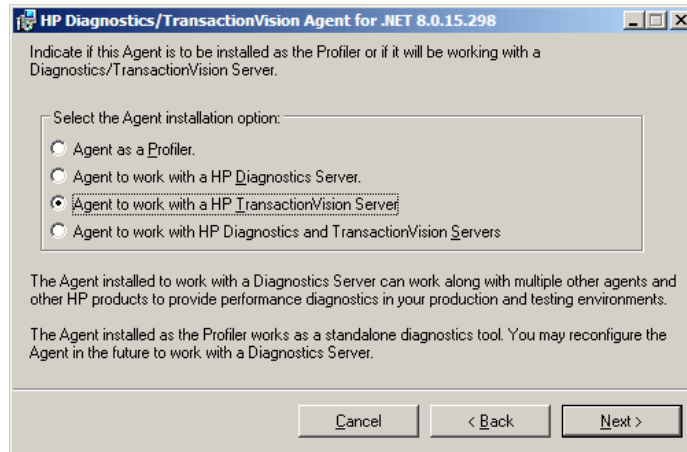
3 Select the .NET Agent features that you want to install.



To capture system metrics on the host machine without monitoring any applications, select **Metrics Agent** only.

If you want to check the amount of available disk space on the drives of the host, click **Disk Cost**. Use this functionality to make sure that there is enough room for the Agent installation.

4 Select whether you want to install the Agent as the Diagnostics Profiler only, as a probe reporting to a Diagnostics Server, or as an Agent reporting to the TransactionVision Analyzer.



- ▶ Select **Agent to work with an HP TransactionVision Server**.

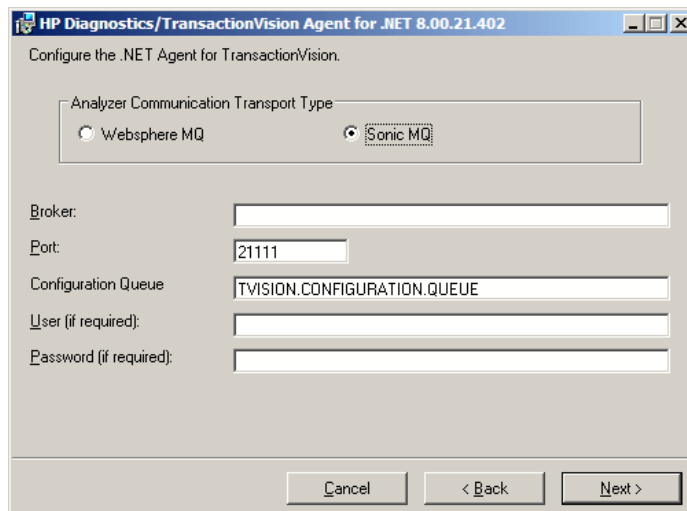
Click **Next** to continue.

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

Installing the Agent to Work in a TransactionVision Environment

If you are installing the Agent to work with in a TransactionVision environment, continue with the following procedure.

- 1 The Configure the .NET Agent for TransactionVision dialog appears.



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

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

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

3 For SonicMQ, enter the following:

Broker. Host name on which the Sonic broker is running. Typically this will be the Analyzer hostname.

Port. The port on which the broker communicates. By default, 21111.

Configuration Queue. Name of the configuration queue. By default, TVISION.CONFIGURATION.QUEUE.

User. User id if required by SonicMQ installation for connection. By default, no username is required.

Password. Password if required by SonicMQ installation for connection. This is in the obfuscated form created by using the PassGen utility. By default, no password is required. For more information about **PassGen**, see "Command-Line Utilities" in *Using Transaction Management*.

4 For WebSphere MQ, enter the following:

Host. The host on which the WebSphere MQ queue manager resides.

Port. Port number for WebSphere MQ queue manager.

Configuration Queue. Name of the configuration queue.

User. User id if required by WebSphere installation for connection.

Password. Password if required by the WebSphere MQ installation for connection. This is in the obfuscated form created by using the PassGen utility. For more information about **PassGen**, see "Command-Line Utilities" in *Using Transaction Management*.

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

Websphere MQ Q Manager. CCSID for WebSphere.

Click **Next** to continue.

5 The pre-installation summary dialog appears.

Click **Install** to proceed with the installation.

After the installation completes, you must restart IIS (see "Restarting IIS" on page 215). You can perform any custom configuration if desired as described in the section that follows.

Configuring the .NET Agent

The default configuration of the .NET Agent allows you to begin capturing performance metrics for the monitored application. The Agent's configuration can be customized to suit the configuration of your environment and the performance issues that you would like to diagnose.

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

Use the following elements:

- "<tv> element" on page 209
- "<timeskew> element" on page 210
- "<transport> element" on page 211
- "<logging> Element" on page 213
- "<modes> Element" on page 213

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

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

This section also includes these additional configuration topics:

- ".NET Remoting Client and Server Applications" on page 214
- "Enabling Correlation of .NET Events" on page 215

<tv> element**Purpose**

Configure the .NET Agent for use with TransactionVision.

Attributes

Attributes	Valid Values	Default	Description
eventthreads	number	3	(Read on startup) The number of threads spawned by the Agent to send events to the Analyzer.
eventthreadsleep	number	100	(Dynamic) The time in milliseconds the event thread sleeps after sending a message(event package).
eventmemorythreshold	number	250,000,000	(Dynamic) The memory consumed by the internal buffer(Q) after which the Agent will try and send the message on the application thread.
configthreadsleep	number	10,000	(Dynamic) The time in milliseconds the event thread sleeps after browsing the configuration queue.

Elements

Number of Occurrences	1 (one)
Parent Elements	ProbeConfig
Child Elements	transport, timeskew

Example

```
<tv eventthreads="3" eventthreadsleep="80"
eventmemorythreshold="25000000" configthreadsleep="10000" >
  <timeskew historysize="24" checkinterval="300000" latencythreshold="100"
  retrythreshold="8"/>
  <transport type="sonicmq"
  connectionstring="broker=myhost.mydomain.com;
  port=21111; user=; password=;
  configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
</tv>
```

<timeskew> element

Purpose

Calculates the time difference between the time server and the host on which the .NET Agent is running. The frequency of checking with the time server can be configured.

Attributes

Attributes	Valid Values	Default	Description
historysize	number	24	(Read on startup) number of time skew samples to store and compare for best sample.
checkinterval	number	300,000 ms.	(Dynamic) The time in milliseconds to wait before checking the time server for the skew time calculation.
latencythreshold	number	100 ms.	(Dynamic) The maximum time in milliseconds a reply from a time server can take for a valid time skew value.
retrythreshold	number	8	(Dynamic) Number of times to try when request to time server fails.

Elements

Number of Occurrences	1 (one)
Parent Elements	tv
Child Elements	none

Example

```
<timeskew historysize="24" checkinterval="300000" latencythreshold="100"
retrythreshold="8"/>
```

<transport> element

Purpose

Configure the events channel used by TransactionVision.

