# TransactionVision®

# Installation and Configuration

*Version 4.2.1*

Bristol Technology Inc.
39 Old Ridgebury Road
Danbury, CT 06810-5113
USA
(203) 798-1007

Bristol Technology BV
Plotterweg 2A
3821 BB Amersfoort
The Netherlands
+31 (0)33 450 50 50

Printed March 04, 2005

This manual supports TransactionVision Release 4.2.1 SupportPac A.


Bristol Technology Inc.
39 Old Ridgebury Road
Danbury, CT 06810-5113 U.S.A.



Part No. TV12050304 Version 4.2.1 SupportPac A

# Contents

# Chapter 1
# TransactionVision Overview

TransactionVision is the transaction tracking solution that graphically shows you the interaction between all the components of your system. TransactionVision non-intrusively records individual electronic events generated by a transaction flowing through a computer network. More importantly, TransactionVision's patent-pending "Transaction Constructor" algorithm assembles those events into a single coherent business transaction.

Graphical analysis of business transactions enable you to:

- Find lost transactions
- Monitor and meet service level agreements
- Improve efficiencies of your business processes

## TransactionVision Basics

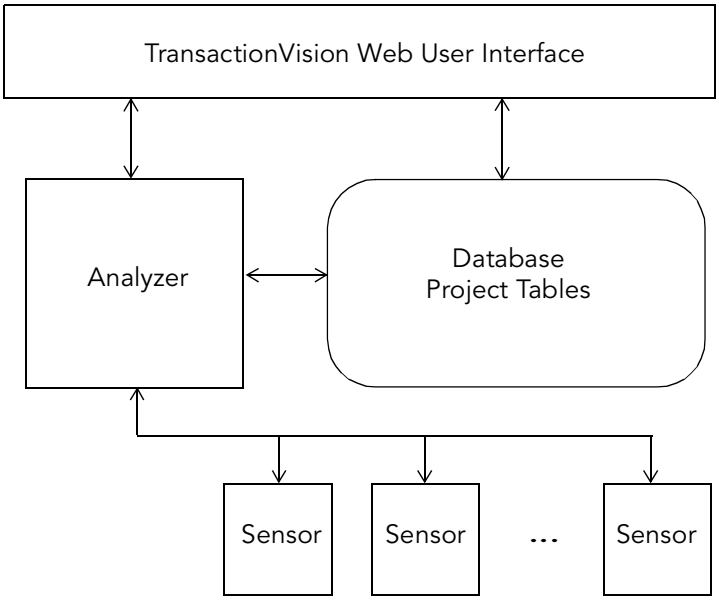
To understand the tasks required to administer TransactionVision, you must understand the TransactionVision components, as well as some basic concepts.

### Components

TransactionVision consists of three major components:

- Sensors
- Analyzers
- TransactionVision Web User Interface

The following diagram shows the relationship between these components:




## Sensors

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

TransactionVision provides several types of Sensors:

- WebSphere MQ Sensors
- Servlet Sensor
- JMS Sensor
- EJB Sensor
- CICS Sensor

### WebSphere MQ Sensors

The **WebSphere MQ Sensor** tracks MQ API calls. These API calls include the entire MQ API set, the major APIs being MQPUT, MQGET, MQCONN, MQDISC, MQOPEN, MQCLOSE, etc. There are two types of WebSphere MQ Sensors provided by TransactionVision on distributed platforms: the WebSphere MQ Library Sensor and the WebSphere MQ API Exit Sensor. Both of these Sensors report the same information from an MQ API call. They differ primarily in the mechanism by which they intercept MQ API calls, their usage, and the amount of data they collect from the system.

- The **WebSphere MQ Library Sensor** intercepts a WebSphere MQ API call by the shared library (or DLL) interception method on distributed platforms. This involves placing the TransactionVision Sensor libraries before the WebSphere MQ libraries in the application library path. This method is useful if you need to track MQ APIs for specific applications.
- The **WebSphere MQ API Exit Sensor** uses the WebSphere MQ API exit support available on distributed platforms in WebSphere MQ v5.3 and later. This Sensor is registered as an exit to the queue manager and invoked when any program connecting to the queue manager invokes a WebSphere MQ API. This method is recommended to collect MQ events from all applications on a queue manager and in particular the listener and the channel agents.
- **z/OS WebSphere MQ Sensors** are provided for tracking MQI API calls in the CICS, batch and IMS environments on the IBM z/OS system. In the CICS environment, the API crossing exit provided by the CICS adapter for WebSphere MQ is used to intercept the MQ API. In the batch and IMS environments, the application has to be re-bound with the Sensor to intercept MQ API calls.

The following supplemental Sensors are available for WebSphere MQ:

- The **Proxy Sensor** correlates business transactions into process that are not monitored using the TransactionVision Sensor libraries (for example, events between a Sensored application and an application running on a system where no Sensor is installed such as an external partner system).
- The **WebSphere Business Integration Sensor** (previously known as the MQSI Sensor) distinguishes the various message flows and identifies individual logical transaction paths within WBI. This Sensor is a WBI plugin that provides a trace node, which is inserted into the normal execution path of a message flow, and a failure node, that is inserted into the failure path of a message flow. These nodes generate a MQSI2TRACE event that allows tracking of the message flow within WBI.

- The **WebSphere MQ-IMS Bridge Sensor** tracks WebSphere MQ-IMS bridge messages rather than the WebSphere MQ API calls made by the calling applications. The MQ-IMS Bridge is a component that enables WebSphere MQ applications to invoke IMS transactions and receive their reply messages. The MQ-IMS Bridge Sensor tracks MQ messages coming into the bridge and correlates them with the reply received from IMS. Two events, MQIMS_BRIDGE_ENTRY and MQIMS_BRIDGE_EXIT are generated for every message coming in and going out of the bridge. These events contain the MQ message header and information about which IMS transaction is invoked.

### Servlet Sensor

The **Servlet Sensor** tracks servlet methods in a J2EE application server. This Sensor tracks HTTP calls such as HTTP_POST, HTTP_GET, HTTP_PUT, etc., which result in method calls into the J2EE container. The Servlet Sensor tracks these method invocations by instrumenting the servlet to collect events at the entry and exit of each call.

### JMS Sensor

The **JMS Sensor** tracks WebSphere MQ Java Message Service events from standalone Java applications as well as from J2EE application servers. This Sensor tracks JMS interface methods such as send, receive, etc. These methods are tracked by instrumenting the JMS library to collect events at the entry and exit of each call.

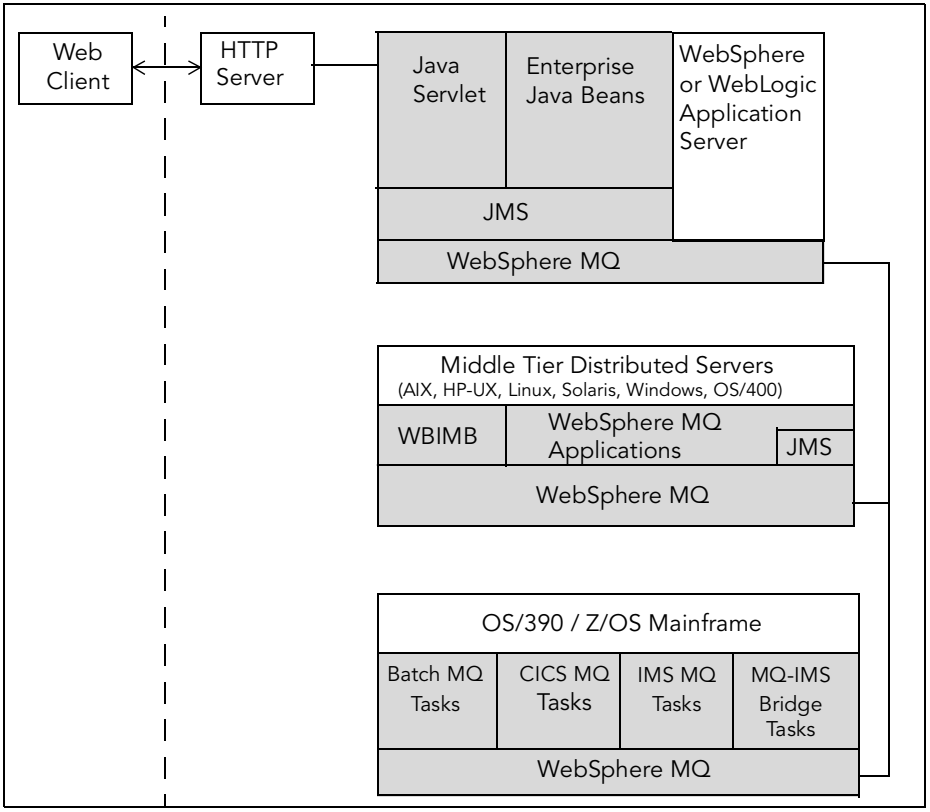
### EJB Sensor

The **EJB Sensor** tracks transactions through business logic within a J2EE application server. This Sensor tracks all public business methods in an entity bean, session bean or message driven bean. In addition to the business methods, this Sensor tracks the ejbCreate, ejbPostCreate, ejbRemove, ejbLoad, ejbStore and onMessage methods. These methods are instrumented by the Sensor to collect events at the entry and exit of each call.

### CICS Sensor

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

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




## Analyzer

The Analyzer is a service that communicates with Sensors via WebSphere MQ. It generates and delivers configuration messages to Sensors by placing them on a designated configuration queue. Configuration messages specify Sensor configuration information such as the name of the event queue where the Sensor should place event messages and data collection filter definitions for the project.

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

Each TransactionVision project is assigned a single host running the Analyzer. Projects enable you to easily group and manipulate communication links, data collection filters, database schemas, and Analyzers as one entity. When you start a project, the Analyzer on the host assigned to the project is started automatically. You may also start the Analyzer on a host from the Analyzers page in the web user interface, in which case the Analyzer starts processing events on all active projects it is assigned to.

### Web User Interface

The TransactionVision web user interface is an enterprise application for IBM WebSphere or BEA WebLogic that provides the TransactionVision graphical interface. All interaction is done through web pages. Users and administrators login to the web user interface through a web browser. The web user interface communicates with the Analyzer to provide data collection configuration information such as communication links and data collection filters. It also connects to project database schemas to display project analysis and report results.

## Contacting Bristol Support

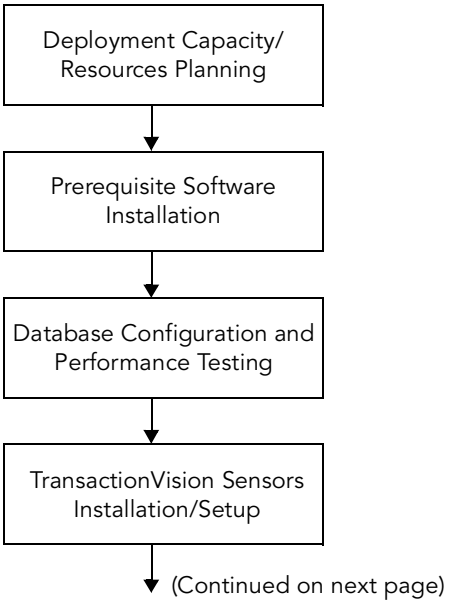
If you encounter a problem installing or configuring TransactionVision, contact Bristol Support by any of the following methods:

- TransactionVision: Choose the Help > Bristol Support menu item. From the submenu, you may open the Bristol Support home page, open the TransactionVision knowledgebase, or open a problem report form.
- Web: http://www.bristol.com/support
- Email: support@bristol.com
- Phone: **203-798-1007** Press 3 (US & Far East)
  or **+31-33-450-5050** (Europe, Africa, and Middle East)

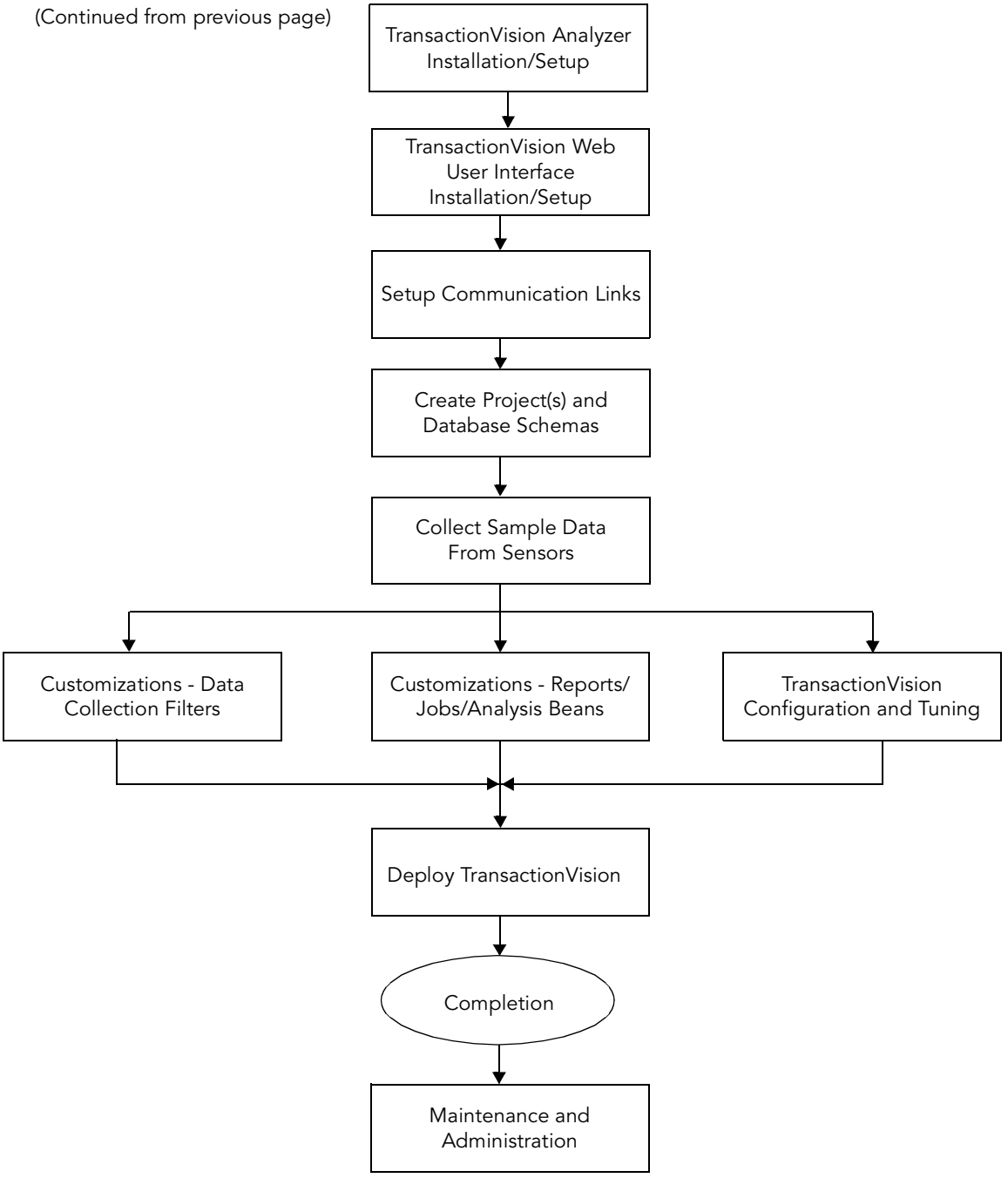
# Chapter 2
# Planning TransactionVision Deployment

Bristol recommends that you plan your TransactionVision deployment carefully before installing TransactionVision. It is important to install TransactionVision components on the appropriate hosts in your environment. This chapter provides a roadmap and guidelines for planning your deployment.

The following diagram shows a high level task flow depicting a typical deployment scenario. It provides an overview of the tasks required for a successful TransactionVision deployment.



(Continued from previous page)



## Deployment Planning

Proper planning is important for successfully deploying TransactionVision, and the amount of planning necessary will depend on the nature of deployment. Large production deployments will require more up-front effort in capacity and resource planning, while smaller development and QA deployments with a single machine environment may require minimal planning.

In many cases, a prototype TransactionVision deployment may be used to self-discover existing transaction paths, which may help to determine the scope and capacity of the installation. Install the TransactionVision Sensor on the servers running the major applications of the system:

1. Set up communication links from the Sensor to the Analyzer.
2. Run the applications with the Sensor enabled, with data collection filters set to collect all data, for an appropriate length of time to collect all necessary data.
3. Use the data collected by TransactionVision to estimate message volumes, data sizes, number of queues and queue managers, application names, etc. This data can then be used to size Analyzer server hardware.

Deployment planning should include collection of the following information:

- Overall monitoring strategy
    - What applications and WMQ objects need to be monitored?
    - Where are the Sensors going to be deployed?
    - What communication links need to be created between the Sensor and the Analyzer?
    - How much data needs to be collected at each monitoring point?
    - What is the transaction rate?
    - What is the peak transaction rate?
    - What is the total amount of data that needs to be retained in the database before archiving/deletion takes place?
    - Can TransactionVision data be partitioned? If so, what is the criteria?
    - What is the level of tolerance regarding to loss of TransactionVision data?

- Deployment strategy
  - Where will the TransactionVision Analyzer(s) be hosted?
  - Where will the Websphere or WebLogic Application Server that will run the TransactionVision web application be hosted?
  - Where will the DBMS be hosted?
  - Personnel resource availability to install third-party software TransactionVision depends upon, such as IBM DB2, Oracle, WebSphere and WebSphere MQ on any new hardware.
  - Resource availability to install and setup TransactionVision.
  - What are the reports to be run and how often they will be run?
  - What access authorizations are to be provided to users for administering and operating TransactionVision and accessing TransactionVision data?

The first step in the planning process is to determine which servers you need to monitor.

### Identify Servers to Monitor

TransactionVision provides Sensors for WebSphere MQ applications, JMS applications, Servlets, Enterprise Java Beans, and CICS. The following section provides a worksheet to help you plan and list out the servers to be monitored by TransactionVision.

Note that the following information is optional, but may be useful for your capacity planning and recording purposes. Initial prototype deployments may not need to record this information. You will not be prompted to enter this information into TransactionVision during installation.

| Server Host Name | Type (WMQ/WBIMBI/ JMS/Servlet/ EJB/CICS) | Queue Manager(s) | Client/Server Connection | OS/390 CICS/ IMS Region |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Use this table to record the number of servers where the TransactionVision Sensor has been installed, which Sensor has been installed (WebSphere MQ, WebSphere Business Integration Message Broker, JMS, Servlet, EJB, CICS), the queue manager(s) the Sensor connects to, whether a client or server connection is used and if OS/390 is the platform, which CICS or IMS region the Sensor is installed under.

Additional information such as application name, WBIMB broker name etc. also may be collected.

## Dependent Software Installation

TransactionVision requires IBM DB2 or Oracle database software, IBM WebSphere MQ, and aIBM WebSphere BEA WebLogic application server software. Please refer to the release notes for the required versions of these software packages. Refer to the TransactionVision support knowledge base on the Bristol web site for common setup issues related to these packages.

TransactionVision uses the DBMS extensively for its data collection and analysis. Hence, the performance of the DBMS is vital to the overall performance of TransactionVision.

For details on DBMS configuration and tuning, see Chapter 4, “Configuring Databases.”

The Websphere Application Server should have a quick network access to the DBMS server. Running Websphere Application Server on a system with a slow link to the DBMS will result in very poor response time for the end user from the TransactionVision user interface.

## TransactionVision Installation

### Sensor Installation

To fully monitor your transactional environment, it is not necessary to understand the transaction flow, but to identify all the applications involved in the environment. The first step in the planning process is to determine which applications you need to monitor. TransactionVision provides Sensors for WebSphere MQ applications, JMS applications, Servlets, Enterprise Java Beans, CICS, the WebSphere MQ Business Integration Message Broker and the WebSphere MQ-IMS bridge. Refer to Chapter 3, “Installing TransactionVision,” for details on installing Sensors.

## Analyzer Installation

The appropriate capacity (CPU/memory/disk size) for the systems on which the Analyzer is being deployed needs to be determined based on the message rate from all the applications being monitored. Depending on your transaction volume, deploying several Analyzers might be advantageous; however, this approach requires partitioning data. In a partitioned environment, the information requested in the following tables should be provided for each data partition.

Bristol can determine sizing for your environment based on the information provided in the following tables.

| Data Partition (Business Process Name) | Estimated Average # of End-to-End Business Transactions per Day | Estimated Peak End-to-End Business Transactions per Day | Estimated Total # of API Calls per End-to-End Business Transaction to Be Monitored | Estimated Average Message Size including User Buffer |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

There are two alternatives for partitioning data.

- TransactionVision's data collection filters can be used to partition data. In order to take advantage of TransactionVision's filters, events captured by TransactionVision need to contain enough information to enable the data filters to partition data. For example, the WebSphere MQ header often contains information that can be used to identify specific business applications. Refer to the *TransactionVision Administrator's Guide* for details on using data collection filters and communication links.
- Alternatively, an organization's system architecture can be used to partition data. Many system architectures require a different message route for each business application. In these environments, data can be partitioned based on the company-wide system architecture.

Based on the message rate and TransactionVision data partition, the appropriate configuration (number of CPUs, memory sizes) of the systems for hosting TransactionVision Analyzers can be calculated based on the TransactionVision performance reference document.

| Analyzer Host | Host Type | Message Rate | Number of CPUs / CPU Speed | Memory Size | Disk Space |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |

The Analyzer may be deployed on a machine in the existing transactional architecture or on separate dedicated machines on the network. Take the following points into account during the deployment process:

- The machine must meet all TransactionVision software requirements. Refer to the release notes for required software versions.
- The machine must have access to the network where the transactional environment is deployed.
- The installation user must have administrative privileges to the machine, DB2/Oracle, and WebSphere MQ.
- The machine must have at least a client installation of WebSphere MQ.

Once all prerequisites are met and all of the above considerations are taken into account, refer to Chapter 3 for installation instructions.

### Web User Interface Installation

The TransactionVision web user interface may be deployed on any machine on the network that satisfies the following requirements:

- It has sufficient processing power to support the WebSphere or WebLogic application server and the number of users that are expected to use TransactionVision. Also, consider whether any report or analysis jobs are being planned to be deployed.
- All prerequisites are met. Refer to the release notes for required software versions.

- The machine has network access to the machine running the TransactionVision Analyzer. Many times it may be sufficient to run the web user interface and Analyzer on the same machine.
- You must have administrative access to the machine and WebSphere or WebLogic.

Once all prerequisites are met and all of the above considerations are taken into account, refer to Chapter 3 for installation instructions.

## TransactionVision Setup

Chapters 5 through 10 walk you through the process of setting up the TransactionVision product. This involves enabling the Sensors and setting up database access from the Analyzer and web component.

Refer to the *TransactionVision Administrator's Guide* for details on setting up projects, projects and database schemas, creating communication links for events to flow from the Sensor to the Analyzer, creating data collection filters (optional) and setting up users and their access rights (optional).

### Communication Link Setup

Once you have installed TransactionVision components and created projects, you must create communication links between the Analyzer and Sensor components. The communication links describe the path from the Analyzer to the different queue managers in the transactional environment. These only include the queue managers used by the applications being monitored.

The communication link essentially contains two queues that are used for communication between the Analyzer and the Sensor. The configuration queue is used to send configuration messages to the Sensor while the event queue is used to send information to the Analyzer from the Sensor. The configuration queue must be a local queue residing on the queue manager described in the communication link.

The main point to consider while defining communication links is to consider whether the Analyzer will connect directly to the queue managers in the transactional environment using a client connection or not. The reason for not using a client connection may be because some systems like the mainframe systems described in our sample architecture do not have the client connection infrastructure installed. Given this condition, we can complete the following tables to define our communication links.

| Sensor Host | Queue Manager Sensor Connects To | Can the Analyzer client connect to the queue manager? |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

The JMS, EJB, and Servlet Sensors need to be configured to use one queue manager. Refer to Chapter 8 for details on setup of JMS, EJB, and Servlet Sensors.

For communication links that cannot use client connections between the Analyzer and the queue manager listed above, communication paths from the queue manager local to the Analyzer to the queue managers in question must be defined. Essentially, it is necessary to set up the transmission queue with channels and, if necessary, remote queue definitions.

For each queue manager listed in the table, create a communication link. Please refer to the *TransactionVision Administration Guide* for details.

Once these communication links are created, they can be dynamically turned on or off in the projects within TransactionVision to control the amount of information coming in at any time.

## User Access

Using TransactionVision in conjunction with an LDAP server provides the administrator with role-based user access control. This user access control can be based on projects, reports, queries etc. Note, that setting up user rights is optional. A demo mode provides default users called “Admin”, “Developer”, “Operator” and “User”.

The users of TransactionVision are as follows:

- Administrators - have access to all features of TransactionVision.
- Business Users - have access to certain reports in the TransactionVision GUI.
- Support Engineer - have access to the control of a single project in TransactionVision. Can change the data collection criteria and setup reports for one single project.

- Support Specialist - They have access to the views and reports available in a project setup by the Support Engineer.

Once you have listed all users with the required access rights, refer to the *TransactionVision Administrator's Guide* for information about setting up user privileges.

## TransactionVision Customizations

Depending on the data collected, TransactionVision can be customized in several ways. The primary customizations include:

- Writing Analyzer beans and XDM files to extract message data fields, writing custom event correlation beans or writing transaction classification rules. Often, message data will contain useful information from which custom reports can be built. The first step for using this information is to convert binary message data into XML by writing an unmarshaller bean and then mapping XML fields into database table columns using TransactionVision XDM files. Event correlation beans may need to be written if parts of your system are not being monitored by TransactionVision. Transaction classification rules can be added to classify your system transactions into different categories and add attributes to those categories.
- Writing custom reports. Custom reports perform analysis on data collected by TransactionVision that is specific to your system being monitored.
- Writing job beans to perform batch data analysis or administration tasks. Job beans may be written if you need to perform any kind of a batch job such as data backup, deletion, custom analysis etc.

For details on implementing the above customizations, see the *TransactionVision Programmer's Guide*.

## Maintenance and Administration

The following tasks need to be planned for ongoing TransactionVision administration:

- DBMS storage space management - an effective mechanism needs to be devised to control the storage space used by TransactionVision data. This may include a regular data backup and deletion schedule. A policy needs to be in place to decide what data needs to be backed up. This can be decided based on what the data is being used for. While every scenario may be different, some examples might be to save complete event data which may be used for short term diagnostics and may be preserved for just a week. Alternatively, message data may be used for auditing and may need to be preserved for years. Statistical data may need to be preserved for a long period to observe trends in volume and response time.

- DBMS performance maintenance - RUNSTAT (for DB2) needs to be run on a regular basis to ensure optimal performance.
- Adding and changing projects, schemas, users, queries, filters, Sensors and communication links. Refer to the *TransactionVision Administristration Guide* for details on these tasks.

# Chapter 3
# Installing TransactionVision

The TransactionVision installation consists of the following three TransactionVision packages:

- The Analyzer binaries and configuration files. Install this package on all hosts where you want the Analyzer service to run.
- The TransactionVision Web User Interface, which consists of WebSphere or WebLogic related files such as the .ear file, JSPs, servlets, etc. Install this package on the hosts that users and administrators will access via web browsers to use and manage TransactionVision.
- TransactionVision Sensors. Install these packages on all hosts running applications that you wish to collect information about.

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

## Upgrading from Previous Releases

The TransactionVision 4.2.x release is backward compatible with version 4.1.

- For WebSphere MQ Sensors, you may use the TransactionVision 4.1 WebSphere MQ Sensor with TransactionVision 4.2, in which case you do not need to install the TransactionVision 4.2 Sensor. However, we recommend you install the TransactionVision 4.2 Sensor to take advantage of the latest updates. If you choose to upgrade to the TransactionVision 4.2 WebSphere MQ Sensor, you must first uninstall the previous version. See the installation instructions for your specific platforms for more information. You may **not** use TransactionVision 4.2 Sensors with the TransactionVision 4.1 Analyzer and Web User Interface.

- For components other than the WebSphere MQ Sensors, the installation utility will uninstall TransactionVision 4.1 components before installing TransactionVision 4.2 components. Before uninstalling, however, it will provide options for backing up and migrating configuration files to the new installation. See the component installation instructions for more information. To migrate projects created with an earlier release, use the **`MigrateDB`** utility after installing TransactionVision. For information about this utility, see Appendix A.

## Software Requirements

Before installing the packages, make sure the systems you are installing on meet the software requirements for TransactionVision. Requirements vary for each package (Sensors, Analyzer, and Web User Interface). In addition, make sure your system meets requirements for database management systems, web browsers, Java, WebSphere Business Integration Message Broker (optional), and LDAP. See the TransactionVision Release Notes for detailed software requirements.

## Installing TransactionVision on Windows

### Installing the Analyzer and Web User Interface on Windows

The TransactionVision Analyzer and web user interface are installed as a single package on Windows. Note that you must be logged into the target system either as Administrator or as a user with Administrator privileges.

*Important!* For migration to work properly on Windows when you are upgrading from a previous release, either the JAVA_HOME environment variable must be set or Java must be present in your path **before** you begin the installation. Migration requires Java; if neither of these is set, the installation may fail to migrate configuration files and you may need to run **`TVisionSetupInfo`** manually.

To install this package, perform the following steps:

1. Close all Windows programs currently running on your computer.
2. In the Windows Explorer, double-click tvision_421_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 (C:\TEMP\Bristol TransactionVision), 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 components on this computer, continue with "Initial Installation." If an earlier version of TransactionVision is installed on this computer, continue with "Upgrade Installation."

### Initial Installation

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

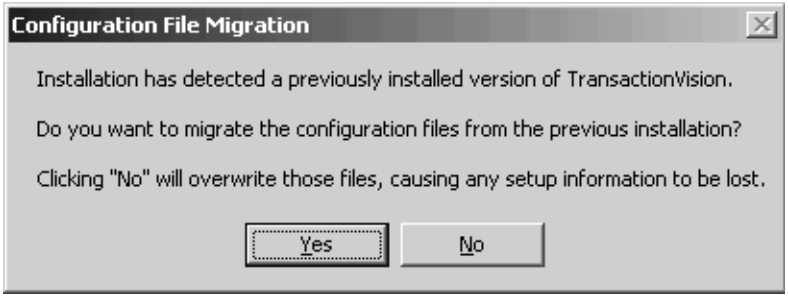
1. On the Setup Welcome screen, click Next> to display the TransactionVision license agreement.
2. Click Yes to accept the license agreement. The User Information screen appears.
3. Enter your name and company name, then click Next>. The Destination Location screen appears.
4. To use the default installation folder (C:\Program Files\Bristol TransactionVision), click Next>. To choose a different installation folder, click Browse..., select the desired installation folder, then click Next>. The Setup Type screen appears.
5. To install both the Analyzer and web user interface, select Complete and click Next>. To install only the Analyzer, select Custom, click Next>, deselect the web user interface, and click Next>.

   The selected packages are installed in the specified location. The Setup Complete page appears.
6. Click Finish to complete the installation.
7. If you are running WebLogic as a Windows service, continue with the steps in "Modifying the Registry for WebLogic."

### Upgrade Installation

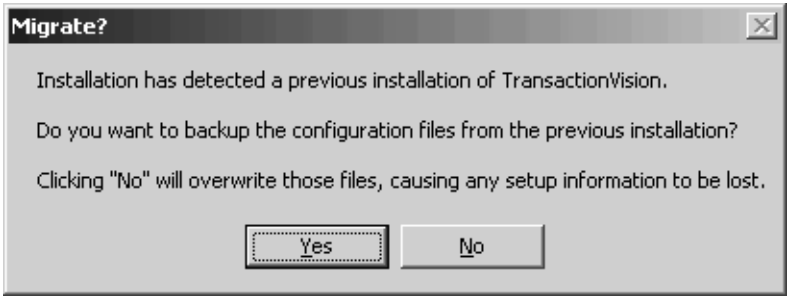
For an upgrade installation, the installation wizard displays the setup maintenance menu.

1. If you wish to install TransactionVision with **different** settings from the previous installation, select Remove and click Next> to uninstall the previous installation, then begin the installation procedure again. If you are upgrading from a previous release, select Reinstall and click Next> to install TransactionVision using the settings from the previous installation. The Configuration File Migration dialog appears:




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

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




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

3. Click Finish to complete the installation. If you wish to display the migration log file created when migrating configuration files to the new version, check View Migration Log File before clicking Finish.