Attributes

Attributes	Valid Values	Default	Description
type	mqseries sonicmq	sonicmq	The event transport provider being used by the Agent.
connectionString	See below.		The connection information for the event transport provider.

connectionString Syntax when type=sonicmq

```
broker = <broker>; port = <port>; user = <user>; password = <password>;
configurationQueue = <configurationQueue>
```

Where:	Is:
broker	Host name on which the Sonic broker is running. Typically this will be the Analyzer hostname.
port	The port on which the broker communicates. By default, 21111.

Where:	Is:
user	User id if required by SonicMQ installation for connection. By default, no username is required.
password	Password if required by SonicMQ installation for connection. This is in the obfuscated form created by using the PassGen utility. By default, no password is required. For more information about PassGen , see ee "Command-Line Utilities" in <i>Using Transaction Management</i> .
configurationQueue	Name of the queue which has the configuration messages for the .NET TransactionVision Agent.

conectionString Syntax when type=mqseries

```
host= <host>; queuemanager=<queuemanager>; port= <port>; channel=,channel>
configurationQueue = <configurationQueue>
```

Where:	Is:
host	Host on which the TransactionVision configuration queue is hosted.
queuemanager	Name of the queuemanager.
port	MQSeries port on which the QueueManager communicates.
channel	MQSeries channel which will be used to communicate.
configurationQueue	Name of the queue which has the configuration messages for the .NET TransactionVision Agent.

Elements

Number of Occurrences	1 (one)
Parent Elements	tv
Child Elements	None

Example

For SonicMQ:

```
<transport type="sonicmq" connectionstring="broker=brokerHost;
  port=21111; user=; password=;
  configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
```

For MQ Series:

```
<transport type="mqseries" connectionstring="host=mqHost;
  queuemanager=; port=1414; channel=TRADING.CHL;
  configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
```

<logging> Element

The "TVDEBUG" tag can be specified for tracing TransactionVision specific code in the .NET Agent.

For example,

```
<logging level="TVDEBUG"/>
```

For more information about the **<logging>** element, see the *HP Diagnostics Installation and Configuration Guide*.

<modes> Element

The "tv" modes tag specifies the TransactionVision mode for the Agent.

For example,

```
<modes tv="true"/>
```

When this is the only mode specified, the Agent works in a "TV only" mode, that is, the Diagnostics profiler and the Diagnostics probe are disabled and only TransactionVision events are generated.

When other modes are specified, both TransactionVision and Diagnostics events are generated.

For more information about the **<modes>** element, see the *HP Diagnostics Installation and Configuration Guide*.

.NET Remoting Client and Server Applications

The following example shows the configuration changes required to monitor a .NET Remoting environment.

General changes to probe_config.xml:

```
<process name="TVRemotingClient">
  <points file="Remoting.points"/>
  <points file="TVRemotingClient.points"/>
  <modes tv="true"/>
</process>
```

To enable WCF instrumentation, the WCF.points file must be included for the processes that you want to collect events for.

```
<process name="WCFService">
  <points file="SimpleConsoleHost.points"/>
  <modes tv="true"/>
</process>
```

By default the ASP.NET process is configured for WCF instrumentation by the inclusion of the WCF.points file in the probe_config.xml:

```
<process enablealldomains="true" name="ASP.NET">
  <points file="ASP.NET.points"/>
  <points file="ADO.points"/>
  <points file="WCF.points"/>
</process>
```

To turn off WCF instrumentation for the ASP.NET process, remove the WCF.points file for this process as follows:

```
<process enablealldomains="true" name="ASP.NET">
  <points file="ASP.NET.points"/>
  <points file="ADO.points"/>
</process>
```

Enabling Correlation of .NET Events

By default, .NET Events are not correlated. To enable the user correlation bean, make the following changes to the configuration on the Analyzer host:

- uncomment the .NET Agent section in **EventCorrelationDefinition.xml** file
- uncomment the following tag in **Beans.xml**:

```
<Attribute name="UserCorrelationBean"
value="com.bristol.tvision.services.analysis.eventanalysis.XMLRuleCorrelationB
ean"/>
```

For more information about the **EventCorrelationDefinition.xml** file, see the *TransactionVision Advanced Customization Guide*.

Restarting IIS

After installing the .NET agent and modifying the configuration or creating custom instrumentation, as needed, restart IIS before using the .NET Agent with ASP.NET applications.

To restart IIS from the command line or from the Start > Run menu, type:

```
iisreset
```

This command restarts the Web publishing service but does not immediately start the agent. The next time that a Web page in the application is requested, the agent is started, the applications are instrumented, and the agent reads the Configuration Queue Messages from the Analyzer.

Note: ASP.NET automatically restarts applications under various circumstances, including when it has detected that applications have been redeployed, or when applications are exceeding the configured resource thresholds.

Determining the Version of the .NET Agent

When you request support, it is useful to know the version of the components that you have installed.

To determine the version of the .NET Agent:

- ▶ Right-click the file `<.net_agent_install_dir>\bin\HP.Profiler.dll`, and display the component version information by selecting **Properties** from the menu.

Uninstalling the .NET Agent

To uninstall the .NET Agent, follow these steps:

- 1** Stop all Web applications that are using SOAP.
- 2** Use the **Add/Remove Programs** utility in Windows.
Remove the HP TransactionVision/Diagnostics Agent for .NET program.
- 3** Restart the Web applications.

SSL Configuration for .NET Agents

If the .NET Agent is using SonicMQ for the messaging middleware, SSL can be enabled. See the *HP Business Service Management Hardening Guide* PDF.

18

Installing and Configuring agents on NonStop TMF

This chapter includes:

- About the NonStop TMF Agent on page 218
- Preparing for the Installation on page 218
- Installing the NonStop TMF Agent on page 219
- Startup/Shutdown on page 220
- Configuring the NonStop TMF Agent on page 221
- Uninstalling the NonStop TMF Agent on page 222

About the NonStop TMF Agent

All audited transactions on HP NonStop Guardian systems are logged to audit trail files. This include every kind of database access from indexed (b-tree) files to SQL databases. The product that generates the audited trail data is called Transaction Monitoring Facility (TMF), and because TMF protects any audited files on the NonStop system, it is the repository of all changes. To support On Line Transaction Processing (OLTP) applications, TMF can monitor thousands of complex transactions sent by hundreds of users to the audited trail database. TMF also provides transaction protection, database consistency, and database recovery critical in high-volume transaction processing.

The NonStop TMF Agent monitors the TMF audited trail database for changes (adds, deletes, and modifies) on Enscribe databases. The NonStop TMF Agent does not monitor changes to SQL relational databases, only Enscribe file monitoring is supported.

Because TMF records all audited changes to the audited trail database, the NonStop TMF Agent is able to read the audited trail records of interest and create TransactionVision user events which are forwarded to the TransactionVision Analyzer for analysis.

Preparing for the Installation

You start the agent installation from the HP Business Service Management product installation disk or from the Downloads page in BSM.