4. If you are running WebLogic as a Windows service, continue with the steps in "Modifying the Registry for WebLogic."

### Modifying the Registry for WebLogic

If the WebLogic application server is running as a Windows service you must modify the registry after installing the TransactionVision web user interface.

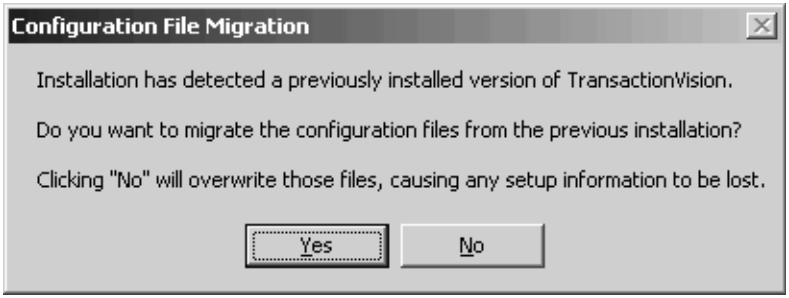
1. Using **`regedit`**, navigate to the `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\beasvc <domain_name>_<server_name>\Parameters` entry.
2. Add the following to the `CmdLine` value, replacing <TVISION_HOME> with your particular TransactionVision installation folder pathname:

   ```
   -Dcom.bristol.tvision.home=<TVISION_HOME>
   ```

### Modifying the Installation

After you install TransactionVision components on a host, you may wish to modify your installation. For example, suppose you initially install the Analyzer on a host, then later decide to install the Web user interface. To modify your installation perform the following steps:

1. Close all Windows programs currently running on your computer.
2. In the Windows Explorer, double-click tvision_421_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 (C:\TEMP\Bristol TransactionVision), click Next>. To choose a different folder, click Change, select the desired folder, then click Next>. InstallShield extracts the installation files and displays the maintenance menu.
5. To install an additional component or remove an installed component, select Modify. To reinstall all components installed by the previous setup, select Reinstall. To uninstall all components, select Remove.
6. Click Next> and select the additional components you wish to install. The Configuration File Migration dialog appears:




7. Click No, because migration is not required for installing additional components. 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 No again to continue the installation without backing up configuration files. The installation wizard then installs the selected TransactionVision components and displays the Setup Complete screen.
8. Click Finish to complete the installation. If you wish to display the migration log file created when migrating configuration and project database files to the new version, check View Migration Log File before clicking Finish.

## Installing Sensors on Windows

The TransactionVision Application Server, JMS, and WebSphere MQ Sensors are installed as a single package on Windows. Note that you must be logged into the target system either as Administrator or as a user with Administrator privileges. To install this package, perform the following steps:

1. Close all Windows programs currently running on your computer.
2. In the Windows Explorer, double-click tvision_sensor_421_win.exe. The InstallShield Welcome screen appears.
3. Click Next> to display the InstallShield Save Files screen.
4. To use the default folder for extracting installation files, click Next>. To choose a different folder, click Change, select the desired folder, then click Next>. InstallShield extracts the installation files.

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

### Initial Installation

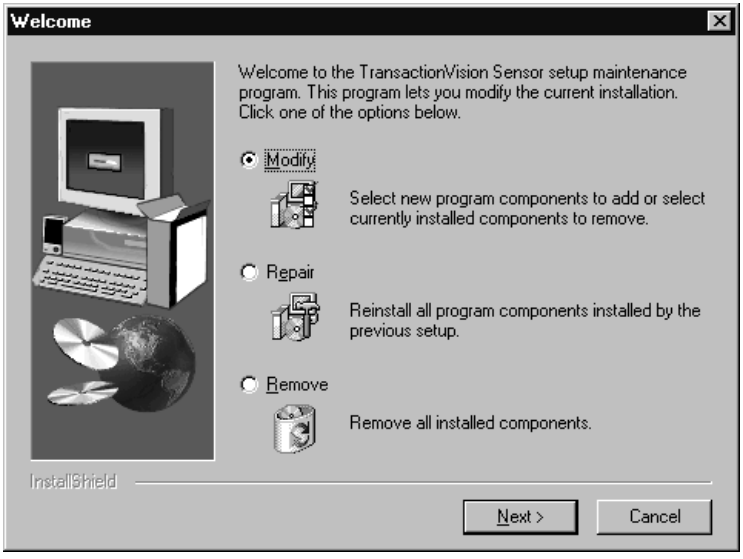
For an initial installation, the Setup Welcome screen is displayed.

1. On the Setup Welcome screen, click Next> to display the TransactionVision license agreement.
2. Click Yes to accept the license agreement. The User Information screen appears.
3. Enter your name and company name, then click Next>. The Destination Location screen appears.
4. To install the Application Server, JMS, and WebSphere MQ Sensors, select Complete and click Next>. To install only some Sensors, select Custom, click Next>, select the desired Sensors, and click Next>.

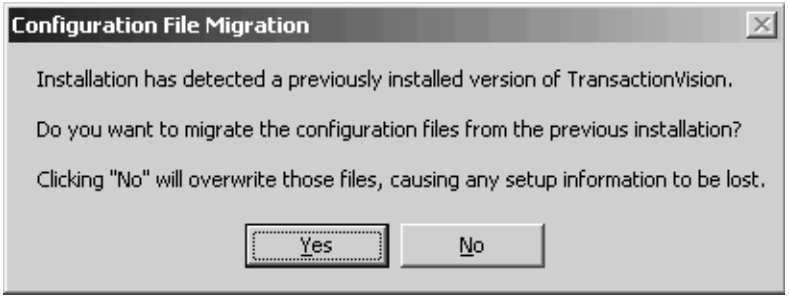
   The selected Sensors are installed in the specified location. The Setup Complete page appears.
5. Click Finish to complete the installation.
6. If you are running WebLogic as a Windows service and installed the Servlet and/or EJB Sensor, continue with the steps in "Modifying the Registry for WebLogic."

### Upgrade Installation

For an upgrade installation, the installation wizard displays the Sensor setup maintenance menu:

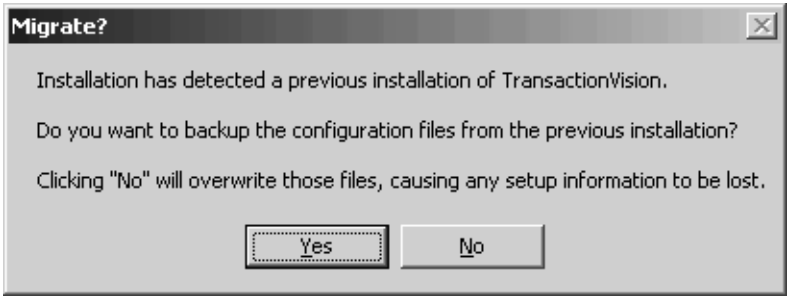


1. If you wish to install TransactionVision Sensors with **different** settings from the previous installation, select Remove and click Next> to uninstall the previous installation, then begin the installation procedure again. If you are upgrading from a previous release, select Repair and click Next> to install TransactionVision Sensors using the settings from the previous installation. The Configuration File Migration dialog appears:




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

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




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

3. The installation wizard installs the new version of TransactionVision and displays the Setup Complete screen.

4. Click Finish to complete the installation.

5. If you are running WebLogic as a Windows service and installed the Servlet and/or EJB Sensor, continue with the steps in "Modifying the Registry for WebLogic."

### Modifying the Registry for WebLogic

If the WebLogic application server is running as a Windows service you must modify the registry after installing the TransactionVision Servlet and/or EJB Sensor.

1. Using **`regedit`**, navigate to the `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\beasvc <domain_name>_<server_name>\Parameters` entry.

2. Add the following to the `CmdLine` value, replacing <TVISION_HOME> with your TransactionVision installation path:

```
-Dcom.bristol.tvision.home=<TVISION_HOME>
-Dweblogic.classloader.preprocessor="com.bristol.
tvision.sensor.servlet.WebLogicClassPreProcessor"
-Dcom.bristol.tvision.sensor.disableApps=
"TransactionVision,console,wl_management_internal1,
wl_management_internal2,uddi,uddiexplorer"
```

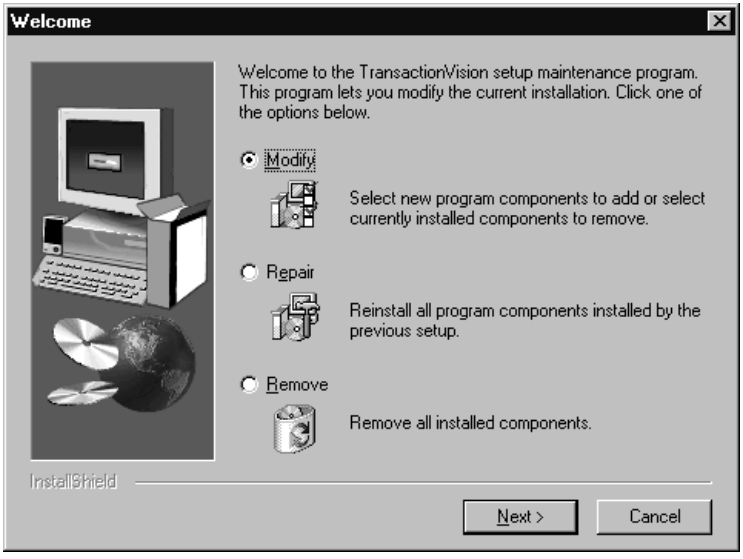
3. Prepend the following classes to the classpath, replacing <TVISION_HOME> with your TransactionVision installation path and <WMQ_HOME> with your WebSphere MQ installation path:

```
<TVISION_HOME>\java\lib\tvisionsensorservlet.jar;<TVISI
ON_HOME>\java\lib\tvisionsensorejb.jar;<TVISION_HOME>\j
ava\lib\tvjavaext.jar;<TVISION_HOME>\java\lib\weblogic.
jar;<TVISION_HOME>\java\lib\webservices.jar;<TVISION_HO
ME>\java\lib\tvisionsensorjms.jar;<TVISION_HOME>\java\l
ib\com.ibm.mqjms.jar;<TVISION_HOME>\java\lib\xercesImpl
.jar;<TVISION_HOME>\java\lib\xmlParserAPIs.jar;<WMQ_HOM
E>\Java\lib;<WMQ_HOME>\Java\lib\com.ibm.mqjms.jar;<WMQ_
HOME>\Java\lib\fscontext.jar;<WMQ_HOME>\Java\lib\provid
erutil.jar;<WMQ_HOME>\Java\lib\connector.jar;<WMQ_HOME>
\Java\lib\jta.jar;<WMQ_HOME>\Java\lib\com.ibm.mqbind.ja
r;<WMQ_HOME>\Java\lib\com.ibm.mq.jar;
```

### Modifying the Installation

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

1. Close all Windows programs currently running on your computer.
2. In the Windows Explorer, double-click tvision_sensor_421_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 and displays the maintenance menu:




5. To install an additional Sensor or remove an installed Sensor, select Modify. To reinstall all Sensors installed by the previous setup, select Repair. To uninstall all Sensors, select Remove.
6. Click Next> and select the Sensor packages to install.
7. Click Finish to complete the installation.

## Installing TransactionVision on Distributed Platforms

To install the TransactionVision components on distributed platforms, perform the following steps. Note that TransactionVision provides a unique package for the Sensor Servlet/EJB, JMS, and WebSphere MQ Sensors, as well as the Analyzer and Web User Interface.

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

   ```
   su
   ```

3. Enter the following command to begin the installation procedure:

   ```
   ./install.sh
   ```

   The following menu is displayed:

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

   Unziping/Untaring common package files ...
   Unziping/Untaring Analyzer package files ...
   Unziping/Untaring web package files ...
   Unziping/Untaring Application Server sensor package
   files ...
   Unziping/Untaring JMS sensor package files ...
   Unziping/Untaring WebSphere MQ sensor package files ...
   ```

**NOTE: The above six lines are only for Solaris and HP-UX installation.**

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

1. TransactionVision Analyzer
2. TransactionVision Web
3. TransactionVision Application Server Sensor
4. TransactionVision JMS Sensor
5. TransactionVision WebSphere MQ Sensor

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

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

*Important!* Note that actual options depend on the installation files available on your computer: Also note that the Application Server Sensor package installs both the Servlet and EJB Sensors.

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

### Initial Installation

1. To install only a single component, type the number associated with the TransactionVision component package and press Return.

   To install multiple, but not all components, type the numbers associated with the components you wish to install, separated by commas, and press Return.

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

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

2. To quit the installation procedure, type **q** and press Return. To install additional components, see the installation instructions for those components.

### Upgrade Installation

1. To install only the Sensor package , type the number associated with the TransactionVision Sensor and press Return.

   To install all available components, type **99** and press Return. See the installation instructions for the other components you wish to install for information about installing those components.

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

   ```
   There is an earlier version of TransactionVision
   installed on the system.
   The earlier version has to be uninstalled before
   ```

```
installing the current package(s).
Continue with the uninstallation? (Y/N) [N]:
```

2. Type **Y** and press Return to uninstall the previous version. If you type **N** and press Return, the installation process ends without uninstalling the previous version or installing the new version.

   Before uninstalling the previous version, TransactionVision provides the option of migrating configuration files and project database schemas to the new installation:

   ```
   Installation has detected an previous installation of
   TransactionVision.
   Migration converts configuration files and database
   table schemas from the previous installation to the new
   installation.
   This step is required for TransactionVision to work
   correctly with existing projects generated by an
   earlier version of TransactionVision.
   Continue to migrate TransactionVision project tables
   and backed up configuration files from the previous
   version? (Y/N) [Y]
   Answer N and the database migration step can be
   performed at a later time using the MigrateDB.sh
   script.
   ```

3. Type **Y** and press Return to migrate configuration files and project database schemas from the previous installation to the new version. The installation script automatically creates a backup copy of existing configuration files for the migration. It then uninstalls the previous version of TransactionVision, installs the new Sensor package (and addition components, if specified), and displays the installation menu. Continue to step 4.

   If you choose not to migrate configuration files and project database schemas at this time, you may migrate project database schemas at a later time, after configuring TransactionVision components with the **`MigrateDB.sh`** script. See Appendix A, “Utilities Reference,” for information about this utility.

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

   ```
   Back up the configuration files from the previous
   installation? (Y/N) [Y]
   ```

```
Answer N will over write those files, causing any setup
information to be lost.
```

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

```
Backup copy of configuration files to
[/opt/TVision/migrate_tv421_date_time]:
```

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

- Copies the current configuration files to the specified directory
- Uninstalls the previous version of TransactionVision.
- Installs the new web package (and additional components, if specified).
- Displays the installation menu again.

4. To quit the installation procedure, type **q** and press Return.

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

*Important!* The JMS Sensor does not support the WebSphere Application Server 5.1 embedded JMS implementation. The embedded JMS implementation is only meant to be used internally by applications running on WebSphere Application Server; messaging into or out of the WebSphere environment is not supported. The full WebSphere MQ product is needed for this. WebSphere MQ 5.3 CSD 1 or higher can run on the same machine as WebSphere Application Server's embedded JMS provider; however early versions of WebSphere MQ cannot coexist with the embedded provider. Only WebSphere MQ 5.3 CSD 1 or later versions are supported for use with WebSphere Application Server version 5.1. For more information, see http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q=&uid=swg21105544&loc=en_US&cs=utf-8&lang=en.

### Installation Files

The following table shows the installation file names for the TransactionVision web package for each distributed platform. Note that if you install the Servlet or JMS Sensor, the common package is installed automatically because these Sensors depend on it.

| Platform | Files |
|---|---|
| **TransactionVision Analyzer** | |
| AIX | tvision_common_421_aix.bff<br>tvision_analyzer_421_aix.bff |
| Linux | tvision_common_421_linux.rpm<br>tvision_analyzer_421_linux.rpm |
| Solaris | tvision_common_421_solaris_tar.gz<br>tvision_analyzer_421_solaris_tar.gz |
| **TransactionVision Web User Interface** | |
| AIX | tvision_web_421_aix.bff |
| Linux | tvision_web_421_linux.rpm |
| Solaris | tvision_web_421_solaris_tar.gz |
| **TransactionVision Sensors** | |
| AIX | tvision_common_421_aix.bff<br>tvision_sensor_jms_421_aix.bff<br>tvision_sensor_appserv_421_aix.bff<br>tvision_sensor_wmq_421_aix.bff |
| HP-UX | tvision_sensor_wmq_421_hpux11_tar.gz |
| Linux | tvision_common_421_linux.rpm<br>tvision_sensor_jms_421_linux.rpm<br>tvision_sensor_appserv_421_linux.rpm<br>tvision_sensor_wmq_421_linux.rpm |

| Platform | Files |
|---|---|
| Solaris | tvision_common_421_solaris_tar.gz<br>tvision_sensor_jms_421_solaris_tar.gz<br>tvision_sensor_appserv_421_solaris_tar.gz<br>tvision_sensor_wmq_421_solaris_tar.gz |

## Installing TransactionVision on OS/400

To install the Sensor on the OS/400 platform, perform the following steps:

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

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

2. On an AS/400 machine, either find and existing library to use or create a new a library to copy the installation file to (for example, `TVTMP`).

3. On a PC, FTP the Sensor installation file `sensor421.savf` from the CD-ROM to the library created in step 1 on your OS/400 machine. Be sure to set binary mode transfer as follows:

   ```
   ftp> bin
   ftp> cd /qsys.lib/tvisiontmp.lib
   ftp> put sensor421.savf
   ```

4. On the AS/400 machine, run the following command to install the Sensor. Note that you may need to replace TVTMP in the command with the name of the library in which the sensor421.savf package resides

   ```
   RSTLICPGM LICPGM(3RBB9ES) DEV(*SAVF)
   SAVF(TVTMP/SENSOR421)
   ```

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.

# Installing TransactionVision on OS/390

## Installing the CICS Sensor on IBM OS/390

To install the CICS Sensor, perform the following steps, substituting a valid dataset name high-level qualifier for &hlq, for example, TVISION.

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

```
ftp> quote site fixrecfm 80 lrecl=80 recfm=fb blksize=3120
ftp> bin
ftp> put sld421.f1 '&hlq.sld421.f1'
ftp> put sld421.f2 '&hlq.sld421.f2'
ftp> put sld421.f3 '&hlq.sld421.f3'
ftp> put sld421.mcs '&hlq.sld421.mcs'
```

2. Use the TSO RECEIVE command to create the product distribution datasets from the transferred files. The following table shows the RECEIVE commands and the filename to enter for each one in response to the prompt, "INMR906A Enter restore parameters or 'DELETE' or 'END':"

| Command | Filename |
|---|---|
| RECEIVE INDSNAME('&hlq.SLD421.F1') | DSN('&hlq.ASLD421.F1') |
| RECEIVE INDSNAME('&hlq.SLD421.F2') | DSN('&hlq.ASLD421.F2') |
| RECEIVE INDSNAME('&hlq.SLD421.F3') | DSN('&hlq.ASLD421.F3') |
| RECEIVE INDSNAME('&hlq.SLD421.MCS') | DSN('&hlq.ASLD421.SMPMCS') |

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

| Dataset | Member(s) | Description |
|---|---|---|
| &hlq.ASLD421.F1 | ASLD421 | SMP/E JCLIN |
| &hlq.ASLD421.F2 | SLDMOD01-23 | CICS Sensor modules |
| | MQCONNX | Dummy module for WebSphere 5.1 installations. |

| Dataset | Member(s) | Description |
|---|---|---|
| &hlq.ASLD421.F3 | SLDACCPT | Sample SMP/E job |
| | SLDALLOC | Sample SMP/E job |
| | SLDAPPLY | Sample SMP/E job |
| | SLDCICSD | Sample customization job |
| | SLDCRTQS | Sample customization job |
| | SLDDDDEF | Sample SMP/E job |
| | SLDDZON | Sample SMP/E job |
| | SLDGZON | Sample SMP/E job |
| | SLDINSTL | Sample non-SMP/E install job |
| | SLDRECV | Sample SMP/E job |
| | SLDTZON | Sample SMP/E job |
| | TVISION | Sample product startup procedure |
| | TVISIONC | Sample product startup procedure |
| | VERSION | Build number information |
| &hlq.ASLD421.SMPMCS | SMP/E MCS | |

If any of the above datasets are missing, recheck Steps 1 and 2. If discrepancies remain unresolved, please contact Bristol Technology Support for assistance.

**Note:** A sample job is provided for each of the major installation steps. Please copy each sample job member to a private data set or member, customize it for your local requirements as directed by the member's commentary, and run it to perform the installation step. You may perform a non-SMP/E install by customizing and running the SLDINSTL member, skipping the SMP/E steps (Steps 4-8), and resuming with Step 9. Also, skip Step 17. For an SMP/E install, perform all remaining steps. The SMP/E FMID for this installation is ASLD421.

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

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

5. If you are installing the Sensor into an existing SMP/E global zone, skip this step. If you are installing the Sensor into a new SMP/E global zone, customize and run members—SLDGZON, SLDDZON, and SLDTZON—to create new global, target, and distribution zones.

6. Create the SMP/E DDDEFs in the distribution and target zones by customizing and running the SLDDDDEF member. You may install into any SMP/E target zone desired. The following DDDEFs will be created by the job:

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

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

7. SMP/E RECEIVE the Sensor installation package product by customizing and running the SLDRECV member.
8. SMP/E APPLY the Sensor installation package by customizing and running the SLDAPPLY member. After APPLY processing, verify the following:

| Library | Contents |
|---|---|
| &THLQ.SSLDLOAD | SLDPCCX, SLDPCDR, SLDPCMX, SLDPCPX, SLDPCSX, SLDPDSX, SLDPFCX, SLDPICX, SLDPPCX, SLDPPSX, SLDPTCX, SLDPTDX, SLDPTSX, MQCONNX |
| &THLQ.SSLDINST | SLDACCPT, SLDALLOC, SLDAPPLY, SLDCICSD, SLDCRTQS, SLDDDDEF, SLDDZON, SLDGZON, SLDINSTL, SLDRECV, SLDTZON, VERSION |
| &THLQ.SSLDPROC | TVISION, TVISIONC |
| &THLQ.SSLMSAMP | No members in this release |
| &THLQ.SSLDAUTH | SLDPASM, SLDPBQM, SLDPCMD, SLDPCSC, SLDPCSI, SLDPCSM, SLDPITM, SLDPMON, SLDPSSS, SLDPTVM |

9. You may wish to review Chapter 9, “Configuring the CICS Sensor,” before performing the remaining installation steps.
10. Update your CICS CSD file with the required resource definitions for the Sensor by customizing and running the SLDCICSD member. If you run multiple CICS regions with separate CSDs or different startup lists, repeat this step for each CICS region to be monitored by the Sensor.
11. Place the CICS Sensor program load modules in a library in your DFHRPL concatenation. Either copy the modules into an existing library already in your DFHRPL concatenation or add the SSLDLOAD library to the DFHRPL concatenation. The CICS modules are: SLDPCCX, SLDPCMX, SLDPCPX, SLDPCSX, SLDPDSX, SLDPFCX, SLDPICX, SLDPPCX, SLDPPSX, SLDPTCX, SLDPTDX, and SLDPTSX. If you run multiple CICS regions with separate startup JCL, repeat this step for each CICS region to be monitored by the Sensor.

12. Optional. To automatically enable and disable the Sensor's CICS exit programs at CICS startup and shutdown, define SLDPCSX as a second pass PLTPI and SLDPCPX as a second pass PLTSD. Refer to *CICS Resource Definition Guide*, DFHPLT section. Add these definitions to your existing DFHPLTxx. If new PLTs are defined, add references to them in the CICS System Initialization Table (SIT). Refer to *CICS System Definition Guide*, "Specifying CICS system initialization parameters." Repeat this step for each CICS region to be monitored by the Sensor.

    Sample SIT entries:

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

    Sample defnitions of PLTs with the suffixes BT and BS:

    ```
    //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

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

    //
    ```

13. APF-authorize the TransactionVision library, &hlq.SSLDAUTH. Note that &hlq.SSLMLOAD must not be authorized. Refer to *MVS Initialization and Tuning Reference*, PROGxx or IEAAPFxx system parameters.

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

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

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

15. Customize the sample Sensor startup procedures, TVISION and TVISIONC, in the &hlq.SSLDPROC data set and copy them to an appropriate procedure library for your site. See Chapter 9, "Configuring the CICS Sensor," for guidelines for customizing the startup procedures.
16. Create appropriate Sensor configuration and event queues on the WebSphere MQ queue manage to be used to communicate between the CICS Sensor Driver component and the TransactionVision Analyzer. Customize and run the SLDCRTQS member. See Chapter 9, "Configuring the CICS Sensor," for more information.
17. When you are satisfied with the results of your installation, perform an SMP/E ACCEPT by customizing and running the SLDACCPT member.

## Installing the WebSphere MQ Sensor on IBM OS/390

There is a single Sensor installation package for WebSphere MQ CICS, batch, IMS, and MQSeries-IMS bridge Sensors on the OS/390. Some configuration steps only apply to the WebSphere MQ Sensor for CICS; they are noted as "CICS only." If you do not intend to use the WebSphere MQ Sensor for CICS, omit those steps. The MQSeries-IMS bridge Sensor also requires additional steps, which are detailed in "Additional Setup for the MQSeries-IMS Bridge Sensor" on page 53. If you do not intend to use the MQSeries-IMS bridge Sensor, omit those steps.

### Before You Install the WebSphere MQ Sensor for CICS

The WebSphere MQ Sensor for OS/390 CICS consists of two component programs. The program that monitors an application's WebSphere MQ API calls is an WebSphere MQ API crossing exit program named CSQCAPX. This program must be installed in your CICS region in order for the Sensor to collect any events. A second program, SLMC (with an associated CICS transaction of the same name), is optional. It can be run to automatically enable and disable the crossing exit Sensor program in your CICS region as needed by the Sensor. You do not have to run SLMC for the Sensor to function correctly, but doing so improves your region's WebSphere MQ application performance when no Analyzer is monitoring the region. If you choose not to run SLMC, ensure that the crossing exit is left enabled in your CICS region.

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

The WebSphere MQ Sensor for OS/390 CICS requires OS/390 Language Environment runtime support. Before installing the Sensor, be certain that your target CICS region has LE support enabled. The Sensor requires the CICS region to support LE programs compiled from the C language. The procedure for enabling LE support is described in the *CICS System Definition Guide* (the "Installing Application Programs" chapter in the CICS TS 1.2 documentation). Consult the documentation for your version of CICS for full details.

### SMPE Installation Procedure

To install the WebSphere MQ Sensor on OS/390, perform the following steps:

1. FTP the Sensor installation files from `Sensors/os390/smpe` directory of the TransactionVision CD-ROM to your OS/390 machine. Be sure to set binary mode transfer and the dataset characteristics as follows:

```
ftp> quote site fixrecfm 80 lrecl=80 recfm=fb
blksize=3120
ftp> bin
ftp> put slm421.f1
ftp> put slm421.f2
ftp> put slm421.f3
ftp> put slm421.mcs
```

*Important!* If you receive a dataset full error during this step (typically with `SLM421.F2`), use the following dataset characteristics (entered on a single line) instead, where *&YRVOL* is the volume on your system to save the installation files and *&UNIT* is the name of the unit for *&YRVOL*:

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

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

| Command | Filename |
|---|---|
| `RECEIVE INDSNAME(SLM421.F1)` | `dsn(ASLM421.F1)` |
| `RECEIVE INDSNAME(SLM421.F2)` | `dsn(ASLM421.F2)` |
| `RECEIVE INDSNAME(SLM421.F3)` | `dsn(ASLM421.F3)` |

| Command | Filename |
|---|---|
| `RECEIVE INDSNAME(SLM421.MCS)` | `dsn(ASLM421.SMPMCS)` |

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

| Dataset | Member | Description |
|---|---|---|
| `ASLM421.SMPMCS` | | SMP/E MCS file for Sensor |
| `ASLM421.F1` | `ASLM421` | JCLIN for SMP/E |
| `ASLM421.F2` | `MQCONNX` | Dummy module for MQ51 |
| | `SLMMOD01` | Sensor crossing exit program for OS/390 CICS |
| | `SLMMOD02` | SLMC crossing exit management program for OS/390 CICS |
| | `SLMMOD03` | Sensor stub module for OS/390 Batch (non-RRS) |
| | `SLMMOD04` | Sensor stub module for OS/390 Batch (RRSI) |
| | `SLMMOD05` | Sensor stub module for OS/390 Batch (RSTB) |
| | `SLMMOD06` | Sensor server module for OS/390 Batch |
| | `SLMMOD07` | Sensor stub module for OS/390 CICS BTTRACE. |
| | `SLMMOD08` | Sensor stub module for OS/390 IMS |
| | `SLMMOD09` | Sensor server module for OS/390 IMS |
| | `SLMMOD10` | MQSeries-IMS bridge support module |
| | `SLMMOD11` | MQSeries-IMS bridge support module |
| | `SLMMOD12` | MQSeries-IMS bridge support module |
| | `SLMMOD13` | MQSeries-IMS bridge support module |
| | `SLMMOD14` | MQSeries-IMS bridge support module |
| | `SLMMOD15` | MQSeries-IMS bridge support module |
| | `SLMMOD16` | MQSeries-IMS bridge support module |

| Dataset | Member | Description |
|---|---|---|
| ASLM421.F2 (continued) | SLMMOD17 | MQSeries-IMS bridge support module |
| | SLMMOD18 | MQSeries-IMS bridge support module |
| | SLMMOD19 | MQSeries-IMS bridge support module |
| | SLMYIOE0 | MQSeries-IMS bridge support module |
| | AIBTDLI | Dummy module for IMS |
| | BTMQEXIT | User Exit |
| | DFHEAI | Dummy module for CICS |
| | DFHEAI0 | Dummy module for CICS |
| | DFHEAII | Dummy module for CICS |
| | SLMBCNFG | Configuration queue table |
| ASLM421.F3 | SLMALLOC | Sample JCL for TransactionVision SMP/E target and distribution library allocation |
| | SLMRECV | Sample JCL for SMP/E Receive step |
| | SLMDDDEF | Sample JCL for creating SMP/E DDDEFs for TransactionVision |
| | SLMAPPLY | Sample JCL for SMP/E Apply step |
| | SLMCICSD | Sample JCL for updating CICS CSD file with Sensor program definition |
| | SLMACCPT | Sample JCL for SMP/E Accept step |
| | SLMCRTQS | Sample JCL to create TransactionVision communication queues |
| | SLMGZON | Sample JCL to create the SMP/E global zone |
| | SLMDZON | Sample JCL to create the SMP/E distribution zone |
| | SLMTZON | Sample JCL to create the SMP/E target zone |
| | BTMQEXIT | Header file for Sensor user exits |
| | BTTRACE | Header file for Sensor user tracing |

| Dataset | Member | Description |
|---|---|---|
| ASLM421.F3 (continued) | SLMEXITD | Dummy C source file for Sensor user exits |
| | SLMEXITS | C source file of the Sensor user exits sample |
| | SLMTSAMP | C source file of the Sensor user tracing sample |
| | SLMLKSTB | Sample JCL to section editing user WebSphere MQ application to link to Sensor |
| | SLMEJCL | Sample JCL to compile/bind the Sensor user exits sample |
| | SLMTJCL | Sample JCL to compile/bind the Sensor user tracing sample |
| | SLMTRJCL | Sample JCL to run the Sensor user tracing sample |
| | SLMCONFG | Macro to change the TransactionVision configuration queue name on OS/390 batch and IMS |
| | SLMBCNFS | Sample code to change the Transaction-Vision configuration queue name on OS/390 batch and IMS |
| | SLMBCNFG | Sample JCL to change the Transaction-Vision configuration queue name on OS/390 batch and IMS |
| | SLMBFGQ | Sample job to change the default TransactionVision configuration queue name for the batch and IMS Sensors or the MQSeries-IMS bridge Sensor |
| | SLMCPYDC | Sample job to copy the CICS dummy library members |
| | SLMCPYDI | Sample job to copy the IMS dummy library member |
| | TVISIOND | Sample procedure to start the MQSeries-IMS bridge Sensor event dispatcher address space |
| | TVISIONB | Sample procedure to start the MQSeries-IMS bridge Sensor control address space |
| | VERSION | Text file containing TransactionVision build number information |

If any of the above datasets are missing, recheck Steps 1 and 2. If in doubt, contact Bristol Technology support for assistance.

Note that sample JCL is provided for each of the major installation steps. You can copy this JCL to another dataset, edit it appropriately for your local requirements, and use it to perform the installation.

The SMP/E FMID for this installation is ASLM421.

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

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

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

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

   **Note:** If you do NOT have CICS installed on your system, set the CICS high level qualifier (&CICSQUAL) to be the same as the target library high level qualifier (&THLQUAL). If you do NOT have IMS installed on your system, set the IMS high level qualifier (&IMSQUAL) to be the same as the target library high level qualifier (&THLQUAL.SDFSRESL).

   The following DDDEFs will be created by the job:

| Library | Description |
|---|---|
| target zone DDDEF SSLMLOAD | Points to the SSLMLOAD target library allocated in Step 4. |
| target zone DDDEF SSLMINST | Points to the SSLMINST target library allocated in Step 4. |
| target zone DDDEF ASLMMOD | Points to the ASLMMOD distribution library allocated in Step 4. |
| target zone DDDEF SCEELKED | Points to the LE SCEELKED library. This DDDEF might already exist in your chosen target zone. |
| target zone DDDEF SCSQLOAD | Points to the WebSphere MQ SCSQLOAD library. This DDDEF might already exist in your chosen target zone. |
| target zone DDDEF SDFHLOAD | Points to the CICS SDFHLOAD library. This DDDEF might already exist in your chosen target zone. |
| target zone DDDEF SSLMINCL | Points to the SSLMINCL target library allocated in Step 4. |

| Library | Description |
|---|---|
| target zone DDDEF SSLMSRC0 | Points to the SSLMSRC0 target library allocated in Step 4. |
| target zone DDDEF SSLMSAMP | Points to the SSLMSAMP target library allocated in Step 4. |
| target zone DDDEF SSLMPROC | Points to the SSLMPROC target library allocated in Step 4. |
| target zone DDDEF SSLMAUTH | Points to the SSLMQUTH target library allocated in Step 4. |
| target zone DDEF SCEESPC | Points to the LE SCEESPC library. This DDDEF might already exist on your system. |
| distribution zone DDDEF ASLMMOD | Points to the ASLMMOD distribution library allocated in Step 4. |
| distribution zone DDDEF ASLMINST | Points to the ASLMINST distribution library created in Step 4. |
| distribution zone DDEF ASLMINCL | Points to the ASLMINCL distribution library created in Step 4. |
| distribution zone DDEF ASLMSRC0 | Points to the ASLMSRC0 distribution library created in Step 4. |
| distribution zone DDEF ASLMSAMP | Points to the ASLMSAMP distribution library created in Step 4. |
| distribution zone DDEF ASLMPROC | Points to the ASLMPROC distribution library created in Step 4. |

7. (No CICS installed only) Perform this step only if you do NOT have CICS installed on your system. Copy the CICS dummy stubs to the TVISION library. Sample JCL for this purpose is provided in the SLMCPYDC member. You must tailor this job for the requirements of your installation before executing it.
8. (No IMS installed only) Perform this step only if you do NOT have IMS installed on your system. Copy the IMS dummy stubs to the TVISION library. Sample JCL for this purpose is provided in the SLMCPYDI member. You must tailor this job for the requirements of your installation before executing it.

9. SMP/E RECEIVE the Sensor installation package. Sample JCL for this purpose can be found in the SLMRECV member. You must tailor this job for the requirements of your installation before executing it, changing the &DSNPREFIX and &GZONECSI parameters with your editor. Set the &DSNPREFIX parameter to account for the high level qualifier under which you created the ASLM421.* datasets. For example, if the datasets were installed in USERNAME.ASLM421.* and your global CSI dataset was MY.GLOBAL.CSI, your customized JCL for the SMP/E RECEIVE might look as follows:

```
//RECEIVE EXEC PGM=GIMSMP,REGION=4096K
//SMPCSI DD DSN=MY.GLOBAL.CSI,
//          DISP=SHR
//SMPPTFIN DD DSN=USERNAME.ASLM421.SMPMCS,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SMPCNTL DD *
SET BDY(GLOBAL).
RECEIVE SELECT(ASLM421) RFPREFIX(USERNAME) SYSMODS LIST.
/*
```

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

| Library | Contents |
|---|---|
| &THLQUAL.SSLMLOAD | CSQCAPX, MQCONNX, SLMBRRSI, SLMBRSTB, SLMBSTUB, SLMBSRV, SLMBSRV2, SLMBSRV3, SLMC, SLMCSTUB, SLMQSRV, SLMQSTUB, CSQBBACK, CSQBCLOS, CSQBCOMM, CSQBCONN, CSQBCONX, CSQBDISC, CSQBGET, CSQBINQ, CSQBOPEN, CSQBPUT, CSQBPUT1, CSQBSET, SLMXSRV, BTMQEXIT, SLMBCNFG |
| &THLQUAL.SSLMINST | SLMACCPT, SLMALLOC, SLMAPPLY, SLMCICSD, SLMCRTQS, SLMDDDEF, SLMDZON, SLMGZON, SLMRECV, SLMTZON, SLMCPYDC, SLMCPYDI, VERSION, SLMINSTL |
| &THLQUAL.SSLMINCL | BTMQEXIT, BTTRACE |
| &THLQUAL.SSLMPROC | SLMEJCL, SLMLKSTB, SLMTJCL, SLMTRJCL, TVISIONB, SLMBCFGQ, TVISIOND |

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

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

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

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

12. (CICS only) Place the CSQCAPX and SLMC load modules in a library within your DFHRPL concatenation. You can either copy the modules into an existing library already present in your DFHRPL concatenation or you can add the SSLM-LOAD library to the DFHRPL concatenation.

13. (CICS only) Enable the WebSphere MQ API crossing exit within your CICS region using the CKQC transaction. You can do this from the Connection > Modify > Enable API Exit menu item in the CKQC panel, or by specifying arguments to CKQC when invoking it. For example, you could enter `CKQC MODIFY N E` to enable the crossing exit (E for enable, D for disable).

14. Create appropriate Sensor configuration and event queues on the queue manager. Sample JCL for this purpose can be found in the SLMCRTQS member. You must tailor this job to the requirements of your installation before running it. See Chapter 6, “Configuring WebSphere MQ Sensors,” for more information about Sensor configuration and event queues.

15. When you are satisfied with the results of your installation, run the SMP/E ACCEPT step. Sample JCL for this purpose can be found in the SLMACCPT member. You must tailor this job to the requirements of your installation before running it.

## Configuring SLMC for CICS

SLMC is an optional Sensor component that can improve the WebSphere MQ application performance of programs run in your CICS region when the Analyzer is not monitoring the region. SLMC monitors the TransactionVision configuration queue for messages from the Analyzer, examines any messages present there, and automatically enables or disables the WebSphere MQ crossing exit mechanism as needed by the Analyzer. Disabling the crossing exit when the Analyzer is not actively monitoring the region (when no configuration messages are present or when configuration messages do not apply to the CICS region or host) improves application performance.

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

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

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

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

```
SLMC\
CESF LOGOFF\
```

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

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

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

### Additional Setup for the MQSeries-IMS Bridge Sensor

The MQSeries-IMS bridge Sensor is implemented as three components:

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

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

1. TVISIONB requires one system linkage index (LX), which it reserves the first time it is started after an IPL and thereafter reuses. Refer to the IBM OS/390 or z/OS MVS Initialization and Tuning Reference, IEASYSxx (System Parameter List) NSYSLX=nnn parameter.
2. The TransactionVision library, thlqual.SSLMAUTH, must be APF-authorized. Note that thlqual.SSLMLOAD must not be authorized. Refer to MVS Initialization and Tuning Reference, PROGxx or IEAAPFxx system parameters.
3. The TVISIONB address space must be non-swappable. TVISIONB assures this by issuing a SYSEVENT TRANSWAP macro, so it is not necessary to include the program in the PPT. If you do include it, specify the entry as follows:

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

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

For instructions on completing Sensor setup and operating the MQSeries-IMS bridge Sensor, see Chapter 6, “Configuring WebSphere MQ Sensors.”

## Licensing

To use TransactionVision, you must supply a license code provided to you by Bristol Technology. Two types of licenses are available:

- Evaluation licenses expire after a set time period (typically 30 days). When your evaluation license expires, you will no longer be able to start the Analyzer.
- Enterprise licenses do not expire.

If you do not have a license code, contact Bristol Support. After installing TransactionVision, enter your license code (or upgrade from an Evaluation license to an Enterprise license) in either of the following ways:

- Run the **`TVisionSetupInfo`** script and enter the license code when prompted. For more information on using this script, see Chapter 5, "Configuring the Analyzer and Web User Interface."
- Edit the `<TVISION_HOME>/config/license/License.properties` file and modify the following line with your license code:

  ```
  licenseCode=your-license-code-here
  ```

*Important!* After changing you license code with either of these methods, you must restart the TransactionVision Analyzer service and the TransactionVision web user interface for the new license code to take effect.

## Uninstalling TransactionVision

Instructions for uninstalling TransactionVision components are different for Windows platforms and distributed platforms (AIX, HP-UX, Linux, Solaris, and Tru64).

### Distributed Platforms

To uninstall TransactionVision components, perform the following steps:

1. Login as superuser:

   ```
   su
   ```

2. Enter the following command:

   ```
   install.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 Application Server Sensor
   4. TransactionVision JMS Sensor
   5. TransactionVision WebSphere MQ Sensor

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

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

3. Type the number associated with the TransactionVision package you wish to uninstall and press Return.

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

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

   If you uninstall the servlet Sensor, it will be first turned off in the WebSphere Application Server the Sensor is monitoring. The instrumented jar files under `$WAS_HOME/classes` will be removed.

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

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

### Windows

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 Bristol TransactionVision package you wish to uninstall and click Change/Remove. The maintenance menu screen appears.
4. Select Remove and click Next> to remove TransactionVision components.
5. Click OK to confirm that you wish to uninstall the specified package. The specified package is uninstalled. The following types of files are **not** deleted:

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

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

   - To leave all shared files installed, check Don't display this message again and click No.

- To leave the current file, but display this message for any other shared files, click No.
- To delete the shared file, click Yes.

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

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

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

# Chapter 4
# Configuring Databases

Before configuring TransactionVision components, it is important to ensure that your DBMS environment is configured properly to work with TransactionVision. This configuration differs for DB2 and Oracle databases. It is first necessary to create your DB2 or Oracle database before running the **`TVisionSetup`** script. The **`TVisionSetup`** script does not create a database and relies on a connection to an existing database. Database variables or tuning parameters must then be adjusted to ensure that the database will operate at an acceptable level of performance for your application. Several tools are provided with TransactionVision to aid in the identification and resolution of database performance bottlenecks.

## Setting DB2 Variables

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

| Variable | Value | Description |
|---|---|---|
| APP CTL HEAP SZ | 1024 | Maximum application control heap size. This value indicates the number of 4KB blocks. |
| MAXAPPLS | 150 | Maximum number of active applications |
| APPLHEAPSZ | 1024 | Default application heap size. this value indicates the number of 4KB blocks. |

| Variable | Value | Description |
|---|---|---|
| LOCKLIST | Estimate based on system | Amount of storage (in 4KB units) allocated to the lock list. |

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

### APP CTL HEAP SZ, MAXAPPLS, and APPLHEAPSZ

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

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

In the environment in which WebSphere is started on UNIX systems, the DB2INSTANCE environment variable must also be set to the DB2 instance being used by TransactionVision.

### LOCKLIST

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

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

   ```
   (512 * 36 * mapappls) / 4096
   ```

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

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

   ```
   (512 * 72 * maxappls) / 4096
   ```

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

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

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

   where *n* is the value for LOCKLIST.

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

### DB2_RR_TO_RS

Multiple threads performing concurrent database operations along with a DB2 behavior called Next Row Locking may lead to database deadlocks. If you plan to use multiple event collection threads, set the DB2 variable DB2_RR_TO_RS to ON to avoid deadlocks:

```
db2set DB2_RR_TO_RS=ON
```

This setting is only effective after a DB2 restart. It affects all DB2 applications and cannot be used if other non-TransactionVision applications that require “Repeatable Read” semantics (Transaction Isolation Level RR) are using the same DB2 instance. If other applications using the same instance require “Repeatable Read” semantics, you must either create a separate DB2 instance for TransactionVision or set the number of event collection threads to 1. For information on setting the number of event collection threads, see the “Managing Communication Links” chapter in the *TransactionVision Administrator’s Guide*.

### Bufferpool

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

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

## Setting Oracle Variables

If it is expected that TransactionVision will be used in a relatively simple environment where minimal database connections are required, no special configuration is required for the Oracle DBMS. However, environments where a large number of users will be

simultaneously accessing the web user interface, several Analyzers will be in use, or where the Analyzer will incorporate higher than the default number of threads, it will be necessary to increase the Open Cursors database parameter. Related error messages may be expected to show up in the TransactionVision UI or Analyzer logs if this limit is exceeded. To increase the number of Open Cursors, execute the following command:

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

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

## Connecting to an Oracle Database

To connect to an Oracle database, perform the following steps:

1. If you are planning to use the Oracle oci driver to make database connections, make sure the target database is defined in your `tnsnames.ora` file. For example:

   ```
   TESTDB_SIMPLEX6.BRISTOL.COM =
     (DESCRIPTION =
       (ADDRESS_LIST =
         (ADDRESS = (PROTOCOL = TCP)(HOST = simplex6)(PORT
   = 1521))
       )
       (CONNECT_DATA =
         (SID = testdb)
         (SERVER = DEDICATED)
       )
     )
   ```

   Various Oracle tools can create such an entry for you, or you can add it manually.

2. Run the **`TVisionSetupInfo`** script (see Chapter 5, "Configuring the Analyzer and Web User Interface," for instructions) and enter Oracle when you are prompted to enter the type of database to use, enter desired client type, and (for thin clients) the database listener port. For more information about these prompts, see page 68.

   This script updates the `Database.properties` file based on the settings you select. For example, for a thin client connection, the `Database.properties` file will contain entries similar to the following example:

   ```
   jdbc_driver=oracle.jdbc.driver.OracleDriver
   database_name=TESTDB_SIMPLEX6
   ```

```
database_host=simplex6.bristol.com
database_port=1521
oracle_client_type=thin
```

For more information about these entries, see “Database.properties” on page 201.

## DBMS Performance Tuning

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

### Optimizing I/O Throughput

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

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

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

- Use separate DBMS instances for separate projects - if the data can be partitioned then separate DBMS instances on different hosts can be employed to achieve parallelism. This setup requires setting up multiple instances of TransactionVision.
- Use RAID disks for table space containers. RAID disks provide parallel I/O at hardware level.
- Separate table space containers and log file directories. DB2 log files and Oracle Rollback Segments 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” 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

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

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

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

- Run the insert test with one thread and with record size of 1KB - this will gauge the raw event insert performance.
- Run the insert test with multiple threads and with a record size of 1KB. This will test whether the insert can benefit from multiple threads. Usually the thread count is set to two times the number of CPUs.
- Run the insert test with one thread and with record size of (13KB + average message size) - this will gauge the analyzed event insert performance.
- Run the insert test with multiple threads and with record size of (13KB + average message size). This will test if the insert can benefit from multiple threads. Usually the thread count is set to two to four times the number of CPUs.
- If the Analyzer host and the DBMS host are different, the above tests should be run on the Analyzer host. However at least one test should be run on the DBMS host to see if there is any communication/DB client configuration related issues.

The rate of insert should be on par with the result achieved from similar systems tested by Bristol. 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 Bristol for any configuration issues.

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

## Updating Statistics with RUNSTATS

The database RUNSTATS command updates statistics about the physical characteristics of a table and the associated indexes. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

This command is called when a table has had many updates, such as when data is continuously collected into DB2 by the TransactionVision Analyzer. It could result in large performance gains in queries made by TransactionVision views and reports, as well as queries made internally by the TransactionVision Analyzer to correlate events.

TransactionVision provides the following scripts for running the RUNSTATS command on TransactionVision tables:

- **`DB2RunStats.[sh|bat]`** for a DB2 environment
- **`OracleRunStats.[sh|bat]`** for an Oracle environment

These scripts can be set up to run as scheduled batch jobs using either the UNIX **`cron`** facility or the Windows scheduler.

*Important!* While these scripts are running, TransactionVision Analyzer processing slows down.

These scripts typically should be run daily, though the frequency of execution could be higher for higher message rates.

Run these scripts from a user account that has privileges to perform database operations. The usage is as follows:

```
DB2RunStats database_name schema_name [-v7]
OracleRunStats username passwd database_name schema_name
```

The *database_name* option represents the name of the database to connect to, and *schema_name* is the name of the schema that the project reads and writes data to. For **`DB2RunStats`**, use the `-v7` flag for DB2 7.x; the default operation is for DB2 8.1.

*Important!* These scripts need to be customized based on your system to invoke the right environment initialization script like .profile or .bashrc, and to set the correct DB2 installation location and the correct TVISION_HOME location. For **`OracleRunStats`**, the file `OracleRunStats.sql` **must** be in the current directory before the script is executed.

### Example cron Job

For a UNIX environment, the following example is a typical crontab entry for executing the **`DB2RunStats`** script every night:

```
5 0 * * * <TVISION_HOME>/bin/DB2RunStats.sh database-name
schema-name > <TVISION_HOME/logs/runstats.log 2>&1
```

In this example, <TVISION_HOME> is the TransactionVision installation directory, *database-name* is the database name to connect to, and *schema-name* is the schema on the tables to run RUNSTATS on.

The above entry executes the script every night 5 minutes after midnight. A crontab entry needs to be added for every project schema being used by TransactionVision. Use **`crontab -e`** to create a cron job entry. For details, refer to the man pages of cron and crontab.

### Example Scheduled Task

For a Windows environment, run these scripts through Windows scheduler at regular intervals. Add a scheduler entry for every project schema being used by TransactionVision. The Windows scheduler can be accessed through the Windows Control Panel. Make sure this script runs in the DB2 command window.

The following is an example command for the Windows scheduler to run the **`DB2RunStats`** script:

```
"<DB2-install-dir>\bin\db2cmd.exe" "<TVISION_HOME>\bin\
DB2RunStats.bat" database-name schema-name > "<TVISION_
HOME>\logs\runstats.log"
```

In this example, <TVISION_HOME> is the TransactionVision installation directory, *database-name* is the database name to connect to, and *schema-name* is the schema on the tables to run RUNSTATS on.

## DBMS Disk Space Requirements

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

The formula for calculating disk storage usage (for DB2) is:

(Average message size + 13K Byte) x Event Rate x Event Retention Time

For example, if the average message size is 2K Bytes, the transaction rate is 5 transactions/second (for 8 hours/day and all weekdays, this translates into 720 thousand transactions/week). If there are two collecting points for each transaction. and if TransactionVision data is required to be stored for duration of four weeks before the data is either archived or deleted, then the total required storage is about 90GB ( (2 + 13)K Bytes x 720,000 x 2 x 4 = 90G Bytes).

Using the DB2 compact LOB feature and/or the Analyzer compression function may reduce this number by 30% - 70%.

Chapter 5

# Configuring the Analyzer and Web User Interface

Once TransactionVision is installed,  you must create an IBM DB2 or Oracle database for TransactionVision to save event data. Refer to Chapter 4," Configuring Databases," for the required database parameters. The TransactionVision setup utilities do not create the database.

After the database is created, the following setup tasks must be completed before you can use TransactionVision Analyzer and web user interface:

- Create the TransactionVision database schema.
- Start the Analyzer service.
- Install and start the TransactionVision web application in your application server.
- Initialize LDAP entries for TransactionVision (optional).

TransactionVision provides two utilities to guide you through the setup process:

- The **`TVisionSetupInfo`** utility enables you to enter information it needs for the setup process and sets required environment variables.
- The **`TVisionSetup`** utility uses information you provide to **`TVisionSetupInfo`** to perform setup tasks.

You may follow the instructions in "Configuring Application Servers for TransactionVision" on page 219if you prefer to configure your application server manually.

## Step 1: Provide System Information

1. Set the TVISION_HOME environment variable to the absolute path of the installation directory. For example, on Solaris, TVISION_HOME would be `/opt/TVision`; on AIX it would be `/usr/lpp/TVision`.
2. Run the **`TVisionSetupInfo.[sh|bat]`** script. This utility modifies TransactionVision configuration files, so you must have file modification privileges to run it (typically root for UNIX or Administrator for Windows).

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

The **`TVisionSetupInfo`** utility modifies the following files:

- `TVISION_HOME/config/datamgr/Database.properties`
- `TVISION_HOME/config/ldap/Ldap.properties`
- `TVISION_HOME/install/websphere/tvision-ws-config.properties`
- `TVISION_HOME/install/weblogic/tvision-wl-config.properties`

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

- Saves the installation path for software tools in `TVISION_HOME/config/setup/DefaultInstallPath.xml`
- Generates `TVISION_HOME/bin/SetupEnv.[sh|bat]`, which is run by **`TVisionSetup`** to set the JAVA_HOME, CLASSPATH, and system library path environment variables required by TransactionVision
- Generates either the script `TVISION_HOME/bin/WSSetupEnv.[sh|bat]` to setup command line parameters for configuring TransactionVision web application in WebSphere Application Server , or `TVISION_HOME/bin/WebSetupEnv.[sh|bat]` to setup command line parameters for configuring TransactionVision web application in WebLogic Application Server.

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

*Important!* If you are using the WebLogic application server and configuring the TransactionVision web user interface on a managed server, you must run the WebLogic Admin Server and Node Manager for the TransactionVision setup utilities to work correctly. When invoking the WebLogic Admin Server and Node Manager, your database library (regardless of which database you use) must be in the system library path (PATH on Windows, LD_LIBRARY_PATH on Solaris).

## Configuring Log Files

```
Checking Java version ...

This program collects configuration information in order
to setup the TransactionVision environment. This includes:

- Location of software that TransactionVision depends on
such as WebSphere MQ, the application server and the
relational database system,

- Parameters required to connect to WebSphere MQ,

- Parameters to connect to the database system,

- Optional parameters to connect to the LDAP system

- Setup parameters for the TransactionVision Analyzer, and

- Setup parameters for the TransactionVision web
component.

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


Please specify name of directory where you want to store
your log files [C:\Program Files\Bristol\TVision\logs]:
C:\Program Files\Bristol\TransactionVision\logs
```

```
Modifying *.Logging.xml files in log file directory
C:\Program Files\Bristol\TVision\logs

0 [main] INFO AppLog  - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TVision\config\logging\Analyzer.Logging.xml"
has been successfully updated

78 [main] INFO AppLog  - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TVision\config\logging\Migrate.Logging.xml"
has been successfully updated

141 [main] INFO AppLog  - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TVision\config\logging\Setup.Logging.xml"
has been successfully updated

219 [main] INFO AppLog  - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TVision\config\logging\UI.Logging.xml" has
been successfully updated
```

### Configuring Database Properties

```
Retrieving database configuration parameters ...

Type of Database (Oracle or DB2)? [DB2]: DB2
```

If you specify Oracle as the type of database, you will be prompted to enter the following information:

- The driver to use. For Oracle 9, you may specify `thin` or `oci`; `thin` is the recommended client type.
- The port number for the Oracle listener.

For more information about using Oracle with TransactionVision, see “Connecting to an Oracle Database” on page 60.

```
Value of your environment variable DB2INSTANCE [DB2]: DB2
```

*Important!* Before starting your WebSphere 5 server, your DB2INSTANCE environment variable must be set. Failure to set this variable will cause a failure during TransactionVision startup and cause the WebSphere server to fail to start and initialize.

`Database connection name to be used by TransactionVision [TVISION]: TVDB`

Enter the name of the database connection to be used.

`Name of database TransactionVision connects to [TVISION]: TVDB`

Enter the name of the database on the server to connect to. This name may be different from the "database_connection_name" if a client database connection is used.

`Name of the host the database is running on [local_host]: devpc`

Enter the name of the host the database server is running on.

`Database user name [db2admin]:`

Enter the user name to be used while making the database connection. If this field is empty, the currently logged in user is used to make the database connection. Make sure that the user specified or currently logged in has privileges for database access.

`User password [ibmdb2]:`

Enter the password to be used while making the database connection. If this field is empty, the currently logged in user's password is used to make the database connection.

*Important!* The **`TVisionSetupInfo`** utility automatically searches for a an installed JCE provider that supports the DES encryption algorithm. If such a provider is found, **`TVisionSetupInfo`** sets the `jce_provider` value in the `Database.properties` file to the class name of the JCE provider. If no JCE provider is found, **`TVisionSetupInfo`** displays the following message:

```
Java Cryptography Extension (JCE) is not present in your
current JDK. Password encryption feature will be disabled
and stored in plain text.

2004-08-23 18:14:40,338 [main] - TransactionVision
Info(FileUpdated): File C:\Program
Files\Bristol\TVision\config\datamgr\Database.properties"
has been successfully updated
```

### Configuring TransactionVision Analyzer Settings

```
Please specify JVM flags passed on to TransactionVision
Analyzer's JVM (NOTE: If you are specifying -server flag,
it has to be the first flag) [-server -Xnoclassgc
-Xmx256m]:

Please select your Sensor types by number. If you are
selecting multiple Sensor types, separate them with ",".

        1.  CICS

        2.  EJB

        3.  JMS

        4.  Servlet

        5.  WebSphere MQ

        6.  WebSphere MQ IMS Bridge

        7.  All of the above

Your Sensor type(s) [7] : 7
```

Enter the number corresponding to the types of TransactionVision Sensors you plan to use. For multiple Sensor types, separate the numbers with a comma. For all Sensor types, enter **7**.

```
Modifying Beans.xml file in your TransactionVision
installation directory

2004-08-23 18:14:43,088 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TVision\config\services\Beans.xml" has been
successfully updated
```

### Configuring Installation Paths for Dependent Software

```
Retrieving installation path for dependent software tools
...

DB2 installation location [C:\Program Files\sqllib]:

WebSphere MQ installation location [C:\Program
Files\IBM\WebSphere MQ]:
```

```
2004-08-23 18:14:45,995 [main] - TransactionVision
Info(FileUpdated): File C:\Program
Files\Bristol\TVision\config\setup\DefaultInstallPath.
xml" has been successfully updated
```

### Selecting Web Application Server

```
Please select your web application server by number:

    1. IBM WebSphere Application Server
    2. BEA WebLogic
    3. None

Your web application server product [1]: 2
```

If you select WebLogic, the following question appears:

```
WebLogic installation location [C:\bea\weblogic81]:
enter_your_weblogic_install_directory

2004-08-23 18:14:48,745 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TVision\bin\WebSetupEnv.bat" has been
successfully updated
```

### Configuring LDAP Properties

There are two typical scenarios for a new TransactionVision installation. Either you have an existing LDAP server that you wish to integrate with, or you don't have an LDAP server and need to set one up. This section will give an overview of the most common steps considerations and setup steps for getting your LDAP server working with TransactionVision. These steps require some basic understanding of how LDAP works and focus on just giving an overview of what needs to be configured without getting into the specifics of a particular LDAP server.

See the *TransactionVision Administrator's Guide* for a description of managing LDAP users for TransactionVision.

### Getting Started with LDAP - Quick Steps to Get LDAP Up and Running with TransactionVision

For a first time installation of LDAP, which is being used solely for Transactionvision, after installing the ldap server software, decide is where you want to store the TransactionVision LDAP objects. If you havn't created one yet, a root LDAP suffix must be defined in your ldap directory structure. This name is typically (might vary slightly depending on which LDAP server you use) something in the form of 'o=yourCompanyName,c=country'. Decide a path under your defined suffix to place your TransactionVision users, and their defined security attributes. In these examples, we will put our objects in a directory called 'TVision' off our main suffix - that is 'ou=TVision,o=yourCompanyName,c=country'.

Once you have defined this suffix, and settled on a directory to place the objects, you can proceed with running the **`TVisionSetup`** and **`TVisionSetupInfo`** scripts to initialize LDAP settings. **`TVisionSetupInfo`** will ask a few questions about where TransactionVision objects are expected. When it asks about the User Directory location, give it the path to where you wish to define your users. For example: 'ou=Users,ou=TVision,o=yourCompanyName,c=country'. The Group directory location also allows for defining Transaction vision securities by the groups users belong to—this is an optional value, if it exists it will be searched for user groups, but if you do not plan on using this at least initially, you can leave it blank. See “TVisionSetupInfo Prompts” on page 76 for the **`TVisionSetupInfo`** prompts.

After defining and setting up the above, run **`TVisionSetup`** to initialize and populate LDAP with the default Transactionvision objects. The **`TVisionSetup`** steps will ask you for the username and password to login, and then the directory location under which it should create the 'ou=TVision' directory. In our example, we are just putting this off the main suffix so we enter 'o=yourCompanyName,c=country'. The next question asks if you want to create the default root suffix. This question is important because after you define the suffix in your LDAP server, you still need to create to root object itself. This step provides a shortcut for doing this—if your suffix has not yet been created, answer **Y** to this question and it will be created for you. **`TVisionSetup`** will then create all the LDAP objects.

If you use an LDAP browser to view the results, you will see it has created the default security groups (Administrator, Developer, Operator, User) in directory location 'ou=Security,ou=TVision,o=yourCompanyName,c=country'. It has also created the defaul Admin users in 'ou=Security,ou=TVision,o=yourCompanyName,c=country' , which has a TVAuthoritygroup attribute that points to the Admin Security authorization in the ou=Security directory. At this point you should be able to login to TransactionVision as the Admin user. Any further users can be added to your 'ou=Users,ou=TVision,o=yourCompanyName,c=country' directory (you will need to use your LDAP administration tools for this) to enable them to also access the web pages.

See “6. Initialize LDAP entries for TransactionVision” on page 82 for the **`TVisionSetup`** prompts

### Add or Integrate TransactionVision to an Existing Set of LDAP Users

In this scenario, it is assumed that that the suffix you wish to place TransactionVision objects under, the user directories all allready exist, and you want to enable already existing user objects to access TransactionVision. If you want to create a seperate directory location for TransactionVision from your existing user definitions, you could still follow steps similar to the previous section.

There are two tasks that need to be set to get TransactionVision to work with an existing LDAP directory. First you need to tell TransactionVision where your existing users and/or groups are located. Specify the appropriate directory location when prompted while running **`TVisionSetupInfo`** (or by modifying the corresponding attribute in you `ldap.properties` file). Then run **`TVisionSetup`** to initialize the TranactionVision LDAP entries. This script will do two things: it will add the TransactionVision LDAP schema definitions to your specified LDAP server, and it will create the ou=TVision directory object containing the default security permissions in the directory location you specify.

Second, you need to add the necessary TransactionVision attributes to the users and/or groups you wish to enable for TransactionVision. Enable any user you wish to have acceses to Transactionvision. In order for TransactionVision to recognize a user, it must have an attribute named TVAuthoritygroup which points to the TV security definition for this user. This attribute is created in the schema when **`TVisionSetup`** initialization is run. In order to give a user TransactionVision access, simply add this attribute to their user object in LDAP.

### TVisionSetupInfo Prompts

```
Retrieving LDAP configuration parameters ...

TransactionVision provides an insecure DEMO mode which
does not require an LDAP server setup. This mode provides
four default accounts: Admin, Developer, Operator and User
for logging into the system. Make sure to use this DEMO
mode only for evaluation and test purposes and not to
collect critical data.

Do you want to run TransactionVision in DEMO mode? (Y/N)
[Y]: N
```

The following prompts only appear if you answer N to the previous prompt, indicating that you do **not** want to use DEMO mode:

```
LDAP server name points to the server to connect to. Its
value should contain a URL string such as
"ldap://somehost:389".

LDAP server name [ldap://server_host:server_port]:
ldap://devpc:389
```

Enter the URL of the server to connect to. It directly maps to the attribute of `java.naming.provider.url`, used in creating an initial JNDI context.

```
User directory location specifies to the web application
what path to lookup users under for authentication and
retrieval of TransactionVision authorities.

User directory location [ou=Users,ou=TVision,o=bti,c=us]:
```

Enter the path for the TransactionVision web user interface to use to look up users for authentication and retrieval of TransactionVision authorities. If you wish to use an existing user hierarchy, use this attribute to integrate with the existing directory structure. When a user logs in, the user name is combined with the `UserDirectoryLocation` to perform authentication. For example, if user `joesmith` logs into the web application and the value of this property is `ou=Users,o=MyOrg,c=us` TransactionVision attempts to authenticate `cn=joesmith,ou=Users,o=MyOrg,c=us`.