The following table shows the installation file names for the NonStop TMF packages for each supported platform:

Platform	Files
Guardian	HPTVTMFAgent_<version>_ns.zip

Installing the NonStop TMF Agent

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

- 1 Unzip the install package to a temporary directory.
- 2 Open FTPSCRIPT.txt and make the following changes:
 - <NonStop Username> to a valid NonStop user.
 - <NonStop Userpassword> to the user's password.
 - <NonStop destination volume.subvolume> to the destination volume and subvolume.
- 3 FTP using the modified FTP script.

For example:

```
ftp -s:FTPSCRIPT.txt <NonStop host DNS/IP>
```

- 4 Logon to the NonStop host as super.super and go to the temporary installation \$volume.subvol.
- 5 Run the "INSTALL" TACL macro and provide the destination installation volume/subvolume, and the host DNS name or IP address. For example:

```
135> run install
This program collects configuration information in order to set up the TransactionVision
sensor for the NonStop environment. This includes:
- Installation volume and subvolume where to install the sensor
- Location of the TransactionVision Analyzer Configuration Service

You will be prompted to input required configuration parameters.
If the previous value is provided in (), pressing <Enter> will set
the parameter to the previous value.

Enter the installation volume (): $data02.tvision

Enter the analyzer configuration service DNS or IP address (): 15.178.196.101

Sensor installation complete!
```

Startup/Shutdown

The NonStop TMF Agent has three TACL macros that are used for startup and shutdown:

- **COLDMON**. Starts the agent for the first time.
- **STRTMON**. Starts the agent after it has been stopped.
- **STOPMON**. Stops the agent.

COLDMON

This macro purges all temporary data used by the agent (queued events) and establishes the first audit file to begin looking for events that meet the User Event filter criteria.

Start the cold start macro, by going into the \$volume.subvol where the NonStop Agent is installed, and run COLDMON.

After starting COLDMON you are prompted to input the audit trail file to begin searching for events.

For example:

```
Select TMF Audittrail file to commence scan from:  
File 1: $AUDIT.ZTMFAT.AA000006  
File 2: $AUDIT.ZTMFAT.AA000007  
File 3: $AUDIT.ZTMFAT.AA000008  
Select number from list above :
```

If you know you would like the agent to begin searching in a specific audit trail file, input that file, otherwise, simply begin with the first file displayed (select '1').

The agent begins searching for events that fit the filter criteria up to the present. The first time the agent is started; there will not be any events to collect until a filter is configured. The agent will scan through the audit files, and wait for new audit trail records.

STRTMON

When the agent is stopped, it saves the last position of the record in the audit trail file it was reading. That way, when it is restarted, it will start scanning for events from the position where it let off. To start the NonStop TMF Agent, after it has been stopped, run the TACL macro STRTMON, for example:

```
run STRTMON
```

STOPMON

To stop the NonStop Agent, run the TACL macro STOPMON, for example:

```
run STOPMON
```

Configuring the NonStop TMF Agent

The NonStop TMF Agent is configured from two sources:

- ▶ The INSTALL TACL macro, which sets up values in the startup macros, (COLDMON/STRTMON).
- ▶ Data received from the Analyzer.

The TACL INSTALL macro sets all required values in the COLDMON/STRTMON macros so in most cases users do not need to make any additional changes to the startup macros. After installation of the agent, the agent will work with all of the values set by the INSTALL macro.

The NonStop Agent reads the Analyzer configuration service to determine, among other things, transport details for sending events and what data collection filtering to perform. On the Analyzer, transport details are part of the Communication Link configuration. The data collection filtering information comes from the Data Collection Filter configuration. See Communication Links in *Using Transaction Management*.

Uninstalling the NonStop TMF Agent

To uninstall the NonStop TMF Agent, follow these steps:

- 1 Stop the agent by running STOPMON.
- 2 Remove all files in the temporary installation directory.
- 3 Remove all files in the installation volume/subvolume.

19

Configuring WebSphere MQ Agents

TransactionVision provides two types of WebSphere MQ Agents: WebSphere MQ Agent library and WebSphere MQ API Exit Agents. This chapter describes configuring both types of agents on Windows, UNIX, and i5/OS platforms.

For information about configuring this agent on z/OS, see Chapter 15, "z/OS Components—Operation and Customization."

This chapter includes:

- Configuring the WebSphere MQ Agent Library on page 223
- Configuring the WebSphere MQ API Exit Agent on page 229
- WebSphere MQ Agents and FASTPATH_BINDING on page 237
- Using Agents with WebSphere MQ Samples on page 238
- WebSphere MQ Client Application Monitoring on page 238

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 224
- "Configuring Agent Logging" on page 228
- "Setting the Configuration Queue Name" on page 228
- "Configuring the WebSphere MQ Messaging System Provider" on page 229

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

UNIX and Windows Platforms

On UNIX and Windows platforms, you must add the directory location of the Agent library to an environment variable setting on the computer where the monitored application runs before starting the application.

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

Platform	Environment Variable	Default Directory
Windows	PATH	C:\Program Files\HP\TransactionVision\lib
Sun Solaris	LD_LIBRARY_PATH	/opt/HP/TransactionVision/lib
HP-UX	SHLIB_PATH	/opt//HP/TransactionVision/lib

Platform	Environment Variable	Default Directory
IBM AIX	LIBPATH	/usr/lpp/HP/TransactionVision/lib
RedHat Linux	LD_LIBRARY_PATH	/opt/HP/TransactionVision/lib

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

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

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

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

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

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

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

On UNIX platforms:

```
installation_directory/wmqsensor application_command_line
```

On Windows platforms:

```
installation_directory\wmqsensor application_command_line
```

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

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

z/OS CICS

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

i5/OS Platforms

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

TransactionVision also provides two utility service programs:

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

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

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

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

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

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

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

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

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

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

Configuring 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. Refer to Chapter 21, "Configuring Agent Logging", for details on WebSphere MQ Agent logging on UNIX platforms.

Setting the Configuration Queue Name

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

UNIX, Windows, and i5/OS

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

Setting the Configuration Queue Check Interval

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

Configuring the WebSphere MQ Messaging System Provider

TransactionVision uses queues for the communication between the Analyzers and the agents. You need at least three queues: the configuration queue, the event queue, and the exception queue. The default name of these queues are `TVISION.CONFIGURATION.QUEUE`, `TVISION.EVENT.QUEUE`, and `TVISION.EXCEPTION.QUEUE`, respectively. You must set up these queues on your message system provider in a vendor-specific way. See "Communication Links" in *Using Transaction Management*.

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

Configuring the WebSphere MQ API Exit Agent

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

- `tvisionapiexit`
- `tvisionapiexit_r`

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

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

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

- ▶ There is no need to disable FASTPATH_BINDING (see "WebSphere MQ Agents and FASTPATH_BINDING" on page 237 chapter for more information).
- ▶ The completion and reason codes for MQDISC calls are not collected by the API Exit Agent and are set to fixed values of MQCC_OK and MQRC_NONE, respectively. The event time for MQDISC events is set to the before-MQDISC function invocation time.
- ▶ The API Exit Agent collects some TransactionVision internal events generated from WebSphere MQ (WMQ) calls made by the Analyzer, and also internal WMQ events from Java Agents using a client connection to the listener.

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

This section includes:

- ▶ "Configuring the API Exit Agent on Distributed and i5/OS Platforms" on page 231
- ▶ "Configuring the API Exit Agent on Windows Platforms" on page 234
- ▶ "Identifying Programs to Monitor" on page 235
- ▶ "Discarding WebSphere MQ Events on TransactionVision Queues" on page 237

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

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

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

Configuring Symbolic Links to the WebSphere MQ API Exit Agent

Because TransactionVision supports both 32-bit and 64-bit versions of WebSphere MQ, you must create symbolic links to the appropriate shared object/DLLs.

For all versions of WebSphere MQ, use the following commands to link the agent:

```
In -s <TransactionVision Install Directory>/lib/tvisionapiexit /var/mqm/exits/tvisionapiexit
```

```
In -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:

```
In -s <TransactionVision Install Directory>/lib64/tvisionapiexit /var/mqm/  
exits64/tvisionapiexit
```

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

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

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

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

New Stanzas

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

Add the following stanzas to mqs.ini:

► **ApiExitCommon**

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

► **ApiExitTemplate**

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

Add the following stanza to `qm.ini`:

► **ApiExitLocal**

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

Stanza Attributes and Values

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

► **Name=TransactionVisionWMQSensor**

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

► **Function=TVisionEntryPoint**

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

► **Module=tvisionapiexit**

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

Caution: Do not specify a path; the module path is determined by the link command you used to link to the correct module (see "Configuring Symbolic Links to the WebSphere MQ API Exit Agent" on page 231). 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 237.

► Sequence=sequence_number

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

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

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

Configuring the API Exit Agent on Windows Platforms

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

A property page for the WebSphere MQ Services node, API Exits, describes the two types of API exit managed from this node: ApiExitCommon and ApiExitTemplate.

In the Exits property page for individual queue managers, you can update the ApiExitLocal. The Configure... buttons launch a dialog to manage the entries within each stanza. The dialog consists of a multi-column list of any API exits already defined in the appropriate stanza, with buttons to add, view, change the properties of, and remove exits.

See "Configuring the API Exit Agent on Distributed and i5/OS Platforms" on page 231 for a description of required stanzas and attribute values.

When configuring the API Exit Agent on Windows 64-bit, copy the 32-bit and 64-bit API Exit Agent libraries to the appropriate WebSphere MQ exit folders:

```
copy C:\Program Files\HP\TransactionVision\lib\tvisionapiexit C:\Program
Files\IBM\WebSphere MQ\exits
copy C:\Program Files\HP\TransactionVision\lib64\tvisionapiexit C:\Program
Files\IBM\WebSphere MQ\exits64
```

When referring to the API Exit library in the API Exit definition, use only the library name "tvisionapiexit" without the path. WebSphere MQ will load the appropriate API Exit from the default exits directories.

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

Identifying Programs to Monitor

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

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

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

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

If both `exit_sensor.allow` and `exit_sensor.deny` exist, the agent ignores `exit_sensor.deny`.

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

Note: When using the WebSphere MQ API Exit Agent, if the `exit_sensor.allow` file exists and is left empty, no programs will be monitored.

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

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

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

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

Discarding WebSphere MQ Events on TransactionVision Queues

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

► HP_TV*

“HP_TV” is the prefix for all TransactionVision queue objects.

► *HP_TV

“HP_TV is the suffix for all TransactionVision queue objects.

► *HP_TV*

All TransactionVision queue objects contain the string “HP_TV.”

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

If you require finer control over which queue objects to track, use a data collection filter instead. For instructions on using data collection filters, see "Data Collection Filters" in *Using Transaction Management*.

WebSphere MQ Agents and FASTPATH_BINDING

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

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

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 (`amqspout`, `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

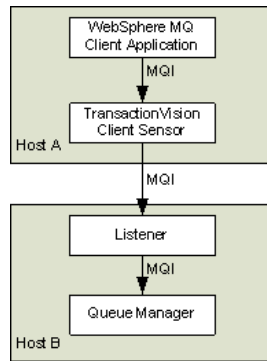
For applications using WebSphere MQ client bindings, TransactionVision is capable of monitoring and tracing these applications' messaging activities in either a distributed or centralized mode.

This section includes:

- ▶ "Distributed Monitoring" on page 239
- ▶ "Centralized Monitoring" on page 240
- ▶ "Installation and Configuration Considerations" on page 242

Distributed Monitoring

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



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

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

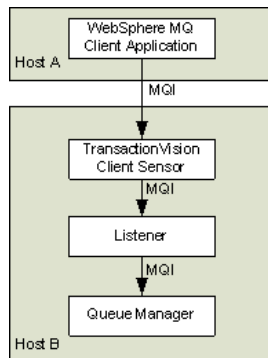
The trace information generated will be based on the client application context. This means information such as program name, program instance, and host name, will be related to the client application directly.

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

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

Centralized Monitoring

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



In this case, the agent is deployed on the host where the listener process and queue manager reside. Instead of direct monitoring of the client application, the agent monitors the listener program instead.

The agent intercepts and reports any MQI calls issued by the listener program. In other words, the listener program will execute the same MQI calls that the client application invokes (with a few exceptions, as described later). Therefore, by examining the listener program MQI call records, TransactionVision users can have a good understanding of the messaging activities originated from the client applications.

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

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

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

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

- ▶ As discussed before, context information reported is associated with the listener program. However, client application origin context information can be retrieved indirectly through the message header (MQMD) structure embedded in the MQPUT/MQPUT1/MQGET feedback data through the call parameters.
- ▶ If you use the Servlet, JMS, or EJB Agents with a client connection to a queue manager through a listener, which is being monitored with the TransactionVision WebSphere MQ Agent, internal TransactionVision events will be captured. It is recommended to either use a separate unmonitored listener for the Servlet, JMS, or EJB Agents or use server binding connections from these agents. If this cannot be done, change the data collection filter to exclude the configuration and event queues.

The table below summarizes the characteristics of the two approaches:

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

Installation and Configuration Considerations

For distributed monitoring, install client agents on each host where WebSphere MQ client applications run. Follow the standard agent installation instructions in "Installing and Configuring the Java Agent on Windows" on page 116.

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

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

20

Configuring the Proxy Agent

This chapter includes:

- ▶ About the Proxy Agent on page 245
- ▶ Application Requirements on page 246
- ▶ Enabling the Proxy Agent on page 246
- ▶ Configuring the Proxy Definition File on page 246
- ▶ Configuring the User Interface on page 249

About the Proxy Agent

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

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

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

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

Application Requirements

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

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

Enabling the Proxy Agent

The Proxy Agent is enabled by the TransactionVision license code.

Configuring the Proxy Definition File

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

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

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

This section includes the following:

- "Example" on page 247
- "Subelements" on page 247
- "Optional Attributes for the Proxy Element" on page 249

Example

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

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

Subelements

Specify the following subelements for each proxy element:

Element	Required?	Attributes	Description
Request	Yes	queue queueManager	The WebSphere MQ queue and queue manager from which the proxy program gets the event.
Reply	Yes	queue queueManager	The WebSphere MQ queue and queue manager where the proxy program puts the reply event of the same message ID and correlation ID as the request event.

Element	Required?	Attributes	Description
Program	Yes	name path	The name and path of the proxy program.
Host	Yes	name os	The name and operating system of the host where the proxy program runs.
Retrieve	No	queue	Causes the Proxy Agent to check the given queue object against its definition rather than the object defined in <Reply>. This element allows the agent to work with looser coupling.
z/OSBatch	No	jobID jobName stepName tcbAddr	If the proxy program is an z/OS Batch job, specify the job ID, job name, step name, and TCB address.
z/OSCICS	No	regionName transactionID taskNum	If the proxy program is an z/OS CICS task, specify the region name, transaction ID, and task number.
z/OSIMS	No	psbName transactionName regionID jobName imsID imsType	If the proxy program is an z/OS IMS job, specify the PSB name, transaction name, region ID, job name, IMS ID and IMS type.

Optional Attributes for the Proxy Element

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

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

Configuring the User Interface

By default, the Component Topology does not show proxy related links in dynamic mode. To enable the proxy node in this view, set the `hasProxySensor` attribute in the `UI.properties` file to true. For more information about changing this configuration file, see *Using Transaction Management*.

21

Configuring Agent Logging

This chapter includes:

- ▶ Log Files on page 251
- ▶ Circular Logging on page 252
- ▶ Trace Logging on page 254
- ▶ Configuring Separate Log Files for Multiple Agent Instances on page 255
- ▶ Using Windows and UNIX System Logs on page 256

Log Files

Log files are different for each agent type.

Java Agents

By default, the TransactionVision Java Agents log error and warning messages to the logs directory where you installed the Java Agent. This location is stored in the **Sensor.Logging.xml** file.

To enable the Java Agents to print banners when activated, set the **com.bristol.tvision.sensor.banner Java** system property to true. The banner is printed to standard output.

WebSphere MQ agents

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

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

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

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 253
- "Maximum Log File Size" on page 253
- "Maximum Number of Backup Log Files" on page 254
- "Changing from Circular to Linear Logging" on page 254

Log File Naming

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

- `sensor.log.2` is removed if it exists.
- `sensor.log.1` is renamed `sensor.log.2`.
- `sensor.log` is renamed `sensor.log.1`.
- A new `sensor.log` is created.

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

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

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

Maximum Log File Size

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

Maximum Number of Backup Log Files

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

Changing from Circular to Linear Logging

To use linear logging rather than circular logging, perform the following steps:

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

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

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

Trace Logging

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

You can enable trace logging in TransactionVision agents to debug agent configuration issues. Trace information for the WebSphere MQ Agent is logged to the UNIX system log, the Windows event log, the z/OS operator console log, or the i5/OS user's job log. For the Java Agents, trace messages are sent to the Agent's log4j `TraceLog`.

To enable or disable agent trace logging in the communication link, see "Communication Links" in *Using Transaction Management*.

Configuring Separate Log Files for Multiple Agent Instances

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

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

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

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

For serverB, set the JVM property as follows:

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

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

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