TransactionVision creates by default a user location under `ou=Users,ou+TVision,o=bristol,c=us` if you do not wish to use an existing user directory. In order to have TransactionVision permissions, users in this directory must have a `TVAuthorityGroup` attribute. See the *TransactionVision Administrator's Guide* for more information.

```
Group directory location specifies to the web application
what path to lookup user groups under for authentication
and retrieval of TransactionVision authorities.

Group directory location [ou=Groups,o=bti,c=us]:
```

This property, similar to the user directory, specifies where to look up user groups. A group is considered to be a definition with one or more unique member attributes (such as the `groupofuniquenames` class) containing the list of users, and a `TVisionAuthorityGroup` attribute indicating what permissions to give this group.

```
2004-08-23 13:42:05,162 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TVision\config\ldap\Ldap.properties" has
been successfully updated
```

### Configuring WebLogic Application Server Properties

If you are using the WebLogic application server, the following prompts appear:

```
The following configuration parameters are required for
setting up the TransactionVision web application in
WebLogic.

 Note: The parameter values are case sensitive.

 Retrieving WebLogic configuration parameters ...

WebLogic Admin Server name [your_adminserver_name]:
AdminServer

WebLogic Admin Server host [your_adminserver_host]: mypc

WebLogic Admin Server port [your_adminserver_port]: 7080

WebLogic User Name for Admin Server [your_user_name]:
user_name
```

```
WebLogic User Password for Admin Server [your_password]:
password

Directory for WebLogic domain which you are configuring
[your_domain_dir]:
C:\bea\user_projects\domains\myworkdomain

Name of WebLogic Server where TransactionVision will be
deployed [your_server_name]: DevServer

2004-08-23 13:26:57,273 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TVision\config\weblogic\tvision-wl-
config.properties" has been successfully updated

2004-08-23 13:27:01,379 [main] - TransactionVision
Info(ScriptCreationSuccessful): Script
"C:\bea\user_projects\domains\myworkdomain\
tvStartWebLogic.cmd" created successfully.

2004-08-23 13:27:01,509 [main] - TransactionVision
Info(ScriptCreationSuccessful): Script
"C:\bea\user_projects\domains\myworkdomain\
tvStartManagedWebLogic.cmd" created successfully.
```

### Configuring WebSphere Application Server Properties

If you are using the WebSphere application server, the following prompts appear:

*Important!* Values for WebSphere Application Server properties are case-sensitive. Be careful to type input in the correct case.

```
The following configuration parameters are required for
setting up the TransactionVision web application in the
WebSphere Application Server. Note: the parameter values
are case sensitive.

Retrieving WebSphere Application Server configuration
parameters ...

WebSphere application server cell name [local_host]: devpc

WebSphere application server node [local_host]: devpc

WebSphere application server name [server1]: server1
```

```
WebSphere virtual host name [default_host]:

Generating script file for WebSphere Application Server
configuration ...

2004-08-23 18:14:53,838 [main] - TransactionVision
Info(FileUpdated): File "C:/Program
Files/Bristol/TVision/bin/WSSetupEnv.bat" has been
successfully updated
```

Completion Information

```
2004-08-23 18:14:53,854 [main] - TransactionVision
Info(TVisionSetupInfoSuccess)
: TVisionSetupInfo has completed successfully.

All program output and user input are logged to
"C:/Program Files/Bristol/TVision/logs/setup.log".

Please run TVisionSetup to complete TransactionVision
setup.

Note:

- TVisionSetup needs to be run in a user environment which
has the correct database environment set.

- For IBM DB2 this includes checking the DB2INSTANCE
environment variable setting and that the user has
database access rights.

- An IBM DB2 database, required to store TransactionVision
data, needs to be created before running TVisionSetup.

- Refer to your IBM DB2 manual for further information on
database setup.
```

## Step 2: Setup Software for TransactionVision

Run **`TVisionSetup`** once after installation. It uses information provided in the **`TVisionSetupInfo`** utility to initialize the DBMS, the TransactionVision Analyzer, and the application server for TransactionVision. By default ,the Analyzer listens for RMI requests on the default port number, but you can specify a different port number.

*Important!* **`TVisionSetup`** needs to be run in a user environment which has the correct database environment set. For IBM DB2 this includes checking the DB2INSTANCE environment variable setting and that the user has database access rights. Refer to your IBM DB2 manual for further information.

### Command Syntax

Use the following commands to run **`TVisionSetup`**:

| Operating System | Script Command |
|---|---|
| AIX, Linux, Solaris | `<TVISION_HOME>/bin/TVisionSetup.sh` |
| Windows | `<TVISION_HOME>\bin\TVisionSetup.bat` |

The syntax for **`TVisionSetup`** is as follows:

```
<TVISION_HOME>/bin/TVisionSetup [-rmiregp PORTNUMBER]
[-debug][-h]
```

- The default port number where the Analyzer listens for RMI calls is 1099; use the `-rmiregp` option to specify a different port number.
- Use the `-debug` option to start rmid and rmiregistry in debug mode.
- Use the `-h` option to print the usage message for this utility.

*Important!* To use **`TVisionSetup`** to configure the TransactionVision web user interface on a WebLogic managed server, you must login as root or have Administration rights. Otherwise, some of the WebLogic configuration files (such as embedded LDAP files) cannot be accessed. To use the embedded WebLogic LDAP server, see the "Managing Users" chapter of the *TransactionVision Administrator's Guide* before running **`TVisionSetup`**.

### Menus

This command provides different menus, depending on whether the host is a WebSphere or WebLogic AppServer with the TransactionVision web user interface installed, and whether LDAP demo mode is used.

Enter the number corresponding to the desired menu action, or enter **99** to perform all actions. The results of each menu action are displayed as they are performed; press the Enter key to continue with the next menu item. Some menu actions require user input.

The following menu appears if WebSphere Application Server and the TransactionVision web user interface are installed, and the DEMO mode is **not** used:

```
1. Initialize database for TransactionVision
2. Verify database initialization for TransactionVision

3. Verify WebSphere MQ setup for TransactionVision

4. Initialize TransactionVision Analyzer
5. Verify TransactionVision Analyzer initialization

6. Initialize LDAP entries for TransactionVision
7. Verify LDAP initialization for TransactionVision

8. Install TransactionVision web application
9. Verify TransactionVision web application installation

99. Select all
V. Display version info for software

Type "quit" or "q" to exit
```

If demo mode is used, or the application server or TransactionVision web user interface are not installed, certain items are omitted from the menu. If an item is omitted, the remaining items are renumbered in sequence.

- If you specified in **`TVisionSetupInfo`** to use DEMO mode for LDAP, items 6 and 7 are omitted from the menu.
- If the WebSphere or WebLogic application server or TransactionVision web user interface are not installed, items 6, 7, 8, and 9 are omitted from the menu.

## User Inputs

The following sections describe the required input for each action.

### 1. Initialize database for TransactionVision

No input required.

### 2. Verify Database initialization for TransactionVision

No input required.

### 3. Verify WebSphere MQ setup for TransactionVision

This menu item displays the following output and prompts:

```
Verifying WebSphere MQ Java setup for TransactionVision
...
```

```
Please specify WebSphere MQ queue manager name:

Do you want to test WebSphere MQ server connection? (Y/N)
[N]:

TransactionVision Info(MQInitialized): WebSphere MQ is
loaded properly for WebSphere MQ server connection

Do you want to test WebSphere MQ client connection? (Y/N)
[N]:
```

*Important!* If you answer **Y**, the following you are prompted to enter the following additional information:

```
Host name for WebSphere MQ queue manager:

Channel name for WebSphere MQ queue manager:

Port number for WebSphere MQ queue manager:

Hit <Enter> to continue ...:
```

#### 4. Initialize TransactionVision Analyzer

This menu item displays the following output and prompts:

```
Initializing TransactionVision Analyzer ...

Do you want to start TransactionVision Analyzer? (Y/N)
[Y]:

Hit <Enter> to continue ...:
```

#### 5. Verify TransactionVision Analyzer initialization

No input required.

#### 6. Initialize LDAP entries for TransactionVision

This menu item displays the following output and prompts:

```
Initializing LDAP for TransactionVision ...

User name is used as the distinguished name for
authentication. Specified user should have authority to
create new LDAP entry.
```

```
Logon user name:
User password:
Directory location to create default TransactionVision
group :
```

Specify the location where the TransactionVision related security definitions will be stored. It can be any valid path in your directory structure. If you are running this LDAP server only for TransactionVision, the typical directory location could be the root directory (the name of your suffix, for example: o=bti,c=us). If you are integrating with an existing LDAP server, consult with your LDAP server administrator to determine the location where these objects should be stored.

```
If you have not yet defined the LDAP suffix you wish to
place TransactionVision objects under (for example,
"o=bti,c=us"), you must first define this suffix using the
LDAP administration tools before continuing. Once the
suffix has been defined, you can either manually create
the suffix node or you can have it created for you by
answering yes to the below question.

Do you want to create default root suffix? (Y/N) [N]: y
Root suffix name:

Do you want to create default Admin user? (Y/N) [Y]:
```

Answer **Y** to create an Admin user object that has privileges to access TransactionVision. This user object is created using the default user object template. If you have existing or non-default users in your LDAP environment that you want the TransactionVision Admin user to be part of, create this user manually using those templates. For more information, see the "Managing Users" chapter of the *TransactionVision Administrator's Guide*.

```
Continue to initialize the LDAP entries for
TransactionVision? (Y/N) [Y]:

Hit <Enter> to continue ...:
```

### 7. Verify LDAP initialization for TransactionVision

This menu item displays the following output and prompts:

```
Verifying LDAP initialization for TransactionVision ...
```

```
User name is used as the distinguished name for
authentication. Specified user should have authority to
read LDAP entries.

Logon user name (ex. cn=root):

User password:

Enter directory location where the default
TransactionVision objects were created:

Continue to verify the LDAP entries for TransactionVision?
(Y/N) [Y]:

If you have created default Admin user, do you want to test
its login (Y/N) [N]:

Hit <Enter> to continue ...:
```

### 8. Install TransactionVision web application

If you use the WebSphere application server, this menu item displays the following output and prompt:

```
Installing TransactionVision web application for
WebSphere. This may take a few minutes...

The WebSphere Application Server where TransactionVision
web application has just been installed must be restarted
in order for TransactionVision to work correctly.
Do you want to restart this application server? (Y/N) [Y]:

Hit <Enter> to continue ...:
```

If the **`TVisionSetup`** script detects whether the TransactionVision web application has already been installed for WebSphere. If so, it prompts you to uninstall the current application before installing the new one.

If you use the WebLogic application server, this menu item displays the following output and prompt:

```
Installing TransactionVision web application for WebLogic.
This may take a few minutes...
```

```
In order for TransactionVision web application to work
correctly, the WebLogic server where TransactionVision
will be deployed must be shutdown in order for setup to
proceed. The server will be restarted automatically after
its startup parameters have been set.

Do you want to shutdown this server now? (Y/N) [Y]

Hit <Enter> to continue ...:
```

9. Verify TransactionVision web application installation

No input required.

## Additional WebSphere Procedures for WebSphere Application Server 5.1 Express Edition

If you are using WebSphere Application Server 5.1 Express Edition, perform the following steps from the WebSphere Admin console.

- Add com.bristol.tvision.home as JVM System property setting.
- Add the DB2 or Oracle JDBC driver to a server configuration
- Install and setup the TransactionVision web user interface

For detailed instructions, see Appendix E, “Configuring Application Servers for TransactionVision.”

## Additional Analyzer Configuration

The following parameters should be checked for the Analyzer:

- Make sure the DBMS schema has been tuned up and tested. For more information, see Chapter 4, “Configuring Databases.”
- Configuration Message Expiry should be set to a value small enough so that the amount of events generated by an "orphaned" configuration message can fit into the event queues without causing major production issues. For more information, see the *TransactionVision Administrator’s Guide*.
- Analyzer thread count should be set to match the test results from the DBMS insert test. For more information, see the *TransactionVision Administrator’s Guide*.

### Windows Service Properties

On Windows, the Analyzer is a Windows service named **TransactionVision Analyzer Service**, which can be managed through the Services administrative tool.

### Service Properties

By default, the Analyzer service is configured to use the local system account and to startup automatically when the system starts. To change the account name or other service properties, use the Windows Services administrative tool, accessible through the Control Panel.

*Important!* Note that the Analyzer must have access to WebSphere MQ and a database. For it to function correctly, the assigned Log On account must have permission to access these resources.

### Error Logging

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

## Multithreaded Servlet/JMS Events

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

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

1. Open the `<TVISION_HOME>/config/services/Analyzer.properties` file.
2. Add the name of the program that spins off threads to the `separate_child_thread_txns` property, as in the following example:

   ```
   separate_child_thread_txns=program1, program2
   ```

   Separate multiple program names with a comma.
3. Close and save `Analyzer.properties`.
4. Restart the Analyzer.

## Local Transaction Matching

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

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

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

- Program name amqcrsta, queue manager QM1, and object TEST.Q1
- Program name amqcrsta, queue manager ALT_QM, and object TEST.Q

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

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

## Reducing Event Database Size

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

After running **`TVisionSetupInfo`**, open the file `<TVISION_HOME>/config/services/Beans.xml`, change the following segment:

**Original Segment:**

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

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

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


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

**Modified Segment:**

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

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

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


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

Make sure only one bean is enabled at one time; otherwise, when both DBWriteEventDefaultBean and DBWriteEventCompressedBean are enabled, an exception will be thrown. When using the compression bean, your `DatabaseDef.xml` must be updated to store the event data in BLOB format. The EVENT and EVENT_OVERFLOW table definitions must be updated, and the event_data column type changed from CLOB to BLOB. After this change has been made you will need to restart your Analyzer and Web application, and create a new project for the changes to take effect. The following example shows `DatabaseDef.xml` before the change to store event data in BLOB format:

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

# Chapter 6
# Configuring WebSphere MQ Sensors

TransactionVision provides two types of WebSphere MQ Sensors:

- The WebSphere MQ Sensor library
- The WebSphere MQ API Exit

## Configuring the WebSphere MQ Sensor Library

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

### Distributed Platforms

Distributed platforms include all platforms other than OS/390 and OS/400. On distributed platforms, you must add the directory location of the Sensor library to an environment variable setting on the computer where the monitored application runs before starting the application. The following table shows the appropriate environment variable and directory location to set for each platform. If a path including the standard WebSphere MQ library exists as part of the environment value, the Sensor entry must appear before it in order to use TransactionVision.

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

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

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.

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

On UNIX platforms:
*`installation_directory`*`/wmqsensor` *`application_command_line`*

On Windows platforms:
*`installation_directory`*`\wmqsensor` *`application_command_line`*

### OS/390 Batch, IMS, and WebSphere MQ-IMS Bridge

On the OS/390 batch and IMS platforms, the SLMLKSTB member of the sample procedure library thlqual.SSLMPROC contains a sample job to bind the Sensor to an WebSphere MQ application program. The WebSphere MQ batch stub section—CSQBSTUB, CSQBRRSI, or CSQBRSTB for Batch or CSQQSTUB for IMS—is replaced in the application load module with the corresponding Sensor batch or IMS stub—SLMBSTUB, SLMBRRSI, SLMBRSTB, or SLMQSTUB. After this bind, the application will invoke the Sensor instead of WebSphere MQ directly.

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

*Important!* Note that if your current applications use the OS/390 Resource Recovery Services (RRS) in batch, the calls to SRRCMIT and SRRBACK are not recorded by the Sensor but are simply passed through to WebSphere MQ.

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

### OS/390 CICS

On the OS/390 CICS platform, Sensors use the API-crossing exit mechanism provided by the CICS adapter of WebSphere MQ for OS/390. See Chapter 3, “Installation,” for more information about the crossing-exit mechanism.

### OS/400

On the OS/400 platform, the main Sensor service program has the same name as the WebSphere MQ service program: LIBMQM for non-threaded programs and LIBMQM_R for threaded programs. The two TransactionVision service programs have the same signature and exported symbols (in the same order) as their WebSphere MQ counterparts.

TransactionVision also provides two utility service programs:

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

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

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

#### User program is created by CRTPGM with the parameter “BNDSRVPGM(QMQM/LIBMQM)” or “BNDSRVPGM(QMQM/LIBMQM_R” or “BNDSRVPGM(QMQM/AMQZSTUB)”.

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

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

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

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

*Important!* 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

## Setting the Configuration Queue Name

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

### UNIX, Windows, and OS/400

On UNIX, Windows, and OS/400 platforms, set the TVISION_CONFIGURATION_QUEUE environment variable to the Sensor configuration queue specified in the communication link for all processes that use the Sensor.

### IBM OS/390 Batch, IMS and WebSphere MQ-IMS Bridge

On IBM OS/390 Batch, IMS, and WebSphere-IMS bridge, a user-installable load module named SLMBCNFG can be generated to control which queue the Sensor reads to retrieve configuration messages from the Analyzer.

1. Edit thlqual.SSLMPROC(SLMBCFGQ) and change as follows:

(a) Change the value of the SET CONFIGQ statement to specify the desired configuration queue name.

(b) Change the SET THLQUAL statement to specify the high-level qualifier for your TransactionVision Sensor libraries.

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

2. Submit thlqual.SSLMPROC(SLMBCFGQ) to effect the change.

For more information about the WebSphere MQ-IMS bridge Sensor, see "Using the WebSphere MQ-IMS Bridge Sensor" on page 114.

### IBM OS/390 CICS

On IBM OS/390 CICS, a user-installable program named SLMCNFQ can be written to control which queue the Sensors read from to retrieve configuration messages from the Analyzer.

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

*Important!* Note that the installation procedure for the OS/390 CICS Sensor does **NOT** create an SLMCNFQ program. For instructions on writing this program, see "Writing SLMCNFQ."

If the attempt to execute the SLMCNFQ fails, the OS/390 CICS WebSphere MQ Sensor tries to load the program SLMBCNFG and gets the configuration queue name. For instructions on generating this program, see "Generating SLMBCNFG.."

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

*Important!* To use a different configuration queue name for each CICS region, see "Using Separate Configuration Qeueus for Each CICS Region".

### Writing SLMCNFQ

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

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

## Generating SLMBCNFG

SLMBCNFG is generated the same way as for OS/390 batch and IMS; that is:

1. Edit thlqual.SSLMPROC(SLMBCFGQ) and change as follows:

   (a) Change the value of the SET CONFIGQ statement to specify the desired configuration queue name.

   (b) Change the SET THLQUAL statement to specify the high-level qualifier for your TransactionVision Sensor libraries.

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

2. Submit thlqual.SSLMPROC(SLMBCFGQ) to effect the change.

### Using Separate Configuration Queues for Each CICS Region

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

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

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

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

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

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

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

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

   - Add an additional step before the actual CICS execution step:

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

```
//L.SYSLMOD DD
DSN=&&CONFIGQ(SLMBCNFG),DISP=(,PASS),SPACE=(TRK,(1,,1))
//*-------------------------------------------------------
```

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

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

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

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

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

## Setting the Configuration Queue Check Interval

By default, Sensors check the configuration queue for new configuration messages every five seconds. On UNIX, Windows, and OS/400 platforms, however, you may specify a different configuration queue check interval. To specify a non-default configuration queue check interval for a Sensor, set the TVISION_CONFIG_CHECK_INTERVAL environment variable to the desired interval, in milliseconds.

## Configuring the WebSphere MQ API Exit Sensor

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

- `tvisionapiexit`
- `tvisionapiexit_r`

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

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

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

- There is no need to disable FASTPATH_BINDING (see "WebSphere MQ Sensors and FASTPATH_BINDING" later in this chapter).
- The completion and reason codes for MQDISC calls are hard-coded to MQCC_OK and MQRC_NONE, respectively. The event time for MQDISC events is set to the before-MQDIC function invocation time.
- The API Exit Sensor collects some TransactionVision internal events generated from WMQ calls made by the Analyzer, and also internal WMQ events from Java Sensors using a client connection to the listener.

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

## Configuring the WebSphere MQ API Exit Sensor on Distributed and OS/400 Platforms

To use the WebSphere MQ API Exit Sensor on distributed platforms or OS/400, you must define it in new stanzas in the `mqs.ini` file, which contains definitions applied to the whole WebSphere MQ environment, and `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.

### New Stanzas

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

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

  The data field can have a maximum of 24 characters; therefore, the TVQ object name value may be up to 20 characters and may include the * wildcard character at the beginning and/or end of the string. Only one queue string may be specified for the TVQ attribute. For more information, see "Discarding WMQ Events on TransactionVision Queues" on page 103.

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

The following is an example illustrating the Sensor configuration per queue manager (`qm.ini`). In this example, events from queue names containing the string "MYTVQUEUE" are discarded.

```
ApiExitLocal:
     Name=TransactionVisionWMQSensor
     Sequence=100
     Function=TVisionEntryPoint
     Module=/opt/TVision/lib/tvisionapiexit
     Data=TVQ=*MYTVQUEUE*
```

## Configuring the WebSphere MQ API Exit Sensor on Windows Platforms

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

A new property page for the WebSphere MQ Services node, API Exits, describes the two types of API exit managed from this node: ApiExitCommon and ApiExitTemplate. In the Exits property page for individual queue managers, you can update the ApiExitLocal. The Configure... buttons launch a dialog to manage the entries within each stanza. The dialog consists of a multicolumn 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 WebSphere MQ API Exit Sensor on Distributed Platforms" for a description of required stanzas and attribute values.

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

### Identifying Programs To Monitor

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

- `exit_sensor.allow` defines which programs will be monitored. All other programs are **NOT** monitored. Note that if this file is empty, **no** programs will be monitored.
- `exit_sensor.deny` defines which programs will not be monitored. All other programs will be monitored.

These files are located at the top level TransactionVision installation directory. For example, on Solaris if TransactionVision is installed at `/opt/TVision`, these two files exist in the `/opt/TVision` directory.

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

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

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

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

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

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

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

### Discarding WMQ Events on TransactionVision Queues

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

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

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

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

## WebSphere MQ Sensors and FASTPATH_BINDING

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

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

## Using Sensors with WebSphere MQ Samples

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

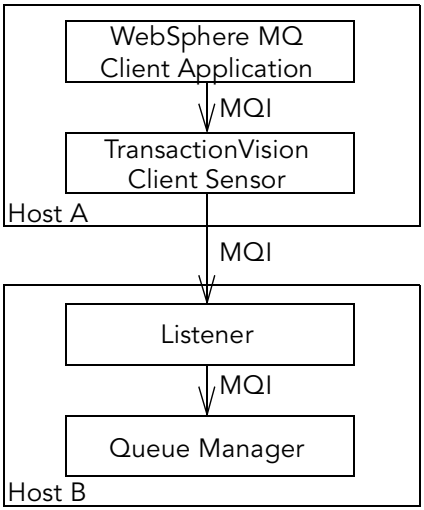
- There are two locations for WebSphere MQ samples on the Windows platform. If you run the samples under `WebSphere MQ\bin`, you must copy the sample executables (amqsput, amqsget, etc.) to a different directory to enable them to load the Sensor library instead of the standard WebSphere MQ library. This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

## WebSphere MQ Client Application Monitoring

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

### Distributed Monitoring

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




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

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

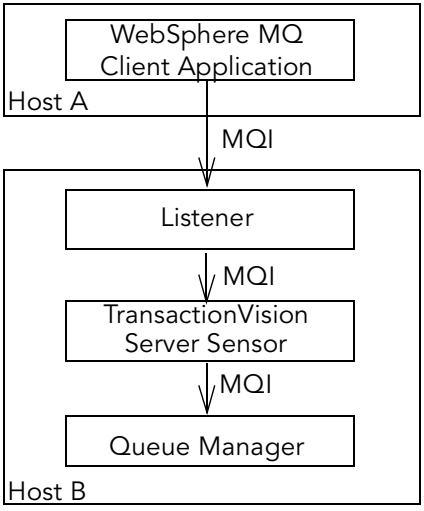
The trace information generated will be based on the client application context. This means information such as program name, program instance, and host name, will be related to the client application directly.

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

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

### Centralized Monitoring

The following diagram illustrates how the Sensor works in a centralized monitoring environment:



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

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

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

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

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

- As discussed before, context information reported is associated with the listener program. However, client application origin context information can be retrieved indirectly through the message header (MQMD) structure embedded in the MQPUT/MQPUT1/MQGET feedback data through the call parameters.
- If you use the Servlet, JMS, or EJB Sensors with a client connection to a queue manager through a listener, which is being monitored with the TransactionVision WebSphere MQ Sensor, internal TransactionVision events will be captured. It is recommended to either use a separate unmonitored listener for the Servlet, JMS, or EJB Sensors or use server binding connections from these Sensors. If this cannot be done, change the data collection filter to exclude the configuration and event queues.

The table below summarizes the characteristics of the two approaches:

| Distributed Monitoring | Centralized Monitoring |
|---|---|
| Direct client MQI tracing. | Indirect tracing through listener MQI calls. |
| Direct client context information access. | Client context information through call parameters. |
| Client Sensor on each client host. | Server Sensor per queue manager host. |
| Monitors applications using WebSphere MQ C binding. | No runtime binding language restriction. |
| Supports client applications running on Windows, Solaris, HP-UX, AIX, and Linux. | Support clients connecting to queue managers running on Windows, Solaris, HP-UX, and AIX. |

### Installation and Configuration Considerations

For distributed monitoring, install client Sensors on each host where WebSphere MQ client applications run. Follow the standard Sensor installation instructions in Chapter 3, “Installation.”

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

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

### Centralized Monitoring with WebSphere MQ 5.2

For WebSphere MQ 5.2, there are two versions of listener programs: **`amqcrsta`** and **`runmqlsr`**.

- **`runmqlsr`** is available on all WebSphere MQ 5.1 and 5.2 distributed platforms.
- **`amqcrsta`** is available on Solaris, HP-UX, and AIX for WebSphere MQ 5.0, 5.1, and 5.2 installations.

**`runmqlsr`** is multi-threaded and takes advantage of the threading environment to serve different client connections. Specifically, the program uses a single thread for each client connection request. This program is started explicitly at startup or another point of time, and proactively monitors any incoming traffic on the port associated with the corresponding queue manager server connection channel.

For TransactionVision monitoring, this program should be started in an environment where the shared library search path has been set up to point to the Sensor directory.

**`amqcrsta`** is available on the various UNIX platforms. It is configured as a service to be invoked by the **`inetd`** daemon upon an incoming request at the queue manager server connection channel associated port. A separate instance of the **`amqcrsta`** program is started for each client connection. In other words, the life cycle of each program instance is related to the corresponding client connection lifetime.

For TransactionVision monitoring, appropriate changes should been made to the inetd configuration file (`/etc/inetd.conf`) so that the listener program will be started under the appropriate search library path. This can be achieved by using the **`Sensor`** script shipped with TransactionVision.

Centralized monitoring configuration considerations vary between individual platforms.

### Windows

In the Windows environment, the **runmqlsr** program resides in the same directory in the file system as the regular WebSphere MQ library (`mqm.dll`). Under the Windows DLL search mechanism, the directory where the running program is installed takes the highest precedence. Thus the regular WebSphere MQ library will always be picked if the **runmqlsr** program is run from the standard location.

To resolve this problem, copy the **runmqlsr** program to another location in the file system, and make sure the Sensor directory takes precedence in the path setup.

Under WebSphere MQ 5.1, the listener program may have been configured to start up automatically. WebSphere MQ 5.1 provides a graphical user interface “MQServices” through the Microsoft Management Console for controlling the listener activities. Since this tool assumes the regular location of the listener program, make sure you stop the listener program through the user interface, and start the program in the way described above. For example, the program can be started from the command line as follows:

```
C:\> myinstall\runmqlsr.exe –m my.qmgr –t tcp –p 1416
```

This starts a listener process for queue manager `my.qmgr`, using TCP transport protocol, and listens at port address 1416.

### Solaris

If your system is configured to use the **amqcrsta** program, modify the appropriate entry in the inetd configuration file (`/etc/inetd.conf`) so that the **wmqsensor** script is used to set up the shared library path and invoke the listener program.

The original entry may look like the following:

```
mq1 stream tcp nowait mqm /opt/mqm/bin/amqcrsta amqcrsta
–m my.qmgr
```

Change this entry as follows:

```
mq1 stream tcp nowait mqm /opt/TVision/wmqsensor wmqsensor
/opt/mqm/bin/amqcrsta –m my.qmgr
```

Essentially, you should change the program to be invoked to the **wmqsensor** script and provide the full path name for the script. You should also modify the last part of the entry to provide the exact command line for invoking the script.

After modifying the configuration file, you can instruct the **inetd** running instance to refresh from the configuration file by the following command:

```
% kill –HUP <inetd process ID>
```

If your system is configured to use the **runmqlsr** program, run the listener program as the mqm user in an environment where the shared library path is set up with the Sensor directory and takes higher precedence than the regular WebSphere MQ runtime directory. For example:

```
% setenv LD_LIBRARY_PATH
/opt/TVision/lib:/opt/mqm/lib:/usr/lib
```

If you want to run the listener program as a user other than mqm, copy **runmqlsr** to another directory and run the listener program there. For example:

```
cp /opt/mqm/bin/runmqlsr /tmp/runmqlsr
/tmp/runmqlsr -m my.qmgr -t tcp -p 1416
```

### HP-UX

On HP-UX, both the **amqcrsta** and **runmqlsr** programs are set up to make use of embedded path (as opposed to dynamic path setting through SHLIB_PATH environment variable) for resolving the location of the shared library. In other words, these programs will always resolve to the regular WebSphere MQ library even if the environment is set up to search the Sensor directory first.

This problem can be resolved by changing the program shared library path search behavior through the **chatr** command. Follow the steps described below:

1. Make a copy of the program in another directory. For example, you can create a new sub-directory under the TVision directory and copy the listener program.

   ```
   % mkdir /opt/TVision/mqprograms
   % cp /opt/mqm/bin/amqcrsta /opt/TVision/mqprograms
   ```

2. Use the **chatr** program to modify the shared library searching attributes:

   - use the +b disable flag to disable embedded path usage

- use the +s enable for to enable shared library search through SHLIB_PATH

```
% chatr +b disable +s enable amqcrsta
```

If your system is configured to use the **amqcrsta** program, modify the appropriate entry in the inetd configuration file (/etc/inetd.conf) so that the **wmqsensor** script is used to set up the shared library path and invoke the listener program. **Make sure you run the copy of the listener program you have modified as described above.**

The original entry may look like the following:

```
mq1 stream tcp nowait mqm /opt/mqm/bin/amqcrsta amqcrsta
–m my.qmgr
```

This entry should be changed as follows:

```
mq1 stream tcp nowait mqm /opt/TVision/wmqsensor wmqsensor
/opt/TVision/mqprograms/amqcrsta –m my.qmgr
```

Essentially, you should change the program to be invoked to the **wmqsensor** script and provide the full path name for the script. You should also modify the last part of the entry to provide the exact command line for invoking the script.

After modifying the configuration file, you can instruct the **inetd** running instance to refresh from the configuration file by the following command:

```
% kill –HUP <inetd process ID>
```

If your system is configured to use the **runmqlsr** program, run the listener program in an environment where the shared library path is set up with the Sensor directory takes higher precedence than the regular WebSphere MQ runtime directory. For example:

```
% setenv SHLIB_PATH /opt/TVision/lib:/opt/mqm/lib:/usr/lib
```

### AIX

If your system is configured to use the **amqcrsta** program, modify the appropriate entry in the inetd configuration file (/etc/inetd.conf) so that the **wmqsensor** script is used to set up the shared library path and invoke the listener program.

The original entry may look like the following:

```
mq1 stream tcp nowait mqm /usr/lpp/mqm/bin/amqcrsta
amqcrsta –m my.qmgr
```

This entry should be changed as follows:

```
mq1 stream tcp nowait mqm /usr/lpp/TVision/wmqsensor
wmqsensor /usr/lpp/mqm/bin/amqcrsta –m my.qmgr
```

Essentially, you should change the program to be invoked to the **wmqsensor** script and provide the full path name for the script. You should also modify the last part of the entry to provide the exact command line for invoking the script.

After modifying the configuration file, you can instruct the **inetd** running instance to refresh from the configuration file by the following command:

```
% refresh –s inetd
```

If your system is configured to use the **runmqlsr** program, run the listener program in an environment where the shared library path is set up with the Sensor directory takes higher precedence than the regular WebSphere MQ runtime directory. For example:

```
% setenv LIBPATH
/usr/lpp/TVision/lib:/usr/lpp/mqm/lib:/usr/lib
```

*Important!* On the AIX platform with WebSphere MQ 5.3, monitoring **runmqlsr** is not supported. However, you can still monitor **amqcrsta**.

### Centralized Monitoring with WebSphere MQ 5.3

For WebSphere MQ 5.3, threaded channels started by the listener do not run under that process, but under a process called AMQRMPPA, known as a pool process.

#### Windows Configuration

To configure centralized monitoring on Windows, perform the following steps:

1. Using the task manager, kill all processes called `amqrmppa`.
2. Copy the `amqrmppa` application to a location other than the WebSphere MQ\bin directory.
3. Set the system PATH variable to pick up the Sensor library location prior to the WebSphere MQ\bin directory.
4. Start an instance of the `amqrmppa` application from the new location for each queue manager as follows:

```
amqrmppa.exe -m QM_name
```

### UNIX Configuration

To configure centralized monitoring on UNIX platforms, perform the following steps:

1. Kill all processes called `amqrmppa`.
2. Set the system dynamic link path environment variable (for example, LD_LIBRARY_PATH on Solaris) to pick up the Sensor library location prior tot the WebSphere MQ directory location. On AIX, you must copy the `amqrmppa` executable to another location.
3. Start an instance of the `amqrmppa` application for each queue manager as follows:

   ```
   amqrmppa -m QM_name
   ```

### Disabling the Pool Process

When trying to isolate one or more channels from the rest of the configuration (when testing channel exits, for example), it can be useful to disable the pool process. This causes the WebSphere MQ listener to revert to 5.2 behavior, so that channels run under the originating process. To disable the pool process, set the environment variable MQNOREMPOOL to any value. If this environment variable exists, set to any value, channel threads are run as part of the listener or channel initiator process.

## Using the WebSphere MQ-IMS Bridge Sensor

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

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

### Sensor Setup

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

1. Customize the sample TVISIONB startup procedure in thlqual.SSLMPROC and copy it to an appropriate PROCLIB. TVISIONB requires four startup parameters, which may be specified in the procedure or on the START command.

   The QMGR parameter specifies the name of the WebSphere MQ queue manager to which TVISIONB must connect to access its configuration and event queues. Note that this queue manager is the one to which the Analyzer connects when establishing a communication link to the Sensor and not necessarily the queue manager(s) to which the WebSphere MQ-IMS bridge is connected. It must be the same queue manager used when defining the configuration and event queues during installation (see the sample job in thlqual.SSLMINST(SLMCRTQS).

   The MAXQ parameter specifies the maximum amount of storage, in megabytes, that TVISIONB will allocate for its buffer queue. Please refer to “The TVISIONB Buffer Queue” on page 116.

   The EDPROC parameter specifies the name of the procedure to start the TVISIOND address space.

   The IMSJOB parameter specifies the jobname of the IMS control region for the IMS system to be monitored.

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

### WebSphere MQ-IMS Bridge Sensor Operation

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

1. Assure that IMS control region is started with the TransactionVision DFSYIOE0 exit routine accessible in its STEPLIB.

2. Start the TVISIONB address space from the system operator's console, specifying any parameters to be overridden in the startup procedure. For example:

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

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

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

TVISIONB will automatically start TVISIOND.

3. Request bridge monitoring from the TransactionVision web application on a connected workstation. Please refer to the *TransactionVision User's Guide* for more information.

4. Ordinarily, the activity of the bridge Sensor is controlled from the TransactionVision web application. However, you may disable the Sensor from the system console with the MODIFY command : F TVISIONB,DISABLE MQIMSBDG. When disabled the TransactionVision exit routine, DFSYIOE0, continues to run in the IMS control region but sends no events to the TVISIONB server component. Re-enable the Sensor as follows: F TVISIONB,ENABLE MQIMSBDG.

5. Stop the TVISIONB address space as follows: P TVISIONB. This will implicitly disable the Sensor; the exit routine continues running but does not attempt to send events to the TVISIONB. TVISIOND will automatically by stopped.

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

### The TVISIONB Buffer Queue

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

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

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

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

### Event Data

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

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

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

- Beans.xml

- IMSBridgeObject.xml

### Beans.xml

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

```
<Module type="Context" name="EventModifierCtx">
    <!--
        This context contains beans that modify XML event,
        which are unmarshalled from the raw event stream.
        User can easily plug in their modifier in this
        section. The sample bean checks the user data to
        see if its a valid XML document. If so, the
        document will be parsed and plugged under the user
        data node

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

### IMSBridgeObject.xml

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

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

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

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

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

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

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

### Data Collection Filters and Queries

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

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

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

## Using the WebSphere Business Integration Sensor

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

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

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

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

### Message Brokers Toolkit for WebSphere Studio Integration

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

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

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

### TransactionVision User-Defined Node Installation for WBIMB

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

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

   **Windows:**
   Copy the library *sensor_install_directory*\mqsi\tvisiontrace.lil to the directory *WBIMB_install_directory*\bin.

   **UNIX:**

   Copy the library to the directory *WBIMB_install_directory*/lil.
3. Restart the WBI message broker(s).

### Node Insertion

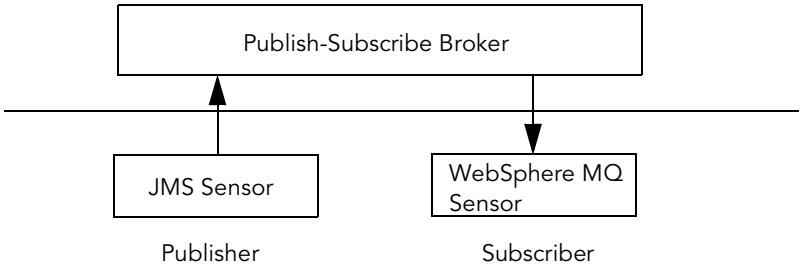
You may now insert any number of TransactionVision Sensor trace and failure nodes into any message flows through the Message Brokers Toolkit. Remember that any changes to the configuration repository must be deployed to the appropriate brokers.

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

## Monitoring WebSphere MQ Publish-Subscribe Topics

Unlike the WBI broker, the publish-subscribe broker does not guarantee that the message ID is the same at both the publisher and subscriber ends. Therefore, the publish-subscribe broker must be monitored for the Analyzer to correlate events.

In the following illustration, only the publishing message and subscribing message are collected, and they have different applications and different message IDs. Therefore, the publisher and subscriber are treated by the Analyzer as two different business transactions, although they actually pass the message under the same publish-subscribe topic.




If the Sensor is monitoring the broker, four events are collected, so that the publisher is correlated to the broker, which is further correlated to the subscriber.

To correlate the publisher and the subscriber **without** monitoring the broker, edit the `<TVISION_HOME>/config/services/Beans.xml` file and un-comment the following beans in `CorrelationMQHelperCtx` and `CorrelationJMSHelperCTX`:

- com.bristol.tvision.services.analysis.eventanalysis.MQPubSubRelationshipBean
- com.bristol.tvision.services.analysis.eventanalysis.JMSPubSubRelationshipBean

These beans detect whether a publisher event and subscriber event are linked under the same topic and whether a message is being passed between the two applications.

*Important!* If you enable these two beans **and** monitor the broker, the Analyzer creates two business transactions: one linking the publisher and subscriber directly, and one linking them through the broker.

# Chapter 7
# Configuring the Proxy Sensor

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

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

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

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

## Application Requirements

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

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

## Enabling the Proxy Sensor

The Proxy Sensor is enabled by the TransactionVision license code.

## Configuring the Proxy Definition File

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

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

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

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

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

### Subelements

Specify the following subelements for each proxy element:

| Element | Required? | Attributes | Description |
|---|---|---|---|
| Request | Yes | queue<br>queueManager | The WebSphere MQ queue and queue manager from which the proxy program gets the event. |
| Reply | Yes | queue<br>queueManager | The WebSphere MQ queue and queue manager where the proxy program puts the reply event of the same message ID and correlation ID as the request event. |
| Program | Yes | name<br>path | The name and path of the proxy program. |
| Host | Yes | name<br>os | The name and operating system of the host where the proxy program runs. |
| Retrieve | No | queue | Causes the Proxy Sensor to check the given queue object against its definition rather than the object defined in <Reply>. This element allows the Sensor to work with looser coupling. |
| OS390Batch | No | jobID<br>jobName<br>stepName<br>tcbAddr | If the proxy program is an OS/390 Batch job, specify the job ID, job name, step name, and TCB address. |
| OS390CICS | No | regionName<br>transactionID<br>taskNum | If the proxy program is an OS/390 CICS task, specify the region name, transaction ID, and task number. |
| OS390IMS | No | psbName<br>transactionName<br>regionID<br>jobName<br>imsID<br>imsType | If the proxy program is an OS/390 IMS job, specify the PSB name, transaction name, region ID, job name, IMS ID and IMS type. |

### Optional Attributes for the Proxy Element

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

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

## Configuring the User Interface

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

Chapter 8

# Configuring the Servlet, EJB, and JMS Sensors

Once the TransactionVision Servlet, EJB, or JMS Sensor is installed, you must instrument certain WebSphere MQ and application server files to enable the Sensors to collect event data for analysis. The **`SensorSetup`** script prompts you for required information, instruments these files, and turns on the Sensor in the application server. You may follow the instructions in Appendix E if you prefer to configure your application server manually.

*Important!* When collecting events from the Servlet, JMS, or EJB Sensors, do **not** enable the WebSphere MQ Library Sensor on the WebSphere MQ listener programs (amqcrsta, runmqlsr, and amqrmppa). Doing so results in WebSphere MQ events from the Servlet, JMS, or EJB Sensor to the TransactionVision configuration and event queues.

After you install the Servlet, EJB, or JMS Sensor, run the **`SensorSetup`** script for configuring the application server environment to use the Sensor. You may also run this script at a later time to change configuration information. To run this script, login as a user with root or Administrator privileges and enter the following command:

| Operating System | Script Command |
|---|---|
| AIX, Linux, Solaris | `<TVISION_HOME>/bin/SensorSetup.sh` |
| Windows | `<TVISION_HOME>\bin\SensorSetup.bat` |

The JVM property `com.bristol.tvision.sensor.properties` specifies the location of the Sensor properties file, which specifies the logging configuration file and Sensor configuration file. If you wish to run multiple Sensor instances and want each instance to use its own log file, see Chapter 10, "Configuring TransactionVision Component Logging."

*Important!* If you are using the WebLogic application server and configuring the TransactionVision Sensor on a managed server, you must run the WebLogic Admin Server and Node Manager for the TransactionVision setup utilities to work correctly.

## SensorSetup User Inputs:

The script displays the following information and prompts:

```
Checking Java version ...

This program collects configuration information about
external tools used by TransactionVision Sensor in order
to setup correct environment first. You will be prompted
to input required configuration parameters. If a default
value is provided in [], press <Enter> will set the
parameter to this default value. Press <Space><Enter> will
set the parameter to an empty value.

Please specify name of directory where you want to store
your log files [C:\Program
Files\Bristol\TransactionVision\logs]:
```

Enter the name of the directory where you want to store log files, or press Return to use the default directory. This information is stored in the `Sensor.Logging.xml` file.

```
Please select your web application server by number:

    1. IBM WebSphere Application Server
    2. BEA WebLogic
    3. None

Your web application server product []: 2
```

Enter the number corresponding to your web application server. The remaining prompts differ, depending on whether you are using WebSphere or WebLogic.

### WebLogic Prompts

```
WebLogic installation location [C:\bea\weblogic81]:
```

Enter the pathname of the installation location of your application server.

```
2003-09-26 13:11:49,097 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TransactionVision\config\setup\DefaultInstal
lPath.xml" has been successfully updated
```

```
2003-09-26 13:11:49,137 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TransactionVision\bin\WebSetupEnv.bat" has
been successfully updated
```

```
The following configuration parameters are required for
setting up the TransactionVision Sensor in the WebLogic.
```

```
Note: The parameter values are case sensitive.
```

```
Retrieving WebLogic configuration parameters ...
```

```
WebLogic Admin Server name [your_adminserver_name]:
```

Enter the name of your WebLogic administration server.

```
WebLogic Admin Server host [your_adminserver_host]: myhpc
```

Enter the host name of your WebLogic administration server.

```
WebLogic Admin Server port [your_adminserver_port]: 7080
```

Enter the port number of your WebLogic administration server.

```
WebLogic User Name for Admin Server [your_user_name]:
user_name
```

Enter your user name for your WebLogic administration server.

```
WebLogic User Password for Admin Server [your_password]:
password
```

Enter the password for the user name specified in the previous question.

```
Directory for WebLogic domain which you are configuring
[your_domain_dir]:
C:\bea\user_projects\domains\myworkdomain
```

Enter the pathname of your WebLogic domain directory.

```
WebLogic Servers to be monitored by TransactionVision
Sensor.

Multiple servers can be listed here and separated by ",".
[your_server_name]: SensorServer, DevServer
```

Enter the names of the all WebLogic servers to be monitored by TransactionVision Sensors. Separate multiple server names with commas.

```
2003-09-26 13:26:57,273 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TransactionVision\config\weblogic\tvision-
wl-sensorconfig.properties" has been successfully updated

2003-09-26 13:27:01,379 [main] - TransactionVision
Info(ScriptCreationSuccessful): Script
"C:\bea\user_projects\domains\myworkdomain\
tvStartWebLogic.cmd" created successfully.

2003-09-26 13:27:01,509 [main] - TransactionVision
Info(ScriptCreationSuccessful): Script
"C:\bea\user_projects\domains\myworkdomain\
tvStartManagedWebLogic.cmd" created successfully.

Retrieving installation path for dependent software tools
...

WebSphere MQ installation location [C:\Program
Files\IBM\WebSphere MQ]:
```

Enter the name of the WebSphere MQ installation directory, or press Return to use the default directory.

```
A copy of WebSphere MQ Java SupportPac (MA88) is found at
C:\Program Files\IBM\WebSphere MQ\Java

Do you want to use the MA88 SupportPac from this directory?
(Y/N) [Y]:
```

Enter Y to use the found support pac. NOTE: This prompt does not appear if your MQ version is 5.3 or later; MQ Java support is built in with the MQ distribution starting with version 5.3.

```
2003-03-18 14:50:11,397 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TransactionVision\config\setup\DefaultInstal
lPath.xml" has been successfully updated

The following configuration parameters are required for
TransactionVision sensor to correctly retrieve it's
configuration messages.
Retrieving sensor configuration parameters ...

Configuration queue manager [config_qm_name]:
devpc.tv1.manager
```

Enter the name of the queue manager the Sensor should use to retrieve configuration messages from TransactionVision Analyzers.

```
Configuration queue [TVISION.CONFIGURATION.QUEUE]:
```

Enter the name of the queue the Sensor should use to retrieve configuration messages from TransactionVision Analyzers.

```
Type of connection to configuration queue manager (S for
server connection, C for client connection): c
```

Enter `S` to use a server connection to the configuration queue manager or `C` to use a client connection.

*Important!* The following three prompts are only displayed if you enter **C** for a client connection:

```
Name of host where configuration queue manager locates
[configuration_qm_host]: mypc
```

Enter the name of the host associated with the configuration queue manager.

```
Port number client uses to connect to configuration queue
manager [1414]:
```

Enter the port number associated with the configuration queue manager.

```
Name of channel client uses to connect to configuration
queue manager [configuration_qm_channel]: MYPC.TV1.CHL
```

Enter the name of the channel associated with the configuration queue manager.

```
2003-03-18 14:51:38,823 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TransactionVision\
config\sensor\SensorConfiguration.xml" has been
successfully updated

You have following TransactionVision Sensors installed on
this host:

Servlet Sensor
JMS Sensor
EJB Sensor

Instrumenting file C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar for TransactionVision JMS
Sensor ...

Instrumenting file C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar for TransactionVision JMS
Sensor has completed successfully. Result file is saved in
directory C:\Program
Files\Bristol\TransactionVision\java\lib\

In order to invoke JMS Sensor, you need to put the
instrumented C:\Program
Files\Bristol\TransactionVision\java\lib\com.ibm.mqjms.jar
in front of the original com.ibm.mqjms.jar shipped with
WebSphere MQ in your java CLASSPATH.
```

*Important!* See "Setting the CLASSPATH Environment Variable" for instructions.

```
Do you wish to monitor JMS methods along with Servlet
methods? (Y/N) [Y]:

Instrumenting file
C:\bea\weblogic81\server\lib\weblogic.jar for
TransactionVision Servlet Sensor ...

Instrumenting file
C:\bea\weblogic81\server\lib\weblogic.jar for
TransactionVision Servlet Sensor has completed
successfully. Result file is saved in directory C:\Program
Files\Bristol\TransactionVision\java\lib
```

```
Instrumenting file
C:\bea\weblogic81\server\lib\webservices.jar for
TransactionVision Servlet Sensor ...

Instrumenting file
C:\bea\weblogic81\server\lib\webservices.jar for
TransactionVision Servlet Sensor has completed
successfully. Result file is saved in directory C:\Program
Files\Bristol\TransactionVision\java\lib

Adding TransactionVision Servlet and JMS Sensor to
WebLogic.

...

2003-03-18 14:54:41,846 [main] - TransactionVision
Info(SensorSetupSuccess): SensorSetup has completed
successfully. All program output and user input are logged
in file "C:\Program
Files\Bristol\TransactionVision\logs\setup.log".
```

### WebSphere Prompts

```
Retrieving installation path for dependent software tools
...

WebSphere MQ installation location [C:\Program
Files\IBM\WebSphere MQ]:
```

Enter the name of the WebSphere MQ installation directory, or press Return to use the default directory.

```
A copy of WebSphere MQ Java SupportPac (MA88) is found at
C:\Program Files\IBM\WebSphere MQ\Java

Do you want to use the MA88 SupportPac from this directory?
(Y/N) [Y]:
```

Enter Y to use the found support pac. NOTE: This prompt does not appear if your MQ version is 5.3 or later; MQ Java support is built in with the MQ distribution starting with version 5.3.

```
WebSphere Application Server installation location
[C:\Program Files\Websphere\AppServer]:
```

Enter the name of the WebSphere Application Server installation directory, or press Return to use the default directory.

```
2003-03-18 14:50:11,397 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TransactionVision\config\setup\
DefaultInstallPath.xml" has been successfully updated

The following configuration parameters are required for
TransactionVision sensor to correctly retrieve it's
configuration messages.
Retrieving sensor configuration parameters ...

Configuration queue manager [config_qm_name]:
devpc.tv1.manager
```

Enter the name of the queue manager the Sensor should use to retrieve configuration messages from TransactionVision Analyzers.

```
Configuration queue [TVISION.CONFIGURATION.QUEUE]:
```

Enter the name of the queue the Sensor should use to retrieve configuration messages from TransactionVision Analyzers.

```
Type of connection to configuration queue manager (S for
server connection, C for client connection): c
```

Enter `S` to use a server connection to the configuration queue manager or `C` to use a client connection.

*Important!* The following three prompts are only displayed if you enter **C** for a client connection:

```
Name of host where configuration queue manager locates
[configuration_qm_host]: mypc
```

Enter the name of the host associated with the configuration queue manager.

```
Port number client uses to connect to configuration queue
manager [1414]:
```

Enter the port number associated with the configuration queue manager.

```
Name of channel client uses to connect to configuration
queue manager [configuration_qm_channel]: MYPC.TV1.CHL
```

Enter the name of the channel associated with the configuration queue manager.

```
2003-03-18 14:51:38,823 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TransactionVision\
config\sensor\SensorConfiguration.xml" has been
successfully updated

The following configuration parameters are required for
setting up the TransactionVision Sensor in the WebSphere
Application Server.
Note: The parameter values are case sensitive.
Retrieving WebSphere Application Server configuration
parameters ...

Servers to be monitored by TransactionVision Sensor: [cell
name/node name/server name]:
```

This prompt only appears for WebSphere Application Server 5. List servers in the format "cell/node/server". Separate multiple servers with a comma.

```
Servers to be monitored by TransactionVision Sensor [node
name/server name]:
```

This prompt only appears for WebSphere Application Server 4. List servers in the format "node name/server name". Separate multiple servers with a comma.

```
Generating script file for WebSphere Application Server
configuration ...

2003-03-18 14:54:03,191 [main] - TransactionVision
Info(FileUpdated): File "C:\Program
Files\Bristol\TransactionVision\bin\WSSensorSetupEnv.bat"
has been successfully updated

You have following TransactionVision Sensor installed on
this host:

Servlet Sensor
JMS Sensor
EJB Sensor

Instrumenting file C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar for TransactionVision JMS
Sensor ...
```

```
Instrumenting file C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar for TransactionVision JMS
Sensor has completed successfully. Result file is saved in
directory C:\Program
Files\Bristol\TransactionVision\java\lib\

In order to invoke JMS Sensor, you need to put the
instrumented C:\Program
Files\Bristol\TransactionVision\java\lib\com.ibm.mqjms.jar
in front of the original com.ibm.mqjms.jar shipped with
WebSphere MQ in your java CLASSPATH.
```

*Important!* See "Setting the CLASSPATH Environment Variable" for instructions.

```
Do you wish to monitor JMS methods in your Application
Server? (Y/N) [Y]:

Instrumenting file C:\Program
Files\WebSphere\AppServer\lib\webcontainer.jar for
TransactionVision Servlet Sensor ...

Instrumenting file C:\Program
Files\WebSphere\AppServer\lib\webcontainer.jar for
TransactionVision Servlet Sensor has completed
successfully. Result file is saved in directory C:\Program
Files\WebSphere\AppServer\classes\

Adding TransactionVision Servlet and JMS Sensor to
WebSphere Application Server

2003-03-18 14:54:41,846 [main] - TransactionVision
Info(SensorSetupSuccess): SensorSetup has completed
successfully. All program output and user input are logged
in file "C:\Program
Files\Bristol\TransactionVision\logs\setup.log".
```

## Additional WebSphere Procedures for WebSphere Application Server 5.1 Express Edition

If you are using the Servlet or JMS Sensor with WebSphere Application Server 5.1 Express Edition, add the Sensor from the WebSphere Admin console. For detailed instructions, see Appendix E, “Configuring Application Servers for TransactionVision.”

## Setting CLASSPATH for the JMS Sensor

Use the following methods to set the CLASSPATH environment variable correctly for the TransactionVision JMS Sensor, depending on your operating environment:

| Environment | Method |
|---|---|
| Windows | From the Windows Control Panel, choose System > Environment. |
| UNIX C shell | `setenv CLASSPATH $TVISION_HOME/java/lib/tvisionsensor-jms.jar:$TVISION_HOME/java/lib/com.ibm.mqjms.jar:$CLASSPATH` |

## Configuring Correlation for Multithreaded Servlet/JMS Events

If a servlet spins off a thread to make some JMS calls, the servlet passes tracking information to the child thread. The result is that both the servlet and JMS events belong to the same business transaction. However, there may be some cases in which you wish to separate these events into different transactions. For example, a servlet may spin off a long-running thread that you do not want to be part of the same transaction as the servlet. For instructions on changing default transaction correlation behavior for multithreaded servlet/JMS events in the Analyzer, see “Multithreaded Servlet/JMS Events” on page 86.

## Monitoring Stand-alone JMS Applications

To monitor a stand-alone JMS application, set the Java system property `com.bristol.tvision.home` to the top level TransactionVision installation directory in addition to setting the CLASSPATH environment variable. For example, use the following command on Windows:

```
java -Dcom.bristol.tvision.home="C:\Program
Files\Bristol\TransactionVision" my_application
```

When monitoring stand-alone JMS applications, if you are monitoring the listener of the queue manager that you have configured the JMS Sensor to communicate with (using a client connection), do NOT include the WebSphere MQ Sensor in the library path. If this cannot be avoided, use a data collection filter to filter out TVISION.CONFIGURATION.QUEUE and your defined event queue to prevent the collection of the internal JMS Sensor MQSeries events. For more information about data collection filters, see the "Managing Data Collection Filters" chapter of the *TransactionVision Administrator's Guide*.

## Monitoring Stand-alone J2EE Applications

To trace JTA activities in J2EE stand-alone applications, which are started through `<WAS_HOME>/bin/launchClient.bat|sh`, you must modify the `setupCmdLine` script so that it uses the instrumented version of `ibmorb.jar` in `TVISION_HOME/java/lib`. To do this, modify the `<WAS_HOME>/bin/setup/clientLine.bat|sh` script, adding `TVISION_HOME/java/lib/tvjavaext.jar` and `TVISION_HOME/ibmorb.jar` to the beginning of the WAS_BOOTCLASSPATH environment variable, as in the following example:

```
SET WAS_BOOTCLASSPATH=$TVISION_HOME/java/lib/tvjavaext.jar;
$TVISION_HOME/java/lib/ibmorb.jar;$JAVA_HOME\jre\lib\ext\ibmorb.
jar
```

To enable TransactionVision to trace the activity of a stand-alone client application in WebLogic, the `TVISION_HOME\java\lib\tvjavaext.jar` file has to be included in the CLASSPATH of the running environment of that application.

## Reinstrumenting the Sensors

To reinstrument the TransactionVision Servlet, EJB, and JMS Sensors at a later time, or turn the Sensor off/on in the application server, run the **`SensorAdmin`** script. For more information about this script, see Appendix A, "Utilities Reference."

*Important!* If you upgrade either WebSphere MQ or your application server, you **must** reinstrument both the Servlet, EJB, and JMS Sensor with **`SensorAdmin`**.

## Monitoring a Clustered J2EE Application Server

To monitor events on clustered J2EE application servers, perform the following steps:

1. Install the TransactionVision Sensor on all hosts where clustered application servers reside.
2. Run the **`SensorSetup`** script to enable the TransactionVision Sensor on the application servers installed on each machine of the cluster. For WebSphere, this will modify the application server's `server.xml` file for each application server you are monitoring with the TransactionVision Sensor.
3. For WebSphere, copy the `server.xml file` (located in `<WAS_INSTALL>/config/cells/<cell_name>/nodes/<node_name>/servers/<server_name>/server.xml`) to the Deployment Manager's installation area manually from each server on which the Sensor has been enabled, on each machine in the cluster.

   For example, consider the following setup of application servers in which there exist the following cells, nodes and servers:

   - AppServerA located in cell hostcNetwork on HostA,
   - AppServerB located in cell hostcNetwork on HostB,
   - DeploymentServer located in cell hostcNetwork on HostC.

Suppose that AppServerA and AppServerB are in a cluster in cell hostcNetwork under a deployment manager on HostC.
In this scenario, you must copy the following files:

- Copy `<WAS_INSTALL>/AppServer/config/cells/hostcNetwork/nodes/HostA/servers/AppServerA/server.xml` from the machine HostA **TO** `<WAS_INSTALL>/DeploymentManager/config/cells/hostcNetwork/nodes/HostA/servers/AppServerA/server.xml`
- Copy `<WAS_INSTALL>/AppServer/config/cells/hostcNetwork/nodes/HostB/servers/AppServerB/server.xml` from the machine HostB **TO** `<WAS_INSTALL>/DeploymentManager/config/cells/hostcNetwork/nodes/HostB/servers/AppServerB/server.xml`

where `WAS_INSTALL` is typically `C:\Program Files\WebSphere` on Windows or `/usr/WebSphere` on UNIX. It may change based on your WebSphere installation location.

4. Stop and restart the Deployment Manger.
5. Stop and restart all clustered servers you are monitoring.

## Monitoring WebService Events

TransactionVision tracks web service traffic under the WebSphere Application Server and BEA WebLogic by monitoring the appropriate servlet activity.

*Important!* In order to capture WebServices related servlet events, you must run `$TVISION_HOME/bin/tvision-webservices-config.bat(sh)` after you run `SensorSetup.bat(sh)`. This script instruments the `webservices.jar` file under the `WebSphere/lib` directory to allow TransactionVision to collect events from WebServices servlets.

Under WebSphere architecture, incoming web service requests pass through the servlet com.ibm.ws.webservices.engine.transport.http.WebServicesServlet:

The HTTP_POST event for this servlet captures the context of the corresponding call and detail of the SOAP request and response object.

TransactionVision collects the following web service details through the event:

- URI for the web service (for example: /StockQuote/services/urn:xmltoday-delayed-quotes)
- Incoming request data in the event user data section under RequestData
- Response data in the event user in the user data section under Response.

The following is an example of the incoming request SOAP envelope:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getQuote soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"
                xmlns:ns1="http://stock.webservices.samples.websphere.ibm.com">
   <in0 xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/
soap/encoding/">XXX</in0>
  </ns1:getQuote>
 </soapenv:Body>
</soapenv:Envelope>
```

The following is an example of the response SOAP envelope:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getQuoteResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"
```

```
                xmlns:ns1="http://stock.webservices.samples.websphere.ibm.com">
   <return xsi:type="xsd:float">55.25</return>
  </ns1:getQuoteResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

# Chapter 9
# Configuring the CICS Sensor

Once the TransactionVision CICS Sensor is installed, you must configure and then start the Sensor.

## CICS Sensor Overview

The CICS Sensor is implemented in four distinct components: the TransactionVision Manager, the CICS Sensor Manager, the CICS Sensor Driver, and the CICS Sensor Exits Manager.

### TransactionVision Manager (TVM)

TVM is an authorized program that runs as a started task. It controls the other components and provides a framework that allows the other components to communicate with each other. A sample startup procedure, TVISION, is provided in the &hlq,SSLDPROC library. TVM requires one parameter, a four character TVID value that is unique for all concurrently running instances of TVM. The TVID must also be different from the SYSID of any concurrently running CICS Sensor Manager. For future considerations, it is highly recommended that the TVID be unique among all TVMs, CICS SYSIDs, IMS IDs, and all MVS subsystem IDs at your site.

After TVM is started, you issue MVS modify commands to the TVM started task jobname to request it to start and stop CICS Sensor Managers or to shutdown TVM. All the commands are listed in the “CICS Sensor Commands” section.

### CICS Sensor Manager (CSM)

CSM is started as result of a start Sensor command to TVM and runs as a subtask of TVM. CSM controls the monitoring of a single CICS region. A startup parameter, SYSID, must match the SYSID of the CICS region to be monitored. A CSM must be started for each CICS region to be monitored.

CSM creates a buffer queue data space to store CICS events and starts the CICS Sensor Driver.

### CICS Sensor Driver (CDR)

CDR is an unauthorized program that runs as a started task in its own address space. CDR is automatically started by CSM when it starts up and terminated when it shuts down. A parameter on the start Sensor command specifies the procedure to be used to start CDR.A sample startup procedure, TVISIONC, is provided in the &hlq,SSLDPROC library.

CDR communicates with the TransactionVision Analyzer via Websphere MQ. In response to configuration messages from the Analyzer, it indicates to the CICS Sensor Exits Manager 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.

### CICS Sensor Exits Manager (CMX)

CMX is a continuously running CICS transaction. It is recommended that it be started and stopped via PLTPI/PLTSD.

CMX starts and stops standard CICS user exits to capture event data in response to requests from CDR indicating which event types to capture. See the “CICS Exits” section for a more detailed explanation of TransactionVision's use of CICS exits.

## CICS Sensor Commands

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

### Start TransactionVision Manager (TVM)

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

where

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

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

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

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

```
SLDS400I TVISION TransactionVision Manager startup in progress.
SLDS401I TVISION tvid TransactionVision Manager startup complete.
```

### Stop TransactionVision Manager (TVM)

```
STOP jobname
```

where

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

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

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

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

Note: All other Sensor commands are issued as standard MVS modify commands. Most of the command and operand names can be abbreviated to as few characters as required to make the name unique. If the abbreviation is not unique, the alphabetically first command or operand name that fits is assumed. However, some names are reserved for future use. For each command, an alternative is also shown with the minimum abbreviations for the command name and each operand name. Of course, the MVS MODIFY command can be abbreviated to "F."

### Start Sensor

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

where

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

START is the command name and is required.

CICS is the Sensor type and is required.

SYSID is required. sysid must specify the SYSID of the CICS region to be monitored.