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

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

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

Using Windows and UNIX System Logs

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

For both Windows and UNIX, you must define a specialized event appender. This section includes:

- ▶ "Windows Event Appender" on page 256
- ▶ "UNIX Event Appender" on page 257

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

22

Configuring Security

This chapter provides information about the security setup procedures of TransactionVision.

This chapter includes:

- ▶ About Security in TransactionVision on page 261
- ▶ Managing User Permissions on page 262
- ▶ Securing with Light Weight Single Sign On (LW-SSO) on page 267
- ▶ Basic Authentication Configuration on page 269
- ▶ Securing the TransactionVision Database on page 269

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.

Securing the TransactionVision Analyzer

The simplest mechanism to control access to an Analyzer from the host it is running on, is file permissions for the TransactionVision install. These permissions can be adjusted as needed. For example, on UNIX, you should only allow read/write/execute permission to either the owner of the install, or those in a designated group that is authorized to manage TransactionVision.

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 263
- ▶ "BSM Predefined Roles Relevant to Transaction Management" on page 265
- ▶ "About User Data" on page 266
- ▶ "Additional Permission Requirements for Some Reports and Topologies" on page 267

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	<p>Change. User is allowed to modify the configuration of any existing Processing Server as well as add and delete Processing Servers.</p> <p>Full Control. User is allowed Change operation as well as ability to grant and revoke operations to other users.</p>
TransactionVision Analyzer	<p>Change. User is allowed to modify the configuration of any existing Analyzer as well as add and delete Analyzers.</p> <p>Execute. User is allowed to start/stop all Analyzers.</p> <p>Full Control. User is allowed Change and Execute operation as well as ability to grant and revoke operations to other users.</p>
TransactionVision Job Manager	<p>Change. User is allowed to modify the configuration of any existing Job Manager as well as add and delete Job Managers.</p> <p>Execute. User is allowed to start/stop all Job Managers.</p> <p>Full Control. User is allowed Change and Execute operation as well as ability to grant and revoke operations to other users.</p>
TransactionVision Query Engine	<p>Change. User is allowed to modify the configuration of any existing Query Engine as well as add and delete Query Engines.</p> <p>Execute. User is allowed to start/stop all Query Engines.</p> <p>Full Control. User is allowed Change and Execute operations well as ability to grant and revoke operations to other users.</p>
Administration	<p>Change. The Transaction Management Administration page is enabled for the user.</p> <p>Full Control. User is allowed Change operation as well as ability to grant and revoke operations to other users.</p>
User Data	<p>See "About User Data" on page 266.</p> <p>View. User is allowed to view all user data.</p> <p>Full Control. User is allowed View operations as well as ability to grant and revoke operations to other users.</p>

Resource	Operations that can be granted
User-created Reports	<p>View. User is allowed to view all user-created reports.</p> <p>Full Control. User is allowed View operations as well as ability to grant and revoke operations to other users.</p>
Applications	<p>Add. User is allowed to create new application instances.</p> <p>Change. User is allowed to create new business transactions under any application as well as modify monitoring and tracing configuration for any business transaction.</p> <p>View. User is allowed to view transaction data associated with any application in the reports.</p> <p>Full Control. User is allowed Add, Change and View operations as well as ability to grant and revoke operations to other users.</p>
An application instance	<p>Change. User is allowed to create new business transactions under the application as well as modify monitoring and tracing configuration for any of its business transaction.</p> <p>View. User is allowed to view transaction data associated with the application in the reports.</p> <p>Full Control. User is allowed Change and View operations as well as ability to grant and revoke operations to other users.</p>

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 267.

About User Data

Some transaction and event content collected by TransactionVision agents may contain data that is considered sensitive to the business. This might include items such as a credit card number, social security number, etc. If it is determined that TransactionVision has sensitive data that only certain people are allowed to view, this can be controlled through the User Data resource. By disabling access to user data, the information shown by Transaction Management is limited to the generic data fields that are available on all event and/or transaction data.

If User Data view permission is denied, the following occurs:

- ▶ Event Details for an event displays: **You do not have permission to view user data.**
- ▶ Transaction Tracking report does not display custom columns.
- ▶ Transaction Detail page does not display custom data in the Summary section.
- ▶ When defining transaction tracing rules, the Create or Edit Match Condition dialog does not allow selection of a transaction and does not populate the Values pane with transaction values if classification is based on user data.

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 ODB and are therefore subject to additional ODB **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 ODB permissions context) to allow users access to these reports and topologies.

Using the BSM roles does not require the additional ODB Business Transactions resource **View** permission.

Securing with Light Weight Single Sign On (LW-SSO)

In order 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 only a correctly authenticated user accesses 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 LWSSO configuration that is set up in BSM.

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. 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 run in Demo mode or disabled altogether.

► LW-SSO in Demo mode

If BSM is running LW-SSO in Demo mode, TransactionVision inherits this when

► LW-SSO Disabled

TransactionVision can work with LW-SSO disabled, however the credentials passed to the TransactionVision Processing Server are passed in a non-secure way. 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 or running in Demo mode causes a vulnerability that can allow authentication to be bypassed, and would not be a recommended approach if you have sensitive data or need to ensure that access to Transaction Management data and administration is only ever possible by those with the correct authorizations.

Select Access and SiteMinder Configuration

The following BSM Gateway server web resource needs to be unprotected:

- `/topaz/services/technical/time`

Basic Authentication Configuration

If the BSM Gateway server is configured with basic authentication, the following web resources need to be unprotected in order 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 SQL Server Authentication for SQL Server.

Database privileges

In order to run TransactionVision successfully in the given database environment, you have to make sure that the database user configured for TransactionVision has the necessary database privileges to access all required database objects. These privileges consist of:

- ▶ SELECT, 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 (e.g. a user with the DBA role) for accessing the database. If this is not possible, the above listed privileges have to be granted specifically. The CREATE TABLE/ CREATE INDEX privileges are not mandatory for running TransactionVision once the tables have been created, so it is possible to avoid granting these privileges by having a database DBA create the 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 Analyzer tables:

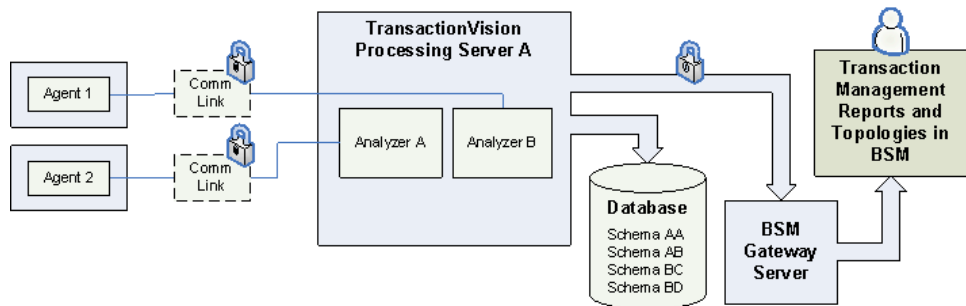
- ▶ CreateSqlScript -create -schema SCHEMA
- ▶ CreateSqlScript -grant -schema SCHEMA

If the configured database user does not have the CREATE TABLE/CREATE INDEX privileges, you will not be able to create the tables for a new Analyzer from within the Transaction Management Administration pages. Instead, the tables will need to be created manually before the Analyzer is created.

Securing with SSL

If the TransactionVision environment includes sensitive data being sent between the components, you can enable SSL for each communication path.

TransactionVision Processing Servers communicate with BSM Gateway Servers and with the communication links collecting events from agents. Each of these data paths are eligible for SSL:



For information about enabling SSL on the data path between Processing Servers and BSM Gateway Servers, see the *HP Business Service Management Hardening Guide* PDF.

Information about enabling SSL on the communication link data path is described in the sections that follow. Enabling SSL on the communication link data path between an Analyzer and agent involves these three areas:

- "Enabling SSL for the Messaging Middleware Provider" on page 271
- "Enabling SSL for the Agent" on page 273

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

Use the Progress SonicMQ Management Console to configure SonicMQ for SSL communication.

- 1 In the Management Console, navigate to the Acceptors item of your broker.

Under there you should see an SSL based Acceptor and a https based acceptor. The Tuxedo and NonStop TMF Agents make use of the https based acceptor. If you are not using those agents, then configuring the https direct acceptor is not needed.

- 2 Open the properties of the SSL or HTTPS direct acceptors and click on the SSL tab. Define the key stores as follows.

Format: PKCS12

Note: In the Generating a Keystore section, the PKCS12 format type is used in the example to generate keys. While Sonic can support the other keystore format (JKS) generated by keytool, it requires being configured differently. If you do chose to use a JKS keystore type will need to set your broker to use a jsseSSLImpl provider and configure it accordingly. Please see the SonicMQ documentation for more information on this.

Path Name: TVHOME/serverkey.p12

Password: enter the password for the key/keystore.

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 271.

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 `TVHOME/config/services/Analyzer.properties`. To add properties to a Java Agent, modify the `tvProperties` field found in `PROBEHOME/etc/TV.properties` files.

Enabling SSL for the Agent

The procedure is specific to the type of agent used in your deployment environment.

To configure Java Agents to use SSL:

- 1 Modify `probe_home/etc/TV.properties` to add `rsa_ssl.jar` to `appTransportLoadPath` as follows:

```
appTransportLoadPath=mfcontext.jar;sonic_Client.jar;sonic_Client_ext.jar;sonic_XA.jar;rsa_ssl.jar;jms.jar
```
- 2 Modify `JAVA_AGENT_HOME/config/SensorConfiguration.xml` to change `SonicMQTransport BrokerURL` from `tcp://yourhost:21111` to `ssl://yourlhost:21112` as follows:

```
<SonicMQTransport>
  <BrokerURL>
    ssl:// yourlhost:21112
  </BrokerURL>
  .....
</SonicMQTransport>
```

By default Java Agents configured to talk to the TransactionVision Message bus via SSL look at the `<TVHOME>/config/sensor` directory for the certificates to trust. So you would need to have the certificate (`serverkey.cer`, in the example above) copied to the `<TVHOME>/config/sensor` directory on any machine with a Java Agents installed.

Note:

- ▶ This default directory location can be changed by setting the `defaultTransport.ssl_cert_dir` property located in the `probe_home/etc/TV.properties` file. The path here can be either a relative path or a full path if directory location exists outside of the TransactionVision install. If you do enter a custom path, it is important to note that only forward slashes '/' should be used when entering the directory path, even on windows.
 - ▶ If the application that is being monitored by a Java Agent is itself using SSL enabled Sonic to communicate with queues, some additional items need to be considered. Because of a limitation in SonicMQ, it is not possible to configure two separate components running within the same process to use a differing directory location for where certificate files are lookup. Thus, both the Agent and the application itself would need to have any trusted certificates located in the same location. If this is the scenario, you would need to change the `defaultTransport.ssl_cert_dir` property located in the `probe_home/etc/TV.properties` file to point to where the application is already configured to look for its certificates and additionally copy the TransactionVision certificate to that directory.
-

To configure the Tuxedo Agent to use SSL:

- 1** To retrieve configuration information from the Analyzer, edit the Tuxedo sensor property file.

This can be found in the directory "`<TVHOME>/tuxedo/config/TuxedoSensor.properties`" on the host on which the Tuxedo sensor is installed.

- 2** In this property file, change the "transport" field to specify the HTTP over SSL URL for the Analyzer configuration service. By default, this URL is "`https://myhost:21103`".

- 3** To configure the Tuxedo Agent to send events to the Analyzer using SSL, edit the communication link used by the Tuxedo agent. For the Agent Connection, set the "Connection URL" field to specify the HTTP over SSL URL for the SonicMQ HTTP Direct for JMS Acceptor. By default, this URL is "https://myhost:21114".

Tuxedo Agents configured to use SSL need to have a copy of the SSL certificate in PEM format ("serverkey.pem" in the "Generating a keystore" example above). This must be copied to the "<TVHOME>/tuxedo/certs" directory on any machine on which the Tuxedo Agent is installed.

Additional properties controlling the Tuxedo Agent's behavior with respect to SSL are available and documented in the Tuxedo sensor properties file "TuxedoSensor.properties".

To configure the .NET agent to use SSL:

Steps 1-19 step 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 this console is running on), 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 using "NETWORK SERVICE" (this is the default) credentials the Certificate Store is that of 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. (serverkey.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://".

For SSL and the WMQ Agents on Windows or UNIX:

If the WMQ agent is monitoring a WMQ application making client connections, the WMQ agent cannot open a connection independent of the type of connection the WMQ application itself makes. Thus if the application is making non-secured WMQ calls, the WMQ Agent is limited to that. If on the other hand the application itself is using an SSL connection, the WMQ Agent would use that secured connection.

If the WMQ Agent is monitoring a WMQ application making server connections, the only place where SSL would come into play is if there is a remote channel between the application's queue manager and a dedicated TV queue manager. This would be purely configured in WebSphere MQ - the application, or agent would not need to be aware of it. If the analyzer is also using a server WMQ connection to retrieve events, it similarly would not need SSL enabled. If the analyzer were 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 271).

For SSL and the WMQ and CICS Agents on z/OS:

Because the z/OS Agents all only use server connections, as mentioned above, SSL configuration is not needed. Again, the only place where SSL would come into play is if there is a remote channel between the application's queue manager and a dedicated TV queue manager. This would be purely configured in WebSphere MQ - the application, or agent would not need to be aware of it. If the analyzer reading these messages is also using a server WMQ connection to retrieve events, it similarly would not need SSL enabled. If the analyzer were 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 271).

Generating Certificates

Before you can configure SSL, you must generate a certificate for use in encrypting the communication. A certificate consists of a private key (accessible only by the server), and public key that is sent to clients in order to encrypt traffic. In order for the client to ensure that it is communicating with who it expects to, when it receives the public key from the server, it must establish whether this public key is trusted. A key is trusted if the key, or the certificate authority signing the key, belongs to a set of trusted entities that the client has been configured to trust.

In setting up TransactionVision components, it may be helpful to keep in mind that setting up the TransactionVision Analyzer (Configuration Message Service) and SonicMQ Broker to accept SSL communication requires configuring both a private and public key. Setting up a communication component with the following clients requires ensuring the server's public key is trusted by the client:

- ▶ Java Agents (to SonicMQ server)
- ▶ TransactionVision Analyzer(to SonicMQ server)
- ▶ Browsers connecting to the Transaction Management UI
- ▶ TransactionVision Processing Server connecting to HP Business Service Management.

The Analyzer acts as both a server for BEA Tuxedo and NonStop TMF Agents requesting configuration messages, and as a client communicating messages to the SonicMQ server. Both of these modes can be configured to communicate using SSL technology.

The agents use SonicMQ to communicate event data for reading by the Analyzer and can be configured to use SSL for this data.

The Analyzer and SonicMQ Server can all be configured to share the same keystore, if they all reside on the same host. If they reside on different hosts they require a separate key to be generated.

The procedure that follows uses the **keytool** Java utility to generate example certificates and keystores. See <http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html> for more information.

Generate a Keystore:

TransactionVision by default ships with and is configured to use a temporary certificate for using SSL. The temporary certificate is intended to act as an example about how and where the configuration is typically set up. This key cannot be used in a real deployment.

To generate a keystore, follow these steps:

1 Generate a keystore with the following command:

```
keytool -genkey -alias tvserverkey -keyalg RSA -storeType PKCS12 -keystore TVHOME/serverkey.p12
```

The command prompts you for information regarding the creation of the key. If you use a password other than the default "changeit", be sure to record it as you it will be needed to access this key.

Note:

- ▶ While keytool does not enforce it, SonicMQ requires that the two-letter country code be no longer than 2 characters. Therefore specify a 1 or 2 character country code only.
- ▶ When generating a key the CN of the certificate should be set to the fully qualified hostname on which the Analyzer runs. In order to accomplish this, when keytool prompts for "your first and last name", the fully qualified Analyzer hostname should be used.

2 Export this servers certificate's public key:

```
Keytool -export -alias tvserverkey -file serverkey.cer -storeType PKCS12 -keystore TVHOME/serverkey.p12
```

This exports the key to a file called **serverkey.cer**. Any client attempting to access use the SSL connection (Analyzer connecting to SonicMQ via SSL, agents connecting to SonicMQ and/or the Analyzer), needs to be configured to trust this certificate.

The BEA Tuxedo Agent requires the server certificate's public key be in PEM format. To export the key in this format use the following command:

```
keytool -export -rfc -alias tvserverkey -file serverkey.pem -storeType PKCS12  
-keystore TVHOME/serverkey.pem
```

Note: The TransactionVision Analyzer and built-in SonicMQ Broker each requires a private key to be created. If two or more of these components reside on the same hostname, they can share the same private key. Otherwise a unique key must be generated for each host that one of these components run on.

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.

After this step, BPI events will be sent with SSL enabled. You must also have set up the Analyzer environment to allow for SSL communication. See *Configuring Analyzer Server for SSL communication* section for the needed steps.

Enabling Authentication in TransactionVision SonicMQ

These are the steps 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 UI.

1 Shutdown the nanny using the command:

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

```
<TVISION_HOME>/bin/run_topaz stop (UNIX)
```

2 Start the SonicMQ domain manager using the script:

```
<TVISION_HOME>\Sonic\MQ7.5\bin\startdm.bat (Windows) or
```

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

3 Start the SonicMQ broker using the script:

```
<TVISION_HOME>\Sonic\MQ7.5\bin\startbroker.bat (Windows) or
```

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

4 Start the Management Console using the script:

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