QBLKSIZE is optional. *qblksize* specifies the size, in megabytes, of each block in the buffer queue data space. The minimum and default value for *qblksize* is 1; the maximum is 100. See “Buffer Queue Considerations” on page 153 for more information.

MAXQBLKS is optional. *maxqblks* specifies the maximum number of buffer queue blocks that the Sensor is allowed to allocate in its data space. Each queue block is the size specified or defaulted by the *qblksize* parameter. The minimum and default value for *maxqblks* is 3; the maximum is 2046. See “Buffer Queue Considerations” on page 153 for more information.

QMGR is optional. *qmgr* specifies the name of the Websphere MQ queue manager through which the Sensor driver component will communicate with the TransactionVision Analyzer. See “Sensor Driver Startup Procedure” on page 157 for more information on the relationship of this parameter with the Sensor driver startup procedure.

CONFIGQ is optional. *configq* specifies the name of the Websphere MQ queue from which the Sensor driver component will receive configuration messages from the TransactionVision Analyzer. See “Sensor Driver Startup Procedure” on page 157 for more information on the relationship of this parameter with the Sensor driver startup procedure.

DRVRPROC is optional. *drvrproc* specifies the name of the cataloged procedure to start the Sensor driver. See “Sensor Driver Startup Procedure” on page 157 for more information. The default name is TVISIONC.

A Sensor manager and Sensor driver are started to allow monitoring of the CICS region specified on the SYSID parameter. The following messages will be displayed:

```
SLDS400I TVISION CICS sysid TransactionVision sensor startup in
progress.

SLDS401I TVISION CICS sysid TransactionVision sensor startup
complete.

IEF403I drvrproc - STARTED […and other MVS messages issued when
starting a job]

+SLDS279I : TransactionVision Sensor: CICS sysid Sensor Driver
startup completed. QMGR=qmgr, CONFIGQ=configq.
```

Other messages will be issued by the exits manager component, if active, in the CICS region. See “CICS Exits” on page 154 for more information.

## Stop Sensor

```
MODIFY tvm_jobname,STOP CICS SYSID(sysid)
```

where

*tvm_jobname* is the MVS jobname specified or defaulted on the start command for the TVM job that controls the Sensor to be stopped.

STOP is the command name and is required.

CICS is the Sensor type and is required.

SYSID is required. *sysid* must specify the same SYSID that was specified on the start Sensor command for this Sensor.

The Sensor manager and Sensor driver identified by the SYSID parameter are stopped. The Sensor manager enters a quiesce stage to wait for all the events in the buffer queue to be retrieved by the Sensor driver before completing the shutdown. This may take some time. The following messages will be displayed:

```
SLDS404I TVISION CICS sysid TransactionVision sensor termination
in progress.
SLDS443I TVISION CICS sysid Sensor quiescing: n events in buffer
queue.
…
SLDS445I TVISION CICS sysid Sensor quiesce completed.
SLDS448I TVISION CICS sysid Sensor statistics:
    Events in queue:                n
```

```
    Events collected:                n
    Events dispatched:               n
    Events_lost:                     n
+SLDS27BI : TransactionVision Sensor: CICS sysid Sensor Driver is
ending
IEF404I drvrproc - ENDED [,,,and other MVS messages issued when
stopping a job]
SLDS405I TVISION CICS sysid TransactionVision sensor termination
complete.
```

Other messages will be issued by the exits manager component, if active, in the CICS region. See “CICS Exits” on page 154 for more information.

### Inquire Buffer

```
MODIFY tvm_jobname,INQUIRE SYSID(sysid) STATISTICS
or
F tvm_jobname,I S(sysid) ST
```

where

*tvm_jobname* is the MVS jobname specified or defaulted on the start command for the TVM job to which the inquire is directed.

INQUIRE is the command name and is required.

CICS is the Sensor type and is required.

SYSID is required. *sysid* specifies the SYSID for the Sensor whose buffer statistics are to be displayed. An "*" may be specified for the *sysid* to display the buffer statistics of all the Sensors controlled by the TVM.

The inquire buffer command will display the following:

```
SLDS448I TVISION CICS sysid Sensor statistics:
        Events in queue:                 n
        Events collected:                n
        Events dispatched:               n
        Events_lost:                     n
```

## CICS Transactions

### Start CICS Sensor Exits Manager

```
SLDS
```

where SLDS is the CICS transaction code.

The exits manager is started. The following messages will be displayed:

```
+SLDS460I TVISION CICS sysid Sensor exits manager started.
+SLDS464I TVISION CICS sysid Program Start (PS) exit enabled.
+SLDS461I TVISION CICS sysid Sensor exits manager waiting for
sensor startup.
```

Other messages will be issued if a Sensor has already been started to monitor the current CICS region. See “CICS Exits” on page 154 for more information.

### Stop CICS Sensor Exits Manager

```
SLDP
```

where SLDP is the CICS transaction code.

The exits manager is stopped. The following messages will be displayed:

```
+SLDS464I TVISION CICS CIC1 Program Start (PS) exit disabled.
+SLDS465I TVISION CICS CIC1 Sensor exits manager terminated.
```

Other messages will be issued indicating the status of all the TransactionVision CICS user exits. See “CICS Exits” on page 154 for more information.

### System Disable/Enable of User Exits

SLDC {ENABLE | DISABLE| *exitid*

where

SLDC is the CICS transaction code.

ENABLE or DISABLE is the function requested, which may be abbreviated to EN or DIS.

*exitid* is 2-character exit id as follows:

| | |
|---|---|
| KC | Task Control |
| PS | Program Start |
| PC | Program Control |
| FC | File Control |
| TD | Transient Data Control |
| TS | Temporary Storage Control |
| IC | Interval Control |

The identified exit is enabled or disabled. Normally, the status of each user exit is determined by the Sensor driver component in response to requests from the Analyzer for the event type(s) captured by that exit. The MVS or CICS operators can override the Analyzer's control using the SLDC transaction. An exit disabled by SLDC can be re-enabled only by another SLDC transaction. Requests from the Analyzer are ignored for any exit disabled by SLDC. The Program Start (PS) exit is a special case; it must remain enabled for the exits manager to run. When a disable request for PS is issued, the exit is stopped but not disabled. See “CICS Exits” on page 154 for more information. The following messages will be displayed:

```
+SLDS464I TVISION exit name (exitid) exit disable requested.
+SLDS464I TVISION CICS sysid exit name (exitid) exit disabled.
```

or

```
+SLDS464I TVISION exit name (exitid) exit enable requested.
+SLDS464I TVISION CICS sysid exit name (exitid) exit enabled.
```

Other messages may be issued if the Sensor is currently requesting event types captured by the exit. See “CICS Exits” on page 154 for more information.

### Disable All User Exits

```
SLDD
```

where SLDD is the CICS transaction code.

All of the TransactionVision CICS user exits are stopped and disabled. SLDD is intended for emergency use only. If the exits manager fails or you suspect that some exits are still enabled after the exits manager has terminated, run SLDD to assure that all exits are stopped and disabled. SLDD has no dependencies on other TransactionVision components so that it can work regardless of the state of the rest of the Sensor. It does not issue standard TransactionVision messages but it does display its activities on the system log, including any errors it encounters stopping or disabling exits. If errors occur, run it again. When all exits are stopped and disabled the following messages will be displayed:

```
+sysid: SLDPTCX  not started
+sysid: SLDPTCX  not enabled
+sysid: SLDPPCX  not started
+sysid: SLDPPCX  not enabled
+sysid: SLDPICX  not started
+sysid: SLDPICX  not enabled
+sysid: SLDPTDX  not started
+sysid: SLDPTDX  not enabled
+sysid: SLDPTSX  not started
+sysid: SLDPTSX  not enabled
+sysid: SLDPFCX  not started
+sysid: SLDPFCX  not enabled
+sysid: SLDPPSX  not started
+sysid: SLDPPSX  not enabled
```

## CICS Sensor Operations

Please refer to the “CICS Sensor Commands” section above for detailed explanations of the required commands and transactions.

### Start the CICS Sensor

To start the CICS Sensor, perform the following steps:

1. Start the TransactionVision Manager. For example:

   S TVISION,TVID=TV01

   Perform the following steps for each CICS region to be monitored:

2. Start an instance of the CICS Sensor Manager using the start Sensor command specifying the SYSID of the CICS region to be monitored. For example:

   F TVISION,S C S(CIC1)

3. Start the CICS Sensor Exits Manager in the CICS region to be monitored by running the SLDS transaction. If the exits manager is started via a PLTPI entry and not previously stopped, this step should be skipped. For example:

   SLDS

4. At an Analyzer workstation, request collection of the desired events from the monitored CICS region. Please refer to the *TransactionVision Administrator's Guide* and the *TransactionVision User's Guide* for Analyzer operations.

### Stop the CICS Sensor

To stop the CICS Sensor perform the following steps for each monitored CICS region:

1. Stop the CICS Sensor Manager instance using the stop Sensor command specifying the SYSID of the monitored CICS region. For example:

   F TVISION,STO C S(CIC1)

2. Stop the CICS Sensor Exits Manager in the monitored CICS region by running the SLDP transaction. If the exits manager is terminated via a PLTSD entry, this step may be skipped, but run it if you wish to immediately terminate all TransactionVision activity in the CICS region. For example:

   SLDP

   When all sensor instances have been stopped and no further TransactionVision activity is anticipated, perform the following step:

3. Stop the TransactionVision Manager. For example:

   P TVISION

## Notes

1. The TransactionVision Manager must be started before any CICS Sensor Managers are started.

2. The TransactionVision Manager is started with a TVID parameter that uniquely identifies the instance. Multiple concurrent TransactionVision Managers may be started, which may be desirable for test and production environments, for example. However, within the same environment, resources will be most efficiently utilized and operations simplified by running multiple sensor managers under control of one TransactionVision Manager.

3. There is a one-to-one correspondence between a CICS Sensor Manager and a CICS Sensor Exits Manager. It makes no difference which is started first. The exits manager runs as a CICS transaction in a CICS region with a specific, unique SYSID value. The corresponding sensor manager is started with that same SYSID value as a parameter. Only one Sensor manager is allowed to use a particular SYSID.

4. The prohibition of SYSID reuse will be enforced across all instances of the TransactionVision Manager, so you cannot start a sensor to monitor a CICS region if a sensor is already monitoring that CICS region regardless of which TransactionVision Manager instances are involved.

5. The TVID must not be the same value as the SYSID value of any Sensor manager. For future considerations, it is highly recommended that the TVID be unique among all TransactionVision Manager TVIDs, CICS SYSIDs, IMS IDs, and all MVS subsystem IDs at your site.
6. Stopping the TransactionVision Manager will automatically stop all the Sensors (CICS Sensor Manager instances) under its control.
7. When a Sensor is stopped, the Sensor manager will signal its corresponding Sensor exits manager to stop all the Sensor's CICS exits to cease further data capture, then enter quiesce mode until the Sensor driver retrieves all the events in the buffer queue. When all events have been retrieved, which may take some time, the Sensor driver will terminate, then the Sensor manager will terminate.
8. TransactionVision Manager termination will not complete until all Sensors under its control have terminated.
9. If you cancel the TransactionVision Manager, all Sensors under its control will immediately terminate and all remaining events in the buffer queue will be discarded.

## Buffer Queue Considerations

To minimize the overhead of its CICS exits, the TransactionVision CICS Sensor stores all captured CICS events in a data space referred to as the buffer queue. Each CICS Sensor Manager has exclusive use of a data space. Therefore, each buffer queue contains events from a single CICS region.

The buffer queue is configured as a number of queue blocks of a certain size. Both of these dimensions are specified in parameters on the start Sensor command. MAXQBLKS specifies the maximum number of queue blocks and QBLKSIZE specifies the size of each queue block in megabytes. The maximum size of the data space used, in megabytes, is the product of MAXQBLKS and QBLKSIZE and must not exceed 2 GB. The default and minimum MAXQBLKS is 3, and the default and minimum QBLKSIZE is 1. Therefore, the minimum data space size requirement is 3MB, which would be suitable only for low volume testing.

The correct size of the buffer queue varies widely depending on the transaction throughput of the monitored CICS region, the number, types, and sizes of events collected, the throughput of the CICS Sensor Driver, and the responsiveness of the buffer queue management functions in the CICS Sensor Manager. These variables will be discussed in more detail later, but before attempting laborious calculations using what may be mere guesses as your parameters, consider taking a trial and error approach. In any event, understanding TransactionVision's use of the buffer queue is necessary.

At Sensor startup, three queue blocks are allocated. When the TransactionVision CICS exits are started, they store events into the first queue block and then to the next, etc. Concurrently, the CICS Sensor Driver is retrieving events from the buffer queue in the same sequential order that they were stored (FIFO). The buffer queue management functions are invoked every tenth 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 maximums 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.

## CICS Exits

The TransactionVision CICS Exits are the most critical component of the Sensor because they run in your CICS region and must not jeopardize that environment.

### Introduction

TransactionVision uses standard CICS exit programming techniques, which are fully documented in the IBM manual, *CICS Customization Guide*. There are two types of user exits: global user exits (GLUEs) and task-related user exits (TRUEs). GLUEs are associated with CICS APIs; TransactionVision uses them to intercept program control, interval control, file control, transient data control, and temporary storage control APIs. TransactionVision also uses one TRUE to capture task start and task end events.

User exits are driven at their associated exit points only when they are both enabled and started. Exit programs are enabled, disabled, started, and stopped via CICS commands. The exits manager (discussed below) performs these functions as required.

### Error protection

Since abends in user exits programs can cause CICS to abend, each TransactionVision CICS exit program implements an abend handler, which traps the abend, issues an error message, and disables the exit so that it will not be driven again. Each exit program also takes the same action for any non-abend errors it detects. Thus, CICS itself is protected from any errors in the exit programs. The system operator may re-enable an exit that has disabled itself due to an error.

### Sensor Exits Manager

The Sensor Exits Manager is started in the CICS region by the SLDS transaction or by a PLTPI entry. It is terminated by the SLDP transaction or a PLTSD entry. The SLDS transaction actually executes a small program, SLDPCSX, which issues a CICS START command to start the sensor manager as a non-facility task. The START command requires a transaction id, so the exits manager has a transaction id of SLDM associated with it. You should not use the SLDM transaction because the exits manager is a continuously running task and will, therefore, tie up the terminal from which the SLDM transaction is issued for as long as it runs; use the SLDS transaction instead.

Whether the exits manager is started via a PLTPI entry at CICS startup or by invoking the SLDS transaction, it may be stopped by invoking the SLDP transaction. When the exits manager is stopped, all the exits that it controls are also stopped and disabled. The exits manager may be restarted by the SLDS transaction. If the SLDS transaction is issued when the exits manager is already started or the SLDP transaction is issued when the exits manager is not running, an error message is issued and the request is ignored.

When the exits manager is started, it immediately enables the program start exit because a global work area is associated with that exit. The global work area is required for communications between the exits manager and all the exits. If the CICS Sensor Manager is already started the exits manager enables all the remaining exits; otherwise, it waits until the Sensor manager is started, then enables them.

After all exits have been enabled, the exits manager queries the TransactionVision Analyzer, communicating through the sensor driver component, to determine which exits are required to capture the event types requested by the Analyzer. Once per second the exits manager checks the Analyzer requests. When an exit is required and not already started, the exits manager starts it. When an exit is not required but is started, the exits manager stops it.

### System Operator Control

The system operator may override Analyzer requests via the SLDC transaction discussed in the “CICS Transactions” section. Any exit disabled by the system operator will not be driven regardless of requests from the Analyzer. This block can be removed only by the system operator issuing another SLDC transaction to enable the exit.

Note that the program start cannot be disabled by the SLDC transaction because, as previously noted, this exit is required for the exits manager to continue running. An SLDC request to disable the program start exit will cause it to be stopped so that it will not be driven and ineligible to be started by Analyzer requests until the block is removed via SLDC ENABLE PS.

An additional control is provided by the SLDD transaction, which is fully documented in the “CICS Transactions” section. Because the SLDD transaction forces the disablement of all the TransactionVision exits, including the program start exit on which the exits manager depends, the exits manager also terminates if it is running.

## TransactionVision Manager Startup Procedure

When you start the TransactionVision Manager (TVM) you must assign it a TVID that is unique among all concurrent instances of TVM (see the "CICS Sensor Commands" section above). The TVID is passed as the single parameter to the main program, SLDPTVM. A sample procedure, TVISION, is provided to start TVM as follows:

```
//TVISION  PROC TVID=TV01,TVHLQ=TVISION
//TVISION  EXEC PGM=SLDPTVM,TIME=1440,PARM='&TVID'
//STEPLIB  DD   DISP=SHR,DSN=&TVHLQ..SSLDAUTH
//         PEND
```

The TVID is defined as a procedure keyword parameter and defaults to TV01. A “START TVISION” operator command will start TVM with a TVID of TV01. A “START TVISION,TVID=TV02” operator command will start TVM with a TVID of TV02. If both commands are issued two instances of TVM are started and both have the same jobname, TVISION.

To stop TVM or to start and stop Sensors under control of TVM requires issuing STOP or MODIFY commands specifying the jobname of the TVM to which the command is directed. If multiple TVMs are started with the same jobname, the command is directed to all of them. To avoid this problem specify a unique jobname on the START command. It is recommended to use the TVID as the jobname. For example:

```
START TVISION,JOBNAME=TV01,TVID=TV01
START TVISION,JOBNAME=TV02,TVID=TV02
```

## Sensor Driver Startup Procedure

When you issue a start sensor command to TVM, the Sensor driver component is started in a separate address space. The Sensor manager automatically starts the Sensor driver using the procedure specified on the start Sensor command (see the "CICS Sensor Commands" section). The following parameters from the start Sensor command are passed to the Sensor driver procedure:

| Parameter | Default |
|---|---|
| SYSID | None |
| QMGR | None |
| CONFIGQ | None |

The DRVRPROC parameter on the start Sensor command specifies the procedure to be invoked to start the Sensor driver. The default procedure name is TVISIONC, a sample for which is provided as follows:

```
//TVISIONC PROC SYSID=,QMGR=CSQ1,
//              CONFIGQ=TVISION.CONFIGURATION.QUEUE,
//              TVHLQ=TVISION,MQHLQ=CSQ520

//TVISIONC EXEC PGM=SLDPCDR,TIME=1440,PARM='&SYSID &QMGR
&CONFIGQ'
//STEPLIB  DD  DISP=SHR,DSN=&TVHLQ..SSLDLOAD
//       DD  DISP=SHR,DSN=&MQHLQ..SCSQAUTH
//       PEND
```

The SYSID, QMGR, and CONFIGQ are defined as procedure keyword parameters with the indicated defaults. Remember that this procedure is automatically invoked by the Sensor manager; you should not invoke this procedure by any other means. For example, you should not issue a "START TVISIONC" command.

You may customize this procedure or create multiple procedures according to the communications links between the Sensor driver and TransactionVision Analyzers. The Websphere MQ queue manager (QMGR) and configuration queue (CONFIGQ) parameters determine which Analyzer(s) communicate with the Sensor driver. In the sample procedure, the QMGR and CONFIGQ parameters can be overridden by specification of the same parameters on the start Sensor command and the SYSID parameter must be supplied from the start Sensor command. The SYSID, QMGR, and CONFIGQ parameters are passed to the Sensor driver main program, SLDPCDR, as space-delimited positional parameters, which allows the omission of only the last parameter, CONFIGQ. If you omit this parameter, it defaults to TVISION.CONFIGURATION.QUEUE. When customizing this procedure or creating your own Sensor driver procedures, you should observe the following conventions:

- Always specify all three parameters—SYSID, QMGR, and CONFIGQ—as procedure parameters.
- Always specify a SYSID parameter default of null.
- Always specify a non-null default QMGR parameter.
- Always pass at least the SYSID value and the QMGR value in the PARM field to SLDPCDR.

# Chapter 10
# Configuring TransactionVision Component Logging

## Log Files

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

- WebSphere MQ Sensors log error messages in the UNIX system log, the Windows event log, the OS/390 operator console log, or the OS/400 user job log.
- 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.
- The web user interface logs messages to the `ui.log` file.
- The Servlet, EJB, and JMS Sensors log these messages to the `sensor.log` file.

*Important!* To enable the servlet and JMS Sensors to print banners when activated, set the `com.bristol.tvision.sensor.banner` Java property to true. The banner is printed to standard out.

## Circular Logging

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

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

- `ui.log.2` is removed if it exists
- `ui.log.1` is renamed `ui.log.2`
- `ui.log` is renamed `ui.log.1`
- A new `ui.log` is created

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

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

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

### Maximum Log File Size

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

### Maximum Number of Backup Log Files

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

### Changing from Circular to Linear Logging

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

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

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

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

## Trace Logging

Trace logging provides verbose information of what a TransactionVision component (the Sensor, Analyzer or web user interface) is doing internally. It is used mainly to troubleshoot problems and should not be turned on in production environments.

### Sensors

You can enable trace logging in TransactionVision Sensors to debug Sensor configuration issues. Trace information for the WebSphere MQ Sensor is logged to the UNIX system log, the Windows event log, the OS/390 operator console log, or the OS/400 user's job log. For the Servlet, EJB, and JMS Sensors, trace messages are sent to the Sensor's log4j TraceLog.

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

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.

### Analyzer

To enable trace logging for the TransactionVision Analyzer, set the value of the `trace` property in `<TVISION_HOME>/config/services/Analyzer.properties` file to `on`. After modifying this configuration file, you must restart the Analyzer for the change to take effect.

### TransactionVision Web User Interface

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

## Configuring Separate Log Files for Multiple Sensor Instances

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

To set up each Sensor instance to log ot 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 stting up the Sensor for two servers, serverA and serverB, copy `Sensor.properties` to `Sensor_serverA.properties` and `Sensor_serverB.properties`.

2. Copy the `<TVISION_HOME>/config/logging/Sensor.Logging.xml` file to another name in the `<TVISION_HOME>/config/logging` directory. For example, for each server (serverA and serverB), copy this file to `Sensor.Logging.serverA.xml` and `Sensor.Logging.serverB.xml`.

3. Modify the `logging_xml` property in each `Sensor.properties` file. For example, in `Sensor_serverA.properties`, change the property line to `logging_xml=Sensor.Logging.serverA.xml`. Similarly, in `Sensor_serverB.properties`, change the property line to `logging_xml=Sensor.Logging.serverB.xml`.

4. Set the JVM property `com.bristol.tvision.sensor.properties` to the appropriate `Sensor.property` file. For example, for serverA set the JFM property as follows:

   ```
   com.bristol.tvision.sensor.properties=sensor/Sensor_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 cosole for the given aplication server under Servers > Application Servers > Process Definition > Java Virtual Machine > Custom Properties.

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

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

## Using Windows and UNIX System Logs

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

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

### Windows Event Appender

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

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

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

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

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

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

### UNIX Event Appender

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

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

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

# Chapter 11
# Integrating TransactionVision with Tivoli Enterprise Console

This section describes how TransactionVision integrates with the Tivoli Enterprise Console (TEC). This integration allows TransactionVision to send events into TEC.

Events generated by TransactionVision can be of two types:

- Internal events, generated by TransactionVision components, Analyzer and UI
- Application events, generated by plugin beans in the Analyzer or scheduled jobs in the user interface components of TransactionVision

## TEC Integration Setup

Setting up TransactionVision's integration with TEC involves TEC server side setup as well as TransactionVision Analyzer and user interface setup.

### TEC Server Side Setup

TEC server side setup enables the reception of TransactionVision events (BTV) into TEC by readying the current TEC rulebase, or by creating a new TEC rulebase by cloning the currently loaded rulebase with the addition of the appropriate BTV class definitions and ruleset. In either scenario, you may load the newly configured rulebase when you run **`Install.sh`**, or at a later time.

### TransactionVision Analyzer and User Interface Setup

Error messages as well as audit logs from the TransactionVision Analyzer and user interface components can be routed to TEC. TransactionVision uses the log4j package (http://logging.apache.org/log4j/docs) for all logging. By default, these log messages are sent into text files. TransactionVision provides an appender (TECAppender) to route these messages into TEC.

## Installation and Setup

### TEC Server Side Setup

*Important!* Perform the TEC server side setup **before** enabling the TransactionVision Analyzer or user interface for generating TEC events. Otherwise, events of undefined classes can be passed to TEC. The TEC server side setup can be done independently of the TransactionVision side setup.

#### Installation Requirements:

- Run this Pac at the TEC server.
- Unix UID with privileges to execute into, and write to the TEC rulebase directory tree
- Tivoli Admin with at least "Senior" authorization

#### Instructions

1. Untar the file `<TVISION_HOME>/tivoli/TVisionTEC.tar` in a scratch directory. The scratch directory will be populated with the following files:
   - Install.sh
   - README
   - btv.baroc
   - btv.rls
   - btv_escalate.sh
   - btv_take_action.sh
2. Run Install.sh locally from within that directory. Follow online prompts.
3. If utilizing ITM, download TVision_ITM_RM.tar from the directory <TVISION_HOME>/tivoli on the Analyzer installation host to the ITM host. Copy it to the resource model tar directory (ex. /usr/local/Tivoli/bin/lcf_bundle.40/Tmw2k/Rm) and import into ITM.

After install, the scratch directory and all its contents may be deleted.

Additional Information:

The btv.rls ruleset will be loaded by default as "last in line" in the rule_sets and rule_sets_EventServer datasets. A class definition file (btv.baroc) and ruleset (btv.rls) will be loaded. If manual loading of these files is preferred, do not run the **`Install.sh`** script.

The rules are delivered in a "commented out" state, and need to be uncommented and recompiled and loaded for use as desired. Ruleset include dupe detect, 1st action, and escalation rules. The action and escalation scripts should be inserted into the TEC scripts directory. Rules and action scripts need to be tailored to your environment before uncommenting. This does not stop the installation of the class definitions and the readying of the TEC server to accept TransactionVision events.

## TransactionVision Analyzer and User Interface Setup

To enable TransactionVision to write events into TEC, you must modify log4j configuration files to use the TECAppender.

1. Edit the Analyzer log4j configuration file `<TVISION_HOME>/config/logging/Analyzer.Logging.xml`.

   To enable error logs to be sent into TEC, add the line in bold below:

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

   To enable audit logs, where every start and stop action is recorded to TEC, add the line in bold below:

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

2. Edit the file `<TVISION_HOME>/bin/SetupEnv.[sh|bat]` to add the following lines:

On Windows, `SetupEnv.bat` (at the end of the file):

```
set CLASSPATH=%CLASSPATH%;%TVISION_HOME%\java\lib\evd.jar
set CLASSPATH=%CLASSPATH%;%TVISION_HOME%\java\lib\tivlog.jar
```

On UNIX, SetupEnv.sh (before the last line in the file):

```
CLASSPATH=$CLASSPATH:$TVISION_HOME/java/lib/evd.jar
CLASSPATH=$CLASSPATH:$TVISION_HOME/java/lib/tivlog.jar
```

3. Restart the Analyzer for this change to take effect.

4. Edit the UI log4j configuration file `<TVISION_HOME>/config/logging/UI.Logging.xml` on the machine you are running the user interface component on your application server.

   To enable error logs to be sent into TEC, add the line in bold below:

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

   To enable audit logs, where every page action by the user is recorded into TEC, add the line in bold below:

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

5. Restart the TransactionVision application loaded within your application server.

6. Edit the file `<TVISION_HOME>/config/logging/tivoli/btv_tec.cfg` file to point to the TEC server host.

```
BufEvtPath=<TEMPDIR>/btv_tec.cache
LogFileName=<TEMPDIR>/btv_tec.log
LogLevel=ALL
```

```
ServerLocation=<TEC server name>
ServerPort=0
```

Errors from the log4j TECAppender go into `<TVision LOGDIR>/tecappender.log` (typically `<TVISION_HOME>/logs/tecappender.log`, look in the file `<TVISION_HOME>/config/setup/Setup.properties` to see where this has been set to).

7. A monitoring profile that allows Tivoli ITM to monitor the healthy functioning of the TransactionVision Analyzer process is also available.

   Import the supplied Resource Model (`TVisionITM_RM.tar`) into ITM (for example, **`wdmrm - add TVisionITM_RM.tar`**) and distribute it as usual with respect to ITM. This Resource Model checks the state of the Analyzer and sends an alert to TEC if the Analyzer is deemed down. It is based on ITM V5.1.1

# Appendix A
# Utilities Reference

## CreateSqlScript

Location:

```
TVISION_HOME/bin/CreateSqlScript.[sh|bat]
```

**Purpose**:

Allow the user to create and optionally execute a SQL script to create, drop, import or export TransactionVision system tables or project tables.

Syntax:

```
CreateSqlScript
    {-create(-c) | -drop(-d) | -import(-i) | -export(-ex)
| -resetseq(-r)}
    {-system(-s) | -project(-p) SCHEMA | -table(-t) TABLE
SCHEMA}
    [[-noscript(-n)] -execute(-e)]
    [-tablespace(-ts) TABLESPACE]
    [-fileType(-f) IXF|DEL] [-lobPath(-lp) PATH]
    [-dbproperties(-db) FILE]
```

Options:

| Option | Description |
|---|---|
| `-drop (-d)` | Drop tables |

| Option | Description |
|---|---|
| `-create (-c)` | Create tables |
| `-execute (-e)` | Execute script |
| `-noscript (-n)` | No script generation |
| `-system (-s)` | Create/drop system tables |
| `-project (-p) SCHEMA` | Create/drop project tables in schema SCHEMA |
| `-table (-t) TABLE SCHEMA` | Create/drop table TABLE in schema SCHEMA. |
| `-tablespace (-ts) TBSPC` | Use tablespace TBSPC. |
| `-dbproperties (-db) FILE` | Use Database.properties file FILE. |
| `-import (-i)` | Generates a database import script. To run database scripts, use the command `db2 -n -t -f <sql script filename>`. For a DB2 database, you can combine this option with the `-fileType` and `-lobPath` options to customize the data format and the location of LOB. You may NOT combine this option with the `-noscript` or `-execute` options. |
| `-export (-ex)` | Generates a database export script. To run database scripts, use the command `db2 -n -t -f <sql script filename>`. For a DB2 database, you can combine this option with the `-fileType` and `-lobPath` options to customize the data format and the location of LOB. You may NOT combine this option with the `-noscript` or `-execute` options. |
| `-resetseq (-r)` | Resets the sequence start number to match to match the imported data. |

| Option | Description |
|---|---|
| `-fileType (-f) IXF\|DEL` | Specify the DB2 data output file format used for importing/exporting data. The default type is IXF. For the IXF file type, the import/export script uses the LOBFILE option for rows that contain greater than 32K data. For the DEL type, the import/export script exports LOBFILEs into a single file (requires FixPack 8). |
| `-lobPath (-lp) PATH` | Specifies the directory of LOBFILEs. The default value is the current directory. |
| `-noLob (-nl)` | Do not generate DB2 export/import SQL with LOBINFILE option. This option will truncate LOB data to the first 32K bytes. |

Examples:

1. Create system tables with schema as TVISION and execute the procedure without generating SQL script:

   ```
   CreateSqlScript –e –n –c –s
   ```

2. Generate SQL script for creating project tables with schema as PROJECT without executing the procedure:

   ```
   CreateSqlScript –c –p PROJECT
   ```

3. Drop table EVENT in schema PROJECT and execute the procedure without generating SQL script:

   ```
   CreateSqlScript –e –n –d –t EVENT PROJECT
   ```

## DB2RunStats

Location:

```
TVISION_HOME/bin/DB2RunStats.[sh|bat]
```

Description:

Updates statistics about the physical characteristics of a table and the associated indexes. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

This command is called when a table has had many updates, such as when data is continuously collected into DB2 by the TransactionVision Analyzer. It could result in large performance gains in queries made by TransactionVision views and reports, as well as queries made internally by the TransactionVision Analyzer to correlate events.

This script can be set up to run as a scheduled batch job using either the UNIX **`cron`** facility or the Windows scheduler.

*Important!* While this script is running, TransactionVision Analyzer processing slows down.

This script typically should be run daily, though the frequency of execution could be higher for higher message rates.

Run this script from a user account that has privileges to perform database operations. This script must be run under a DB2 database instance account.

*Important!* This script needs to be customized based on your system to invoke the correct environment initialization script like .profile or .bashrc, and to set the correct DB2 installation location and the correct TVISION_HOME location.

Syntax:

`DB2RunStats` *`database_name schema_name`* `[-v7]`

Options:

| Option | Description |
|---|---|
| `database_name` | The name of the database to connect to |
| `schema_name` | The name of the schema that the project reads and writes data to |
| `-v7` | Use in a DB2 7.x environment. The default operatin is for DB2 8.1. |

## DB2Test

Location:

`com.bristol.tvision.admin.DB2Test`

**Description:**

Measures DB2 database INSERT performance.

The utility inserts sample event data into the RAW_EVENT table of the specified schema. Before running the test, create a new project schema with CreateSqlScript (which can be deleted after running the test).

Syntax:

```
DB2Test databaseName user passwd schema eventCount even-
tSize threadCount {-batch n} {-noseq}
```

Options:

| Option | Description |
|---|---|
| `databaseName` | Name of the DB2 Database that contains the schema to be used for the test. |
| `user` | DB2 user name |
| `passwd` | DB2 password |
| `schema` | TransactionVision schema in which sample event data will be saved. |
| `eventCount` | Number of events to generate. |
| `threadCount` | Number of threads to use to generate events. |
| `-batch` *`n`* | Executes every n insert statements as part of a single batch operation. Default is to commit each insert individually. |
| `-noseq` | Causes event_id sequence numbers to be maintained by the test utility. Default is to obtain next event_id sequence number by querying the database. |

## MigrateConfig

Location:

`TVISION_HOME/bin/MigrateConfig.[sh|bat]`

Description:

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

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

## MigrateDB

Location:

`TVISION_HOME/bin/MigrateDB.[sh|bat]`

Description:

Migrates project database files from an older version of TransactionVision to the current version. This script may be run by the TransactionVision installation process, or at a later time. It must be run in configured TransactionVision environment; the `Database.properties` must be set correctly for communication with the database.

Syntax:

```
MigrateDB
```

## OracleRunStats

Location:

`TVISION_HOME/bin/OracleRunStats.[sh|bat]`

Description:

Updates statistics about the physical characteristics of a table and the associated indexes. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

This command is called when a table has had many updates, such as when data is continuously collected into Oracle by the TransactionVision Analyzer. It could result in large performance gains in queries made by TransactionVision views and reports, as well as queries made internally by the TransactionVision Analyzer to correlate events.

This script can be set up to run as a scheduled batch job using either the UNIX **`cron`** facility or the Windows scheduler.

*Important!* While this script is running, TransactionVision Analyzer processing slows down.

This script typically should be run daily, though the frequency of execution could be higher for higher message rates.

Run this script from a user account that has privileges to perform database operations. This script must be run under an Oracle user account.

*Important!* This script needs to be customized based on your system to invoke the correct environment initialization script like .profile or .bashrc, and to set the correct Oracle installation location and the correct TVISION_HOME location. Additionally, the file `OracleRunStats.sql` must be in the current directory when running this script.

Syntax:

`OracleRunStats` *`user_name passwd database_name schema_name`*

Options:

| Option | Description |
|---|---|
| `user_name` | The Oracle user account to run the script under. |
| `passwd` | The password associated with *user_name* |
| `database_name` | The name of the database to connect to |
| `schema_name` | The name of the schema that the project reads and writes data to |

## OracleTest

### Location:

`com.bristol.tvision.admin.OracleTest`

### **Description:**

Measures Oracle database INSERT performance.

The utility inserts sample event data into the RAW_EVENT table of the specified schema. Before running the test, create a new project schema with CreateSqlScript (which can be deleted after running the test).

### Syntax:

```
OracleTest databaseName host port user passwd schema
eventCount eventSize threadCount {-VARCHAR} {-BLOB} {-LON-
GRAW} {-batch n} {-thin} {-noseq} {-parallel} {-url URL}
```

### Options:

| Option | Description |
|---|---|
| `databaseName` | Name of the Oracle database that contains the schema to be used for the test. |
| `host` | Name of host system on which Oracle server exists |
| `user` | Oracle user name |
| `passwd` | Oracle password |
| `schema` | TransactionVision schema in which sample event data will be saved. |
| `eventCount` | Number of events to generate. |
| `eventSize` | Size of event user data buffer (default is 1024 bytes) |
| `threadCount` | Number of threads to use to generate events. |
| `-VARCHAR` | Use this option of the RAW_EVENT table has been created with a VARCHAR column definition. |

| Option | Description |
|---|---|
| `-BLOB` | Use this option of the RAW_EVENT table has been created with a BLOB column definition. |
| `-LONGRAW` | Use this option of the RAW_EVENT table has been created with a LONGRAW column definition. |
| `-batch n` | Executes every n insert statements as part of a single batch operation. Default is to commit each insert individually. |
| `-thin` | User thin client driver. Default is to use oci client driver. |
| `-noseq` | Causes event_id sequence numbers to be maintained by the test utility. Default is to obtain next event_id sequence number by querying the database. |
| `-parallel` | Use the Oracle INSERT PARALLEL option instead of the standard INSERT INTO. |
| `-url URL` | By default, OracleTest will use an appropriate JDBC URL for thin or oci client drivers. However the default may be overwridden by specifying the JDBC URL here. |

## rebind_sensor

### Location:

`TVISON_HOME/bin/rebind_sensor.sh`

### Description:

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

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

To work around this problem, run the **`rebind_sensor`** script whenever a WebSphere MQ support pac that modifies the WebSphere MQ libraries (libmqm.a, libmqic.a, libmqm_r.a, libmqic_r.a) are modified.

It modifies the TransactionVision Sensor libraries in TVISION_HOME/lib.

Syntax:

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

Options:

| Option | Description |
|---|---|
| `-v` | Writes errors to the console. The default behavior is to write errors to `TVISION_HOME/logs/mqsensorbind.log`. |
| `-s` | Uses silent mode, which does not prompt before executing. |
| `-h` | Displays usage message. |

## SensorAdmin

Location:

```
TVISON_HOME/bin/SensorAdmin.[sh|bat]
```

Description:

For IBM WebSphere application servers, this script provides the following menu selections if the Servlet or EJB Sensor is installed:

| Menu Selection | Description |
|---|---|
| 1. Instrument Servlet Sensor | Instruments only those files required for the Servlet Sensor. |
| 2. Instrument WebSphere EJB Sensor | Instruments only those files required for the EJB Sensor. |
| 3. Instrument WebSphere MQ JMS Sensor | Instruments only those files required for the JMS Sensor. |
| 4. Add Servlet Sensor to WebSphere Application Server | Adds the Servlet Sensor to WebSphere Application Server without the EJB Sensor. |

| Menu Selection | Description |
|---|---|
| 5. Add Servlet and EJB Sensors to WebSphere Application Server | Adds both the Servlet and EJB Sensors to WebSphere Application Server. |
| 6. Add Servlet, EJB, and JMS Sensors to WebSphere Application Server | Adds the Servlet, EJB, and JMS Sensors to WebSphere Application Server. |
| 7. Remove EJB Sensor from WebSphere Application Server | Removes the EJB Sensor from WebSphere Application Server, along with all EJB Sensor required files from the `WAS/classes` directory. |
| 8. Remove All Sensors from WebSphere Application Server | Removes the Servlet, EJB, and JMS Sensors from WebSphere Application Server, along with all instrumented files from the `WAS/classes` directory. |

This script must be run as root or Administrator with privileges to modify the WebSphere Application Server directory area.

You may use these menu options in combination to achieve desired results. For example, to have only the Servlet and JMS Sensors installed on WebSphere Application Server, choose the following menu selections:

- Option 1 to instrument the Servlet Sensor
- Option 3 to instrument the JMS Sensor
- Option 6 to install all Sensors
- Option 7 to remove the EJB Sensor

If you simply want to monitor JMS calls from your web application, add `TVISION_HOME/java/lib/tvisionsensorjms.jar` to the classpath of your JVM settings.

After enabling both the Servlet and JMS Sensor in your web server, if you want to turn off the JMS Sensor, change the directory for `com.ibm.mqjms.jar` to its original path (where WebSphere MQ Java is installed) in your classpath for the web server's JVM settings.

For BEA WebLogic application servers, this script provides the following menu selections if the Servlet or EJB Sensor is installed:

| Menu Selection | Description |
|---|---|
| 1. Instrument Servlet and EJB Sensors | Instruments those files required for the Servlet and EJB Sensors. |
| 2. Instrument WebSphere MQ JMS Sensor | Instruments only those files required for the JMS Sensor. |
| 3. Add Servlet Sensor to WebLogic Application Server | Instructs you to restart your admin server using `tvStartWebLogic.cmd`. |
| 4. Add Servlet and EJB Sensors to WebLogic Application Server | Instructs you to set JAVA_OPTIONS to true in `tvStartWebLogic.cmd` and restart your admin server with this script. |
| 5. Add Servlet, EJB, and JMS Sensors to WebLogic Application Server | Instructs you to restart the admin server with `tvStartWebLogic.cmd`. |
| 6. Remove EJB Sensor from WebLogic Application Server | Instructs you to set JAVA_OPTIONS to false in `tvStartWebLogic.cmd` and restart your admin server. |
| 7. Remove All Sensors from WebLogic Application Server | Instructs you to restart the admin server iwth `tvStartWebLogic.cmd`. |

This script must be run as root or Administrator with privileges to modify the WebSphere Application Server directory area.

You may use these menu options in combination to achieve desired results. For example, to have only the Servlet and JMS Sensors installed on WebSphere Application Server, choose the following menu selections:

- Option 1 to instrument the Servlet Sensor
- Option 3 to instrument the JMS Sensor
- Option 6 to install all Sensors
- Option 7 to remove the EJB Sensor

If you simply want to monitor JMS calls from your web application, add `TVISION_HOME/java/lib/tvisionsensorjms.jar` to the classpath of your JVM settings.

After enabling both the Servlet and JMS Sensor in your web server, if you want to turn off the JMS Sensor, change the directory for `com.ibm.mqjms.jar` to its original path (where WebSphere MQ Java is installed) in your classpath for the web server's JVM settings.

Syntax:

```
SensorAdmin
```

## SensorSetup

Location:

```
TIVISON_HOME/bin/SensorSetup.[sh|bat]
```

Description:

This utility is launched automatically when the Servlet, EJB, or JMS Sensor is installed. It must be run as root or Administrator with privileges to modify the WebSphere Application Server directory area. It prompts the user to enter the following information:

- The log file directory
- The installation directory of WebSphere MQ and WebSphere or WebLogic
- Information required by WebSphere or WebLogic configuration if the Servlet or EJB Sensor is installed, such as the application server name, admin server host, and admin server port.
- The WebSphere MQ configuration queue manager and configuration queue names.

If the Servlet or EJB Sensor is installed, it does the following:

- For WebSphere, instruments `WAS_HOME/lib/webcontainer.jar` and puts the resulting .jar file under `WAS_HOME/classes`.
- For WebLogic, instruments `WL_HOME/server/lib/weblogic.jar` and `WL_HOME/server/lib/webservices.jar`, and puts the resulting .jar file under `TVISION_HOME/java/lib`.
- Adds required JVM settings to the monitored application server.
- Turn on the Sensor in the application server.

If the JMS Sensor is installed, it instruments `com.ibm.mqjms.jar` and puts the resulting `.jar` file under `TVISION_HOME/java/lib`.

Syntax:

```
SensorSetup
```

## ServicesManager

Location:

```
TVISION_HOME/bin/ServicesManager.[sh|bat]
```

**Purpose**:

Manage the TransactionVision Analyzer service. The Analyzer uses an embedded RMI registry so that Analyzers may be controlled by the TransactionVision web user interface running on remote hosts.

**Syntax**:

```
ServicesManager
  -start [-project (-proj) PROJECTNAME]
  -stop |-exit [-quiesce][-project (-proj) PROJECTNAME]
  -status [-project (-proj) PROJECTNAME]
  -killserver
  -logfile
  -versioninfo
  {[-host HOST][-rmiregp PORTNUMBER][-debug]}
```

Options:

| Option | Description |
|---|---|
| `-start` | Starts the Analyzer process if it is not already running. |
| `-stop` | The Analyzer stops collecting event data. |
| `-exit` | The Analyzer stops collecting event data, then the process exits. |

| Option | Description |
|---|---|
| `-quiesce` | The default behavior of the Analyzer on a stop or exit is to immediately close down collection. If this flag is set, the Analyzer clears out any pending events in the event queue before stopping or exiting. Note that if there is a large event backlog, the Analyzer may take some time to stop or exit. |
| `-status` | Reports the current Analyzer status. |
| `-logconfig` | Reloads Analyzer logging configuration settings. |
| `-versioninfo` | Returns Analyzer version information. |
| `-killserver` | Shuts down the Analyzer immediately. This flag is not recommended; -exit is the preferred method for shutting down the Analyzer cleanly. |
| `-project (-proj) PROJECTNAME` | Name of the project for the specified command. If the project name contains a space, enclose the project name in double quotation marks (for example, `-proj "Project Name"`). The project must have a communication link in order for the Analyzer to process events. Only the -start, -stop, and -status commands be performed on a single project. This option cannot be used in combination with the -host or -rmiregp options. When the -project option is used, the Analyzer host and port is looked up from the database. |
| `-host HOST` | Name of the host where the Analyzer runs. Can be either name or IP address. Local host is used if not specified. |

| Option | Description |
|---|---|
| `-rmiregp PORTNUMBER` | Port number on which the Analyzer listens for RMI connections. The default value is the value specified by the analyzer_port property in the `Analyzer.properties` file. This option only takes effect when communicating with an Analyzer that is already running; the Analyzer always uses the value specified in `Analyzer.properties` when starting up. |
| `-debug` | Start the Analyzer process in debug mode. |

Examples:

1. Start project PROJECT:

   ```
   ServicesManager –start –proj PROJECT
   ```

2. Stop Analyzer on host HOST:

   ```
   ServicesManager –stop –host HOST
   ```

## TVisionSetup

Location:

```
TVISION_HOME/bin/TVisionSetup.[sh|bat]
```

**Purpose:**

This should be run once after installation. This program helps users setup the LDAP, database, TransactionVision services and WebSphere application.

Different options are presented based on which options are installed and/or running. The following menu appears if WebSphere Application Server and the TransactionVision web user interface are installed, and the DEMO mode is **not** used:

```
1. Initialize database for TransactionVision
2. Verify database initialization for TransactionVision

3. Verify WebSphere MQ setup for TransactionVision

4. Initialize TransactionVision Analyzer
5. Verify TransactionVision Analyzer initialization
```

```
6. Initialize LDAP entries for TransactionVision
7. Verify LDAP initialization for TransactionVision

8. Install TransactionVision web application
9. Verify TransactionVision web application installation

99. Select all
V. Display version info for software

Type "quit" or "q" to exit
```

If demo mode is used, or the WebSphere application server or TransactionVision web user interface are not installed, certain items are omitted from the menu. If an item is omitted, the remaining items are renumbered in sequence.

- If you specified in **`TVisionSetupInfo`** to use DEMO mode for LDAP, items 6 and 7 are omitted from the menu.
- If the WebSphere application server or TransactionVision web user interface are not installed, items 6, 7, 8, and 9 are omitted from the menu.

**Syntax:**

```
TVisionSetup [-rmiregp PORTNUMBER][-debug][-h]
```

Options:

| Option | Description |
|---|---|
| `-rmiregp PORTNUMBER` | Port number on which the Analyzer listens for RMI calls. Default to 1099 if not specified. |
| `-debug` | Starts the Analyzer in debug mode. |
| `-h` | Displays a usage message. |

## TVisionSetupInfo

Location:

```
TVISION_HOME/bin/TVisionSetupInfo.[sh|bat]
```

**Purpose**:

Collects information from the administrator about external tools used by TransactionVision so that CLASSPATH and library path can be setup correctly before running TransactionVision. This utility performs the following:

- Saves installation path for software tools in TVISION_HOME/config/setup/DefaultInstallPath.xml.
- Modifies TVISION_HOME/config/datamgr/Database.properties based on user input.
- Modifies TVISION_HOME/config/ldap/Ldap.properties based on user input.
- If you are using the WebSphere application server, modifies TVISION_HOME/config/websphere/tvision-ws-config.properties based on user input.
- If you are using the WebLogic application server, modifies TVISION_HOME/config/websphere/tvision-wl-config.properties based on user input.
- Generates TVISION_HOME/bin/SetupEnv.[sh|bat] which sets the minimum set of environment variables (JAVA_HOME, CLASSPATH, shared library path for different platforms, etc.) required by TVision.
- If you are using the WebSphere application server, generates TVISION_HOME/install/websphere/WSSetupEnv.[sh|bat] to setup command line parameters for importing different .xml files to XMLConfig.

**Syntax**:

```
TVisionSetupInfo.[sh|bat]
```

Notes:

`TVISION_HOME/config/setup/Setup.properties` is used to specify the XML file that stores the default software installation path and setup logging file.

User is free to modify any modified/generated files later on.

Since this utility modifies TransactionVision's configuration files, the user needs to have file modification privilege to run it (root on Unix or Administrator on Windows).

The format of `TVISION_HOME/config/setup/DefaultInstallPath.xml` is as follows:

```
<?xml version="1.0"?>
<DefaultInstallPath>
  <OS name="Windows">
    <DB2>C:\Program Files\IBM\sqllib</DB2>
    <WebSphereMQ>C:\Program Files\IBM\WebSphere
MQ</WebSphereMQ>
    <WebSphereMQJava>C:\Program Files\IBM\WebSphere
MQ\Java</WebSphereMQJava>
  <WebSphereAppServer>C:\Program Files\Websphere\
AppServer</WebSphereAppServer>
  <WebLogic>C:\bea\weblogic81</WebLogic>
  </OS>
  <OS name="SunOS">
    <DB2>/opt/IBMdb2/V7.1</DB2>
    <Oracle></Oracle>
    <WebSphereMQ>/opt/mqm</WebSphereMQ>
    <WebSphereMQJava>/opt/mqm/java</WebSphereMQJava>
    <WebSphereAppServer>/opt/WebSphere/AppServer
</WebSphereAppServer>
<WebLogic>/usr/local/bea/weblogic81</WebLogic>
  </OS>
  <OS name="AIX">
    <DB2>/usr/lpp/db2_07_01</DB2>
    <Oracle></Oracle>
    <WebSphereMQ>/usr/lpp/mqm</WebSphereMQ>
    <WebSphereMQJava>/usr/lpp/mqm/java</WebSphereMQJava>
    <WebSphereAppServer>/usr/WebSphere/AppServer
</WebSphereAppServer>
<WebLogic></WebLogic>
  </OS>
  <OS name="Linux">
    <DB2>/usr/IBMdb2/V7.1</DB2>
    <Oracle></Oracle>
    <WebSphereMQ>/opt/mqm</WebSphereMQ>
    <WebSphereMQJava>/opt/mqm/java</WebSphereMQJava>
    <WebSphereAppServer>/opt/WebSphere/AppServer
</WebSphereAppServer>
    <WebLogic></WebLogic>
  </OS>
</DefaultInstallPath>
```

## tvision-webservices-config

### Location

`$TVISION_HOME/bin/tvision-webservices-config.bat(sh)`

### Purpose

In order to capture WebServices related servlet events, run this utility after you run **`SensorSetup.bat(sh)`**. This script will instrument the `webservices.jar` file under the `WebSphere/lib` directory to allow TransactionVision to collect events from WebServices servlets.

# Appendix B
# WebSphere MQ Capacity Planning

This chapter provides a methodology for determining how TransactionVision will impact the number of messages handled by a given system. Then, these adjusted numbers can be used with the information in the *MQSeries Planning Guide* (CSQZAB03) to calculate storage needs, etc.

## Calculating TransactionVision's Impact on Message Volume

Use the following formula to determine the impact of TransactionVision monitoring on message volume:

*impact factor* = 1 + (*application efficiency factor* / *event packaging number*)

| Variable | Description |
|---|---|
| *impact factor* | The impact of TransactionVision monitoring on message volume. This number can be multiplied times the existing message volume to show how much message volume will be increased with TransactionVision monitoring. Note that when TransactionVision is not actively being used to monitor a system, there is no impact on message volume, so this factor becomes one. |

| Variable | Description |
|---|---|
| *application efficiency factor* | A number from 1 to 5 approximating how efficiently the applications use the WebSphere MQ APIs. It is the ratio of the number of WebSphere MQ APIs called to the number of messages handled. In the worst case scenario, each MQPUT or MQGET will be accompanied by an MQCONN, MQOPEN, MQCLOSE and MQDISC. In this case, the efficiency of the application would be 5, since approximately five APIs are called for every message processed. At the other end of the spectrum, if an application were to call MQCONN, MQOPEN and then MQPUT 1,000 messages and MQCLOSE, MQDISC, the total number of APIs is 1,004. Dividing this by the number of messages processed (1,000) gives us a ratio of 1.004. Estimating this ratio is necessary to determine how many events TransactionVision will generate, and hence how many event messages TransactionVision will send. |
| *event packaging number* | The parameter set in the data collection filter telling the Sensor to pack a certain number of events into one event message. Note that if event packaging is not enabled, then this factor has a value of 1. |

This impact factor may have to be calculated independently for different applications which are being monitored, since these applications may have different efficiency factors.

Let's consider an example. Suppose TransactionVision is being used to monitor an application running in batch mode overnight. This application runs on OS/390 and reads data from some internal datasets and pushes that data out using messages to a variety of other servers. On average the application sends out 2,500 messages spread evenly over 25 different queues. The application is written efficiently - there is only one MQCONN and MQDISC needed and then an MQOPEN/MQCLOSE pair for each queue. Each batch of 1,000 messages is also committed or rolled back using MQCMIT or MQBACK.

An efficiency rating for this application can be calculated as follows:

2,500 (msgs) + 2 (MQCONN/MQDISC) + 50 (MQOPEN/MQCLOSE) + 25 (MQCMIT) = 2577 apis

2,577 apis / 2,500 msgs = 1.03 *application efficiency factor*

Now, we can apply the *application efficiency factor* to the number of messages sent and calculate different *impact factors* based how we set the event packaging.

| Event Packaging Number | Impact Factor |
|---|---|
| 1 | `1+(1.03/1) = 2.03` |
| 10 | `1+(1.03/10) = 1.103` |
| 25 | `1+(1.03/25) = 1.0412` |

Since this is a very efficient application, the TransactionVision impact can be minimized quite well with a reasonable packaging number. Simply multiply the *impact factor* times the number of messages processed, 2,500 in this case, and we find that the total message traffic has increased to approximately 2,603 using the a packaging number of 25.

The *impact factor* can be used in the standard capacity planning formulae in the *MQSeries Capacity Planning Guide* which depend on either total messages or messages/second figures. Simply multiply the *impact factor* times the current messages or messages/second figure to get a new result which takes into account the increased traffic due to TransactionVision. Note that the use of this factor only accounts for increased message volume, but does not account for the message size.

## Message Size

Message size has an impact on the TransactionVision data. If messages are small, only several thousand bytes for example, then the relative size of API overhead (message descriptors, put message options, etc.) tracked by TransactionVision is large. If the messages themselves are large, however, the relative size of overhead is small, but all of the content of the messages is being duplicated. In cases where very large messages (greater than 500K, for example) are sent, it may be desirable to use data ranges in the data collection filter to have TransactionVision collect only a portion of the data. This minimizes strain on the storage requirements of the WebSphere MQ system.

## Data Collection Filters

The TransactionVision impact formula presented in this appendix considers the scenario where the broadest data collection filter is applied—every WebSphere MQ API is captured. The impact factor can be drastically reduced by defining more specific data collection criteria. For example, a simple data collection filter might capture only MQGET and MQPUT1 APIs. In this case, the *application efficiency factor* will be reduced to 1, since no extraneous APIs will be captured.

Since the data collection filter allows for great flexibility, it is difficult to create a formula to calculate the effect of a given data collection filter. Instead, the *impact factor* may be considered a “worst case" scenario which may be reduced through the use of appropriate data collection filters.

# Appendix C
# Configuration Files

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

*Important!* If you modify property files, you must restart the associated application for you changes to take effect. To restart the TransactionVision web user interface, use your application server administration console. To restart the Analyzer on Windows, use the Windows Services management control panel.

## Analyzer.properties

The `<TVISION_HOME>/config/services/Analyzer.properties` file provides general, collection, and analysis configuration information for the TransactionVision Analyzer.

### General Properties

- `rmi_client_timeout`
  This entry specifies the timeout for the RMI client in minutes. The default value is 2.
- `logging_xml`
  Specifies the name of the logging configuration file used by the Analyzer. The default value is `Analyzer.Logging.xml`.
- `trace`
  This property specifies whether Analyzer trace logging is on or off. The default value is off. For more information about trace logging, see “Trace Logging” on page 161.

- `service_jvm_flags`
  Specifies any additional JFM flags you want the Analyzer to run with.

- `service_additional_classpath`
  Specifies the classpaths for any custom beans to be used with your Analyzer.

- `jvm_dll`
  `service_classpath`
  `service_libpath`
  On Windows, these properties are automatically generated by **`TVisionSetupInfo`**. It should not be necessary to change them. If you do change them, note that they will be reset if **`TVisionSetupInfo`** is run again. These properties have no effect on the TransactionVision environment other than the Analyzer running as a Windows service.

- `analyzer_port`
  This entry specifies the default port number that the Analyzer runs on. The default value is 1099. The Analyzer always uses this port number when it starts. The **`SerciesManager.[bat|sh] -rmiregp`** command can be used to set the port number for an Analyzer that is already running.

## Collection Properties

The following properties are used for the event collection operations of the Analyzer:

- `batch_commit_count`
  Specifies the number of commits to batch. The default value is 1.

- `commit_time_threshold`
  Specifies the amount of time, in seconds, after which a commit is forced. The default value is 1.

- `commit_byte_threshold`
  Specifies the number of bytes after which a commit is forced. The default value is 1,000,000.

- `write_to_buffer_table`
  Specifies whether to write a copy of raw events from the queue into the RAW_EVENT table in addition to normal processing. The default value is False. This property should never be set to True if `read_from_buffer_table` is also set to True. Use this property to test custom beans. It should always be set to False in a production environment. Note that when you process the events from the RAW_EVENT table, the Analyzer will only process events if it has communication links.

- `read_from_buffer_table`
  Specifies whether to pull any events from the RAW_EVENT table in addition to reading from the queue. The default value is False. This property should never be set to True if `write_to_buffer_table` is also set to True. Use this property to test custom beans. It should always be set to False in a production environment.

- `raw_event_fetch_sleep`
  Specifies the number of milliseconds to sleep if no raw events are available in the RAW_EVENT table. The default value is 1000. This property is only used if `read_from_buffer_table` is True.

- `keep_events`
  Specifies whether the Analyzer should delete raw events after they have been successfully processed. Set it to **true** to keep the raw events in the RAW_EVENT table. Their event_status column will be set to PROCESSED so that they are not processed again. If you want to process the events again (for example, with a custom Java bean for a different type of analysis), you must first set the event_status to NEW. Set it to **false** (the default value) for the Analyzer to delete raw events from the RAW_EVENT table after it processes them.

- `fail_safe_collection_shutdown`
  If true, causes the Analyzer to stop sending configuration messages if a serious failure occurs in event collection and processing. The default value is true.

- `jdbc_batching | oracle_update_batching`
  The default `jdbc_batching` value (**on**) causes TransactionVision to execute database statements in batch mode. For Oracle environments, `oracle_update_batching` uses the Oracle JDBC driver's batch mode, which might improve performance on database insert/update. These properties are exclusive of each other; if both are set to on, the Analyzer automatically selects `oracle_update_batching` over `jdbc_batching`.

- `jdbc_batch_count`
  Specifies The number of SQL operations to batch in JDBC batching mode or Oracle update batching mode. Note that this number should be equal to or a multiple of the `batch_commit_count` property. The default value is 50.

### Analysis Properties

The following properties are used for the event analysis operations of the Analyzer:

- `minimal_event_document`
  Specifies whether to create a minimal event document. This setting only affects WebSphere MQ event data. The default is off.

- `save_event_document`
  Specifies whether to save event documents in the database. The default is on. If this property is set to off, event detail is not available. Furthermore, the Analyzer will not be able to recover asynchronously flushed static topology data. This means that in the event of a crash or other abnormal program termination, the statistics data for the static topology view may no longer be accurate.
- `eventmatching_interval`
  Specifies the time in seconds to wait between every invocation of event matching. The default is 3600.
- `eventmatching_interval`
  Specifies the time in seconds to wait between every invocation of event matching. The default is value is 600.
- `partial_event_lifetime`
  The maximum time in minutes a partial event entry may exist in the partial_event table. When this time limit is reached, the partial event will be flushed. The default value is 10 minutes.
- `latency_resolution`
  The resolution used to calculate latency between events. Possible values are:

| Value | Description |
|---|---|
| 1 | Seconds |
| 10 | 1/10 seconds |
| 100 | 1/100 seconds (default) |
| 1000 | Milliseconds |

- `separate_child_thread_txns`
  By default, if a servlet spins off a thread to make some JMS calls, the servlet passes tracking information to the child thread. The result is that both the servlet and JMS events belong to the same business transaction. However, there may be some cases in which you wish to separate these events into different transactions. For example, a servlet may spin off a long-running thread that you do not want to be part of the same transaction as the servlet. If you do **not** want threads spun off by a servlet to be included in the same business transaction as the servlet, list the servlet program name as the value for this property. Separate multiple program names with a comma, as in the following example:

  ```
  separate_child_thread_txns=program1, program2
  ```

- `dbcache_thread_count`
  Specifies the number of theads to use for writing the cached analysis data to the database. The best value is dependent on the hardware the Analyzer is running on and can only be determined by performance measuring with diffent values, but a good starting point would be to set this number to half the number of collection threads used for a particular schema. The default value is 2.

## CacheSize.properties

The `<TVISION_HOME>/config/services/CacheSize.properties` file specifies the size of the caches used in the event analysis. A bigger cache size minimizes database access and increases performance, but also increases the amount of memory used by TransactionVision. It sets the following properties:

- `system_model_objects`
  Specifies the number of system model object other than PII cached.
- `pii_objects`
  Specifies the number of PII system model objects cached.
- `event_based`
  Specifies the number of events cached.
- `transaction_based`
  Specifies the number of transactions cached.

## Database.properties

The `<TVISION_HOME>/config/datamgr/Database.properties` file specifies the database the Analyzer and the TransactionVision web application read from and write to.

### Required Entries

The following mandatory entries are required:

- `jdbc_driver=COM.ibm.db2.jdbc.app.DB2Driver`
  This entry is the class name of the JDBC driver used. The default for the IBM DB2 JDBC driver is as above. For Oracle, the default is `oracle.jdbc.driver.OracleDriver`.
- `database_connection_name`
  This entry is the name of the database connection to be used.

- `database_name`
  This entry is the name of the database on the server to connect to. This name may be different from the "database_connection_name" if a client database connection is used.
- `database_host`
  This entry is the host the database server is running on.
- `database_port`
  Specifies the port number for the Oracle listener on the target host. The default value is 1521. This attribute is only used if you are using the Oracle thin client. For more information about using Oracle with TransactionVision, see “Connecting to an Oracle Database” on page 60.
- `oracle_client_type`
  Specifies whether to use the Oracle thin or client JDBC driver. Set it to `thin` (recommended) or `oci` for Oracle 9 or `oci8` for Oracle 8.1.7. The default is `thin`. This attribute is only used if you are using an Oracle database. For more information about using Oracle with TransactionVision, see “Connecting to an Oracle Database” on page 60.
- `db2_instance_env`
  This entry specifies the value of the DB2 environment variable DB2INSTANCE.

### Optional Entries that Override Automatic Detection

The following optional entries may be setup in this file. These entries override automatic detection.

- `connection_type`
  This entry defines how to obtain JDBC connections. Use one of the following values:
  - JDBC: Uses connections obtained from the JDBC driver, with no connection pooling. To use a JDBC connection, you must also set the `database_url` property.
  - DB2DataSource: Uses DB2 connection pooling.
  - OracleDataSource: Uses Oracle connection pooling.
  - JNDI: Uses the data source registered in JNDI. To use a JNDI connection, you must also set the `jndi_url` property.
- `database_url`
  This entry defines the URL to use with the JDBC driver manager.
- `jndi_url`
  This entry defines the JNDI name for the database (defined in WebSphere) when using JNDI connections.

- `use_id_generator`
  This entry specifies whether the TVISION DataManager should generate ID values or not. It is used with DB2 6.1. A value of **true** indicates that IDs are generated by the TVISION DataManager; **false** indicates that IDs are generated by the database (using database SEQUENCEs).

## Optional Entries with Default Values

The following optional entries may be setup in this file. If they are not specified, default values are used.

- `user`
  This optional entry is the user name to be used while making the database connection. If this field is empty, the currently logged in user is used to make the database connection. Make sure that the user specified or currently logged has privileges for database access.

- `passwd`
  This optional entry is the password to be used while making the database connection. If this field is empty, the currently logged in user's password is used to make the database connection.

- `jce_provider`
  This optional entry specifies the Java Cryptographic Extension (JCE) package name to encrypt the database password. The following table shows the valid package names. If this entry is blank, TransactionVision stores the password as plain text.

| Provider | Package Name |
|---|---|
| Sun | `com.sun.crypto.provider.SunJCE` |
| IBM | `com.ibm.crypto.provider.IBMJCE` |

The **`TVision_SetupInfo`** utility automatically searches for a an installed JCE provider that supports the DES encryption algorithm. If such a provider is found, **`TVision_SetupInfo`** sets the `jce_provider` value to the class name of the JCE provider. If no JCE provider is found, **`TVision_SetupInfo`** displays the following message:

```
Java Cryptography Extension (JCE) is not present in
your current JDK. Password encryption feature will be
disabled and stored in plain text.
```

- `reconnect_interval`
  This entry specifies the frequency in seconds that TransactionVision should try to reconnect with the database. The default value is 10.

- `reconnect_timeout`
  This entry specifies the amount of time in seconds that TransactionVision should try to reconnect to the database (at the `reconnect_interval`). The default value is 600.

## JobManager.properties

The `<TVISION_HOME>/config/jobs/JobManager.properties` file specifies configuration information for TransactionVision jobs. This file specifies the following properties:

- `BeansXMLFile`
  Specifies the location of the configuration file listing job templates available when you create a new project.

- `logSize`
  Specifies the number of entries to keep in the log file.

## Ldap.properties

The `<TVISION_HOME>/config/ldap/Ldap.properties` file specifies how the web user interface interacts with an LDAP server. This file specifies the following properties:

- `LDAPServerName`
- `ContextFactory`
- `UserDirectoryLocation`
- `GroupDirectoryLocation`
- `UseSSL`
- `CertificateTrustStore`
- `CryptProvider`
- `UseUID`
- `RDNPrefix`

### LDAPServerName

This property points to the server to connect to. It directly maps to the attribute of `java.naming.provider.url`, used in creating an initial JNDI context. The value of this property should contain a URL string (for example, `ldap://somehost:389`).

### ContextFactory

This property contains the Java class name to be created as the initial context factory, depending on which implementation you wish to use. The value of the property should be the fully qualified class name of the factory class that will create an initial context. If you are using WebSphere 4.0, set it to `com.sun.jndi.LdapCtxFactory`. For WebSphere 3.5.4, set it to `com.ibm.jndi.LDAPCtxFactory`.

### UserDirectoryLocation

This property specifies to the web application what path to use to look up users for authentication and retrieval of TransactionVision authorities. If you wish to use an existing user hierarchy, use this attribute to integrate with the existing directory structure. When a user logs in, the user name is combined with the `UserDirectoryLocation` to perform authentication. For example, if user `joesmith` logs into the web application and the value of this property is `ou=Users,o=MyOrg,c=us` TransactionVision attempts to authenticate `cn=joesmith,ou=Users,o=MyOrg,c=us`.

TransactionVision creates by default a user location under `ou=Users,ou+TVision,o=bristol,c=us` if you do not wish to use an existing user directory. In order to have TransactionVision permissions, users in this directory must have a `TVAuthorityGroup` attribute. See the *TransactionVision Administrator's Guide* for more information.

### GroupDirectoryLocation

This property, similar to `UserDirectoryLocation`, specifies where to look up user groups. A group is considered to be a definition with one or more unique member attributes (such as the `groupofuniquenames` class) containing the list of users, and a `TVisionAuthorityGroup` attribute indicating what permissions to give this group.

To use the `GroupDireectoryLocation` in TransactionVision, you must configure the additional items in your LDAP schema and directory tree.

The way that TransactionVision expects a group to be laid out is a directory object with a number of `uniqueMember` values corresponding to each user in this group, such as the `groupOfUniqueNames` object class.

In addition to the group members, TransactionVision looks for an attribute called `TVAuthorityGroup` in this object. `TVAuthorityGroup` points to the TransactionVision security definition to be assigned to all members of this group. To accomplish this task in your LDAP administration interface, you need to define a new schema object. This object should be derived from `groupOfUniqueNames`, or a similar class that contains a `uniqueMember` attribute. To this new object class, add the attribute `TVAuthorityGroup`. This attribute is added to your LDAP schema by the LDAP Initialization steps of **`TVisionSetup`**, which you need to have already run.

After you have successfully created this schema object, create an instance of it in your directory tree, and assign to it the users that should be a member of it. The following ldif definition gives an example of creating such an entry.