```
<TVISION_HOME>/Sonic/MQ7.5/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 hostname and right-click to view the dropdown menu; select 'Properties' from the menu.
- d** On the Edit Broker Properties Window, enable Security by checking the checkbox next to 'Security' in the Security section.
- e** Set the broker password by pressing 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\Docs7.5\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 script:
<TVISION_HOME>\Sonic\MQ7.5\bin\stopdm.bat (Windows) or
<TVISION_HOME/Sonic/MQ7.5/bin/stopdm.sh (UNIX)
- 9** Stop the SonicMQ broker using the script:
<TVISION_HOME>\Sonic\MQ7.5\bin\stopbroker.bat (Windows) or
<TVISION_HOME/Sonic/MQ7.5/bin/stopbroker.sh (UNIX)
- 10** cd to <SONICMQ_HOME>, which is the same as
<TVISION_HOME>\Sonic\MQ7.5.
- 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 command:
<TVISION_HOME>\bin\SupervisorStart.bat (Windows) or
<TVISION_HOME/bin/run_topaz start (UNIX)

Index

A

Agents

- backward compatibility 32, 33
- client application monitoring 238
- installation on IBM i5/OS 111
- loading WebSphere MQ 228
- logging 251
- multiple log files 255
- overview 24
- See also BEA Tuxedo, CICS, Java, .NET, NonStop TMF, and WebSphere MQ Agents
- trace logging 254

Analyzer

- architecture 23
- installing 75
- analyzer.log 79
- analyzer_startup.log 79
- APP CTL HEAP SZ 65
- APPLHEAPSZ 65
- ASP.NET applications 93
- Audience, for documentation 14
- auto_detect.points file 140

B

Backward compatibility of

- TransactionVision components 32

Batch MQ

- applications 92

BEA Tuxedo Agents

- applications 92
- installing 193
- overview 96
- rebinding 195
- uninstalling 195

BEA Tuxedo Server 92

- Beans.xml 189

C

CICS Agents

- applications 92
- configuring 159
- circular logging 80, 252
- client application monitoring 238
- COLDMON 220
- components 102
- configuration queue check interval 229
- configuration queue name 228
- conventions, typographical 17
- custom user events
- configuring 140

D

database

- configuration and tuning 66
- in deployment environment 27
- overview 24
- performance 66
- storing unicode data 70
- supported platforms 38
- Database.properties file 70
- DB2 variable settings 65
- DB2Test utility 67
- documentation
- download site 17
- overview 15

E

- EJB Agents
 - applications 92
 - overview 95
 - supported platforms 43
- EJB Beans 92
- environment variables
 - LD_LIBRARY_PATH 225
 - LIBPATH 225
 - SHLIB_PATH 225
- event appender
 - UNIX 83
 - Windows 82
- event log 254
- exit_sensor.deny 230

F

- FASTPATH_BINDING 237
- Flash Player support 47

H

- HP Software Support Web site 16
- HP Software Web site 17

I

- I18N support 47
- IMS MQ Agents
 - applications 92
- IMSBridgeObject.xml 189

J

- Java Agent Setup Module 118
- Java Agents
 - about 114
 - installation files 116
 - installing on UNIX 123
 - installing on Windows 116
 - launching the installer on Windows 116
 - logging 251
 - silent installation 141
- Java Servlet Agents

- applications 92
- Java Support 47
- JDBC Agents
 - overview 95
 - supported platforms 45
- JMS Agents
 - overview 94
 - supported platforms 44
- JRE Instrumenter
 - processing 129
 - running on UNIX 134
 - running on Windows 129

K

- Knowledge Base 16

L

- LD_LIBRARY_PATH environment variable 224, 225
- LIBPATH environment variable 225
- linear logging 82
- localization 47
- logging
 - circular 80, 252
 - Java Agents 251
 - linear 82
 - multiple log files 255
 - separate log files for multiple Sensors 255
 - SMTP 84
 - SNMP 85
 - trace 254
 - UNIX event appender 257
 - WebSphere MQ Agents 251
 - Windows event appender 256

M

- MAXAPPL 65
- Messaging middleware 92
- messaging system providers
 - configuring SonicMQ 138
 - configuring WebSphere MQ 229
- MQ_CONNECT_TYPE 237

MQ_IMS Bridge Agents
 applications 92

N

.NET Agents
 configuring 208
 determining version 216
 installing 202
 overview 95
 supported platforms 46
 uninstalling 216
 .NET Remoting 93
 NonStop TMF Agents
 about 218
 applications 93
 configuring 221
 installing 219
 overview 96
 shutting down 220
 starting 220
 supported platforms 46
 NT_EVENT_LOG 82

O

online resources 16
 operator console log 254
 Oracle variable settings 65
 OracleTest utility 67

P

PATH environment variable 224, 225
 Processing Server
 about 51
 backward compatibility 32, 33
 installing on Windows 55
 overview 23
 supported platforms 37
 uninstalling 60
 Proxy Agents
 application requirements 246
 configuring 245
 configuring the definition file 246
 configuring the user interface 249

enabling 246
 option attributes 249
 Proxy Sensors
 subelements 247
 ProxySensorDef.xml 246

S

Servlet Agents
 overview 94
 SHLIB_PATH environment variable 224, 225
 SMTP logging 84
 SNMP logging 85
 SonicMQ
 configuring 138
 SSL
 Configuring for TransactionVision
 278
 STOPMON 221
 STRTMON 221
 subelements of Proxy Sensor 247
 SYSLOG 83
 system log 254
 System requirements 35

T

trace logging 254
 Transaction Constructor algorithm 21
 TransactionVision
 architecture 22
 Configuring SSL for 278
 deployment environment 25
 documentation set 15
 Troubleshooting and Knowledge Base 16
 Tuxedo Server 92
 TVISION_CONFIG_CHECK_INTERVAL
 environment variable 229
 TVISION_CONFIGURATION_QUEUE
 environment variable 228
 TVISION_SYSLOG environment variable 254
 tvisionapiexit 229
 typographical conventions 17

U

- unicode data 70
- UNIX
 - event appender 83
- upgrade 99
- User Event Sensor
 - modifying on Windows 101
- User Event Sensors
 - installing on UNIX 105
 - uninstalling on Windows 102
 - upgrading on Windows 99

W

- Web browsers
 - supported configurations 46
- WebSphere MQ
 - configuring messaging system providers
 - configuring WebSphere MQ 138
 - configuring the messaging system provider 229
- WebSphere MQ Agents
 - applications 92
 - installing on UNIX 105
 - logging 251
 - modifying on Windows 101
 - overview 93
 - rebinding on AIX 108
 - supported platforms 40
 - uninstalling on UNIX 108
 - 108
 - uninstalling on Windows 102
 - upgrading on Windows 99
- WebSphere MQ API Exit Agents
 - configuring on distributed platforms 231
 - configuring on i5/OS 231
 - configuring on Windows 234
 - discarding WMQ events on TransactionVision queues 237
 - overview 93
- WebSphere MQ Library Agents
 - overview 93

Windows

- event appender 82
- WMQ Batch Agent 94
- WMQ Batch Agents
 - configuring and starting 159
- WMQ IMS Bridge Agent 94
- WMQ-IMS Bridge Agents
 - using 184
- wmqsensor 226

Z

- z/OS WebSphere MQ Agents
 - overview 94