```
dn: cn=AdminGroup,ou=TVision,o=bti,c=us
objectclass: TVgroup
objectclass: groupOfUniqueNames
objectclass: top
cn: AdminGroup
TVAuthorityGroup: cn=Admin,ou=Security,ou=TVision,o=bti,c=us
uniqueMember: cn=Joe Smith,ou=Users,ou=TVision,o=bti,c=us
```

The following table describes the contents of the entry:

| Value | Description |
|---|---|
| `dn` | The name and location of the newly created group. In this example, an object named cn=AdminGroup, located in ou=TVision,o=bti,c=us. The ou=TVision,o=bti,c=us would also be the value to assign the `GroupDirectoryLocation` property as that is where TransactionVision will find the group. |

| Value | Description |
|---|---|
| `objectclass` | The object class defined to add the `TVAuthority-Group` attribute, `TVgroup` in this example. Change `TVgroup` to the name you used to create the schema class. |
| `cn` | The group name for this user. |
| `TVAuthorityGroup` | The location of the privileges to be given this user—in this example `cn=Admin,ou=Security, ou=TVision,o=bti,c=us`. |
| `uniqueMember` | The location of a member of this group. Note that `uniqueMember` should contain reference to an already existing directory entry, as the directory server will validate it and possibly generate an error if it doesn't exist. |

### UseSSL

This property is used in conjunction with CertificateTrustStore and CryptProvider to enable communication with the LDAP server using SSL. If UseSSL is true, SSL will be enabled on the directory connection by setting Context.SECURITY_PROTOCOL to “ssl”.

*Important!* To use SLL, JSSE (or another third-party SSL package) extensions must be installed. JSSE is available at http://java.sun.com. For more information about setting up SSL, see the *TransactionVision Administrator's Guide*.

### CertificateTrustStore

Set this property to the key store (created by **`keytool`**) containing the trusted certificate for the LDAP server. The system property javax.net.ssl.trustStore is set to this value.

### CryptProvider

Set this property to the Cryptographic Service Provider you wish to register as set by java.security.Security.addProvider(). If you are using the standard JSSE package, leave this property set to the default value com.sun.net.ssl.internal.ssl.Provider.

### UseUID

If this property is set to true, TransactionVision assumes that the user name entered on the login page is the user id (UID) attribute of a user entry in the directory. It attempts to locate and authenticate the user by searching for user entries with the matching UID attribute. If this property is not defined or is set to false, TransactionVision assumes the user name entered on the login page is the RDN part of the distinguished name of the user entry in the directory.

### RDNPrefix

If this property is specified, TransactionVision uses the prefix value as the name part of the *name=value* pair when it generates the RDN part of the user entry for lookup in the LDAP directory. For example, if the RDNPrefix is set to *sn* and the user name is *joe*, then the RDN for this user is *sn=joe*. This will be concatenated with the subdirectory to form a complete distinguished name for directory entry lookup and authentication.

If this property is not set, TransactionVision assumes the prefix value is *cn*.

### Example

The following is an example `Ldap.properties` file:

```
LDAPServerName=ldap://myhost
ContextFactory=com.sun.jndi.ldap.LdapCtxFactory
IBMContextFactory=com.ibm.jndi.LDAPCtxFactory
UserDirectoryLocation=ou=Users,ou=TVision,o=bristol,c=us
GroupDirectoryLocation=ou=Groups,ou=TVision,o=bristol,c=us
```

## License.properties

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

## Performance.properties

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

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

## Sensor.properties

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

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

## SensorConfiguration.xml

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

| Attribute | Description |
|---|---|
| ConfigurationQM | The name of the configuration queue manager. |
| ConfigurationQ | The name of the configuration queue. |
| ConfigurationQM-Host | The host name of the configuration queue manager. |
| ConfigurationQM-Port | The listener port number of the configuration queue manager. |
| ConfigurationQM-Channel | The channel name of the configuration queue manager. |
| ConnectionRetry-Delay | The delay in milliseconds between connection attempts when the connection to the configuration queue manager is lost. The default is 1000. |
| ConnectionRetry-Timeout | The amount of time in milliseconds to try to reconnect to the configuration queue manager when the connection is lost. A value of -1 (the default value) is to retry forever. |
| ConfigurationRetrieveInterval | The time interval in milliseconds to check the configuration queue for new configuration messages. The default value is 10000. |

| Attribute | Description |
|---|---|
| SensorClient-TimeSkewInterval | The time interval in milliseconds to check the time skew from the host running the Sensor and the host running the configuration queue manager. The default value is 300000 (5 minutes). |
| EventPackageFlush-Timeout | The amount of time in milliseconds that an event package remains idle (no events added) before it is forced to be written to the event queue. The minimum value is 10000; the default is 300000 (5 minutes). |
| RepeatLogInterval | The amount of time in milliseconds to repeat a repetitive error that would normally be suppressed. The default value is 600000. |

## Setup.properties

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

- `default_tool_install_path`
  The directory location of the DefaultInstallPath.xml file, which lists the locations of software components required by TransactionVision
- `logs_dir`
  The name of the directory in which to store log files
- `logging_xml`
  The name of the logging configuration file
- `minimum_java_version`
  The minimum Java version required by TransactionVision
- `minimum_java_version_sun`
  The minimum Java version required by TransactionVision on the Solaris platform
- `maximum_java_version`
  The highest Java version supported by TransactionVision

## StatisticsCache.properties

The `<TVISION_HOME>/config/services/StatisticsCache.properties` file defines setting used in the static mode of the component topology analysis. It contains the following entries:

- `flush_interval`
  Specifies how often the statistics cache is written out to the database. Until the data is written to the database, it won't be visible in graph. Depending on the volume of incoming requests, it might be useful to either flush more often, or less often. A longer flush time means fewer database writes, but will result in a larger cache size.
- `timeslice_interval`
  Specifies the duration in minutes of the timeslice for which event statistics should be calculated. The concept of a timeslice is used to partition data into different pieces so that depending on what timeframe one is interested in looking at this data can be quickly retrieved. A timeslice represents a period of time during which all events belonging to the same program and MQ object are condensed into a single statistic. For example, if you had a program putting messages to a particular queue, all the statistics data relating to the program and the queue (latency times, success counts, etc.) during a particular timeslice are stored in the same database row. By changing the values of the time slice interval you can control how big and how efficient access to the statistics table is. The larger the timeslice, the more efficient the storage of results will be; the trade-off you loose is that your time slice interval is the smallest increment of time that you can view your data in. For example, if you were to set a time slice of one day, you would not be able to view these statistics on an hourly basis.

## tvision-wl-config.properties

The `<TVISION_HOME>/config/weblogic/tvision-wl-config.properties` file specifies WebLogic application server information for the TransactionVision web user interface. This file specifies the following properties:

- `adminserver_name`
  Specifies the administration server name.
- `adminserver_host`
  Specifies the administration server host name.
- `adminserver_port`
  Specifies the administration server communication port number.
- `admin_user_name`
  Specifies the administration user name.
- `admin_user_password`
  Specifies the administration user password.
- `domain_dir`
  Specifies the domain directory.

- `server_name`
  Specifies the name of the server where TransactionVision will be deployed.

## tvision-wl-sensorconfig.properties

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

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

## tvision-ws-config.properties

The `<TVISION_HOME>/config/websphere/tvision-ws-config.properties` file specifies WebSphere application server information for the TransactionVision web user interface. This file specifies the following properties:

- `appserver_node_name`
  Specifies the application server node name.
- `appserver_name`
  Specifies the application server name.
- `virtual_host_name`
  Specifies the virtual host name.
- `cell_name`
  For WebSphere Application Server 5.x, specifies the cell name.

- `root_URL`
  For WebSphere Application Server 4.x, specifies the root URL for the TransactionVision web user interface.
- `adminserver_host`
  For WebSphere Application Server 4.x, specifies the administrative server host name.
- `adminserver_port`
  For WebSphere Application Server 4.x, specifies the administrative server port number.

## tvision-ws-sensorconfig.properties

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

- `appserver_names_v5`
  For WebSphere Application Server 5.x, specifies the cell name, node name, and server name. Separate multiple entries with a comma.
- `appserver_names_v4`
  For WebSphere Application Server 4.x, specifies the node name and server name. Separate multiple entries with a comma.
- `adminserver_host`
  For WebSphere Application Server 4.x, specifies the administrative server name.
- `adminserver_port`
  For WebSphere Application Server 4.x, specifies the administrative server port number.

## UI.properties

The `<TVISION_HOME>/config/ui/UI.properties` file specifies properties used by the TransactionVision web user interface. It contains the following entries:

- `ui_config_dir`
  Specifies the location of the user interface configuration files.
- `view_config_filename`
  Specifies the file defining TransactionVision views.
- `reports_xml_filename`
  Specifies the location of the file listing available reports.
- `category_xml_filename`
  Specifies the location of the query configuration file.

- `category_xsl_filename`
  Specifies the location of the query style sheet.

- `filter_config_xml`
  Specifies the location of the data collection filter configuration file.

- `filter_config_xsl`
  Specifies the location of the data collection filter style sheet.

- `detail_modifier_xsl`
  Specifies the location of the Event Detail view style sheet.

- `logging_xml`
  Specifies the location of the user interface log file.

- `support_url`
  The URL for product support.

- `autologin`
  Enables the “Remember Login” checkbox on the TransactionVision login page. Setting the checkbox causes the user name and password to be saved as a cookie. so that users are not prompted for a username and password the next time they try to access TransactionVision. The password is encrypted using the specified Java JCE provider.

- `jceProvider`
  Specifies the name of the JCE package used to encrypt the password if `autologin` is true. If you don’t have a JCE package, you can download one from `java.sun.com`.

- `trace`
  Enables trace logging for the TransactionVision web user interface. For more information, see “Trace Logging” on page 161.

- `logout_Redirect`
  This property specifies the URL that should be displayed when a user logs out from TransactionVision, such as an SSO logout page.

- `dashboard_xml_filename`
  The dashboard report configuration file name.

- `hasProxySensor`
  Set to True to enable the dynamic Component Topology Analysis view to show proxy related links if you are using the Proxy Sensor.

Appendix D

# TransactionVision Changes to WebSphere Application Server

A number of changes are made to WebSphere Application Server during TransactionVision installation and configuration.

## JVM System Property Settings

The following properties are added to the server's JVM system property settings.

- `com.bristol.tvision.home`
  Specifies the TransactionVision installation directory. This is required by the TransactionVision web user interface and Sensors.

- `com.ibm.websphere.classloader.plugin`
  Loads the TransactionVision servlet Sensor. It should be set to `com.bristol.tvision.sensor.servlet.ClassLoaderPlugin`.

- `com.bristol.tvision.sensor.disableApps`
  Disables the servlet Sensor for any application name listed. Delimit multiple applications with a comma, semi-colon, or colon. The default value is `TransactionVision,adminconsole`.

## JDBC Driver Resource

If it does not already exist, a resource is added to the server for the JDBC driver. This resource is required by the TransactionVision web user interface on WebSphere Application Server 5. The resource depends on the type of database you are using:

### Oracle Database

```
resource name=Oracle JDBC Driver
implementationClass=oracle.jdbc.pool.OracleConnectionPoolDataSource
classpath=${ORACLE_JDBC_DRIVER_PATH}/classes12.zip
```

### DB2 Database

```
resource name=DB2 JDBC Driver
implementationClass=COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
classpath=${DB2_JDBC_DRIVER_PATH}/db2java.zip
```

## JDBC Driver Path Variable

If it does not already exist, a variable is added to the server for the JDBC driver path. This resource is required by the TransactionVision web user interface on WebSphere Application Server 5. The resource depends on the type of database you are using:

### Oracle Database

```
symbolicName=ORACLE_JDBC_DRIVER_PATH
value=<oracle_install_dir>/jdbc/lib
```

### DB2 Database

```
symbolicName=DB2_JDBC_DRIVER_PATH
value=<oracle_install_dir>/java
```

## JVM Classpath Updates

The following are added to the application server's JVM classpath. They are required by the TransactionVision Sensor.

- `<tv_install_dir>/java/lib/tvisionsensorservlet.jar`
- `<tv_install_dir>/java/lib/tvisionsensorejb.jar`
- `<tv_install_dir>/java/lib/tvjavaext.jar`
- `<tv_install_dir>/java/lib/tvisionsensorjms.jar` (if you want to monitor JMS calls)
- `<tv_install_dir>/java/lib/com.ibm.mqjms.jar` (if you want to monitor JMS calls)
- `<was_install_dir>/lib/j2ee.jar`
- `<webspheremq_java_dir>/lib`
- `<webspheremq_java_dir>/lib/com.ibm.mqjms.jar`
- `<webspheremq_java_dir>/lib/com.ibm.mq.jar`

- `<webspheremq_java_dir>/lib/com.ibm.mqbind.jar`
- `<webspheremq_java_dir>/lib/jta.jar`
- `<webspheremq_java_dir>/lib/connector.jar`
- `<webspheremq_java_dir>/lib/providerutil.jar`
- `<webspheremq_java_dir>/lib/fxcontext.jar`

## Added Files

The following files are added to `<was_install_dir>` classes. They are required by the TransactionVision Sensor.

- `tvisionsensorcheck.jar`
- `tvisionservletinstrument.jar`

Appendix E

# Configuring Application Servers for TransactionVision

Typically, you can use the TransactionVision **`TVisionSetupInfo`**, **`TVisionSetup`**, **`SensorSetup`**, and **`SensorAdmin`** utilities to configure your application server for the TransactionVision web user interface and Sensors. However, there may be situations where you prefer or are required to configure your application server manually.

The following sections describe the process required to configure the TransactionVision web user interface and Sensors for IBM WebSphere and BEA WebLogic.

## Configuring WebSphere for TransactionVision

The following tasks are performed by **`TVisionSetup`**. However, if you prefer, you may perform these steps from the WebSphere Admin console.

- Add com.bristol.tvision.home as JVM System property setting
- Add the DB2 or Oracle JDBC driver to a server configuration
- Install and setup the TransactionVision web user interface
- Add the TransactionVision Sensor

### Add com.bristol.tvision.home as JVM System Property Setting

1. Select Servers > Application Servers > server1.
2. Select Process Definition > Java Virtual Machine.
3. Select Custom Properties.

4. Click New and add the following:

   Name: com.bristol.tvision.home
   Value:<TVISION_HOME>

   For example:

   com.bristol.tvision.home = C:\Program Files\Bristol\TransactionVision

### Add the DB2 or Oracle JDBC Driver to a Server Configuration

Perform this task from the WebSphere Admin Console:

1. Select Resources > JDBC Providers.
2. Select the scope you want to use: cell, node, server
3. Click New and select DB2 Legacy CLI-based Type 2 JDBC Driver or Oracle JDBC Driver (depending on what type of database you are using) in the pull down list for field JDBC Providers. Then click OK.
4. Select Environment > Manage WebSphere Variables.
5. Select the scope you want to use: cell, node, server
6. If you are using DB2, find/add the variable "DB2_JDBC_DRIVER_PATH". If it is not already pointed to the right place, set it to the directory where your db2java.zip is located. If you are using Oracle, find/add the variable "ORACLE_JDBC_DRIVER_PATH". If it is not already pointed to the right place, set it to the directory where your class12.zip is located.
7. Save the changes.
8. Restart the server for settings to take effect.

### Install and Set up the TransactionVision Web User Interface

Perform the following steps from the WebSphere Admin Console:

1. Select Applications > Install New Application.
2. Click Browse in section Path to select your `tvision.ear` file. Click Next.
3. Make your selection in section Virtual Host then click Next.
4. Follow instructions on following pages to finish the installation.
5. Save the changes to Master configuration.
6. Select Servers > Application Servers.
7. Select the server where you have installed TransactionVision web application.
8. Select Process Definition in the list of Additional Properties.

9. Select Java Virtual Machine in the list of Additional Properties.
10. Select Custom Properties in the list of Additional Properties.
11. Add following Name/Value pair:

    ```
    com.bristol.tvision.home = <tv_install_dir>
    ```

12. Save the changes to Master configuration.
13. Restart the server for the settings to take effect.

### Add the TransactionVision Sensor (WebSphere Application Server 5.1)

*Important!* Before performing this task, run the **<TVISION_HOME>/bin/SensorAdmin** script to instrument certain application server and JMS JAR files. Then run **<TVISION_HOME>/bin/SensorSetup**to configure the communication links used by the Sensor.

Perform this task from the WebSphere Admin Console:

1. Select Servers -> Application Servers.
2. Select the server you are monitoring.
3. Select Process Definition in the list of Additional Properties.
4. Select Java Virtual Machine in the list of Additional Properties.
5. Add following paths to field Classpath in General Properties section:
    - `<tv_install_dir>/java/lib/tvisionsensorservlet.jar`
    - `<tv_install_dir>/java/lib/tvisionsensorejb.jar`
    - `<tv_install_dir>/java/lib/tvisionsensorjms.jar (if you want to monitor JMS calls)`
    - `<tv_install_dir>/java/lib/com.ibm.mqjms.jar (if you want to monitor JMS calls)`
    - `<was_install_dir>/lib/j2ee.jar`
    - `<webspheremq_java_dir>/lib`
    - `<webspheremq_java_dir>/lib/com.ibm.mqjms.jar`
    - `<webspheremq_java_dir>/lib/com.ibm.mq.jar`
    - `<webspheremq_java_dir>/lib/com.ibm.mqbind.jar`
    - `<webspheremq_java_dir>/lib/jta.jar`
    - `<webspheremq_java_dir>/lib/connector.jar`

- `<webspheremq_java_dir>/lib/providerutil.jar`
- `<webspheremq_java_dir>/lib/fxcontext.jar`

6. Select Custom Properties in the list of Addition Properties.
7. Add following Name/Value pairs:
   - `com.bristol.tvision.home = <tv_install_dir>`
   - `com.ibm.websphere.classloader.plugin = com.bristol.tvision.sensor.servlet.ClassLoaderPlugin`
   - `com.bristol.tvision.sensor.disableApps = TransactionVision,adminconsole`
8. Save the changes to Master configuration.
9. Add the following lines to your `server.xml` file:

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

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

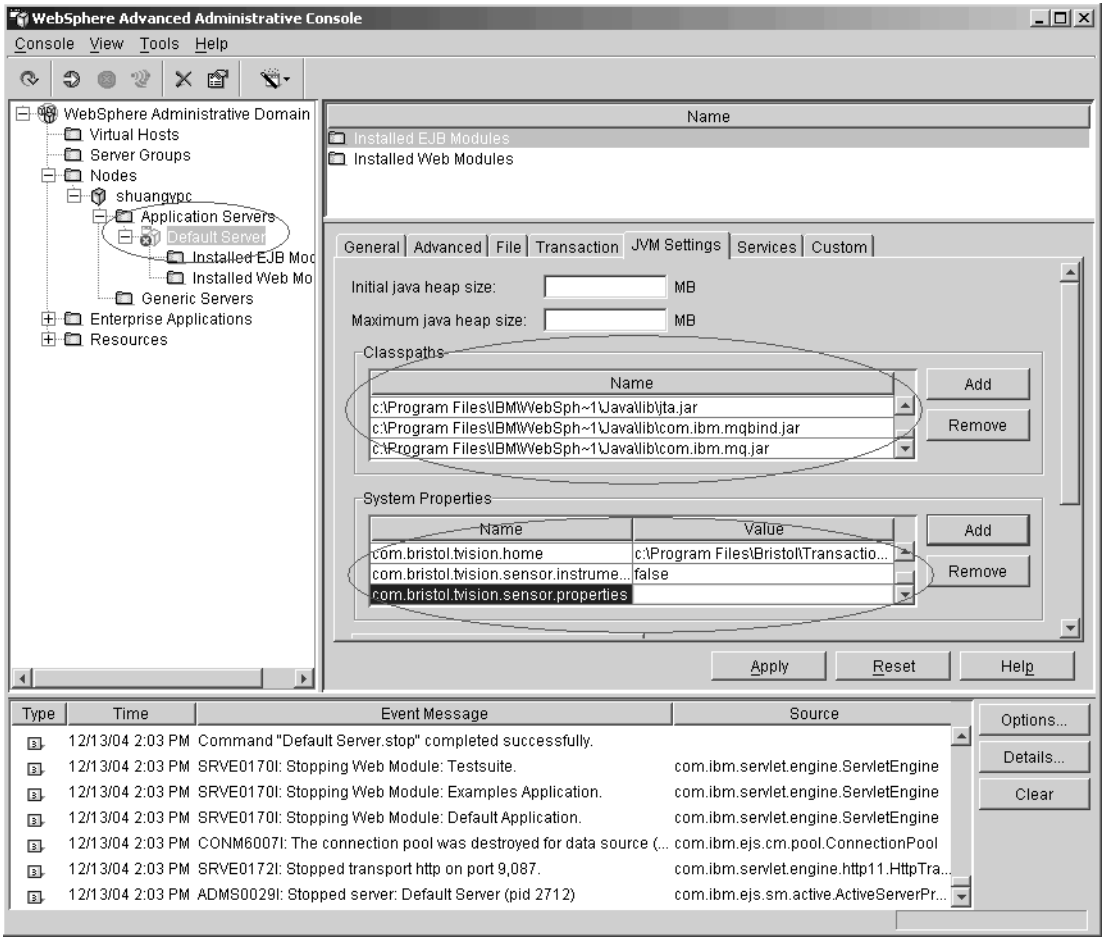
   In “Interceptor X,” the X is a unique number one higher than the existing Interceptor values in your `server.xml` file. For example, Interceptor_18, and Interceptor_19 could be the two names you give if 18 and 19 are not currently in use.
10. Restart the server for settings to take effect.

### Add the TransactionVision Sensor (WebSphere Application Server 4.0.7)

*Important!* Before performing this task, run the **`<TVISION_HOME>/bin/SensorAdmin`** script to instrument certain application server and JMS JAR files. Then run **`<TVISION_HOME>/bin/SensorSetup`**to configure the communication links used by the Sensor.

Perform this task after installing the Sensor:

1. Run **`adminclient.bat`** | **`sh`** to open the WebSphere Administrative Console:




2. In the top left pane, select the application server where you are going to install the Sensor. The right-hand pane will show properties for that application server.
3. On the JVM Settings tab, add the following jar files to Classpaths, replacing $TVISION_HOME, $WAS_HOME, and $MQJAVA_HOME with the local settings:
   - `$TVISION_HOME/java/lib/tvisionsensorservlet.jar`
   - `$TVISION_HOME/java/lib/tvisionsensorjms.jar`
   - `$TVISION_HOME/java/lib/com.ibm.mqjms.jar`
   - `$WAS_HOME/lib/j2ee.jar`

- `$WAS_HOME/lib/xerces.jar`
- `$MQJAVA_HOME/lib/com.ibm.mqjms.jar`
- `$MQJAVA_HOME/lib/fscontext.jar`
- `$MQJAVA_HOME/lib/providerutil.jar`
- `$MQJAVA_HOME/lib/connector.jar`
- `$MQJAVA_HOME/lib/jta.jar`
- `$MQJAVA_HOME/lib/com.ibm.mqbind.jar`
- `$MQJAVA_HOME/lib/com.ibm.mq.jar`

4. Also on the JVM Settings tab, add the following name/value pairs to the System Properties:

| Name | Value |
|---|---|
| com.ibm.websphere.classloader.-plugin | com.bristol.tvision.sensor.serv-let.ClassLoaderPlugin |
| com.bristol.tvision.home | $TVISION_HOME |
| com.bristol.tvision.sensor.instru-ment.ejb | false |
| com.bristol.tvision.sensor.dis-ableApps | TransactionVision,adminconsole |

5. Apply and save your changes.
6. Restart the server for settings to take effect.

## Configuring WebLogic for TransactionVision

The following steps describe the process required to manually configure the TransactionVision web user interface and Sensors for the BEA WebLogic application server.

## Configure WebLogic for Sensors

*Important!* Before performing this task, run the **<TVISION_HOME>/bin/SensorAdmin** script to instrument certain application server and JMS JAR files. Then run **<TVISION_HOME>/bin/SensorSetup**to configure the communication links used by the Sensor.

### Application Server

To monitor a WebLogic Admin Server, copy **startWebLogic.[cmd|sh]** to **tvStartWebLogic.[cmd|sh]** and modify the copied file to add the following environment settings. See "Modify startWebLogic and startManagedWebLogic Scripts" on page 229 for instructions on modifying these files.

**Windows:**

```
set JAVA_OPTIONS= -Dcom.bristol.tvision.home="<tv_install_dir>"
-Dweblogic.classloader.preprocessor="com.bristol.tvision.sensor.servlet.
WebLogicClassPreProcessor"
-Dcom.bristol.tvision.sensor.disableApps="TransactionVision,console,wl_m
anagement_internal1,wl_management_internal2,uddi,uddiexplorer"
%JAVA_OPTIONS%


set CLASSPATH=<tv_install_dir>\java\lib\tvisionsensorservlet.jar;
<tv_install_dir>\java\lib\weblogic.jar;
<tv_install_dir>\java\lib\webservices.jar;
<tv_install_dir>\java\lib\tvisionsensorjms.jar;
<tv_install_dir>\java\lib\com.ibm.mqjms.jar;
<tv_install_dir>\java\lib\xercesImpl.jar;
<tv_install_dir>\java\lib\xmlParserAPIs.jar;
<mqjava_install_dir>\lib;<mqjava_install_dir>\lib\com.ibm.mqjms.jar;
<mqjava_install_dir>\lib\fscontext.jar;
<mqjava_install_dir>\lib\providerutil.jar;
<mqjava_install_dir>\lib\connector.jar;<mqjava_install_dir>\lib\jta.jar;
<mqjava_install_dir>\lib\com.ibm.mqbind.jar;
<mqjava_install_dir>\lib\com.ibm.mq.jar;
<%WL_HOME%>\server\lib\webservices.jar;%CLASSPATH%
```

**UNIX**

```
JAVA_OPTIONS=" -Dcom.bristol.tvision.home="<tv_install_dir>"
-Dweblogic.classloader.preprocessor="com.bristol.tvision.sensor.servlet.
WebLogicClassPreProcessor"
-Dcom.bristol.tvision.sensor.disableApps="TransactionVision,console,wl_m
anagement_internal1,wl_management_internal2,uddi,uddiexplorer"
$JAVA_OPTIONS"
```

```
CLASSPATH="<tv_install_dir>/java/lib/tvisionsensorservlet.jar:
<tv_install_dir>/java/lib/weblogic.jar:
<tv_install_dir>/java/lib/webservices.jar:
<tv_install_dir>/java/lib/tvisionsensorjms.jar:
<tv_install_dir>/java/lib/com.ibm.mqjms.jar:
<tv_install_dir>/java/lib/xercesImpl.jar:
<tv_install_dir>/java/lib/xmlParserAPIs.jar:
<mqjava_install_dir>/lib:<mqjava_install_dir>/lib/com.ibm.mqjms.jar:
<mqjava_install_dir>/lib/fscontext.jar:
<mqjava_install_dir>/lib/providerutil.jar:
<mqjava_install_dir>/lib/connector.jar:<mqjava_install_dir>/lib/jta.jar:
<mqjava_install_dir>/lib/com.ibm.mqbind.jar:
<mqjava_install_dir>/lib/com.ibm.mq.jar:
<$WL_HOME>/server/lib/webservices.jar:$CLASSPATH"
export CLASSPATH
```

*Important!* Add the classes at the beginning of CLASSPATH. For some domain types, the **`startWebLogic`** script calls another WebLogic script. In these cases, modify the called script as well, as described in "Modify startWebLogic and startManagedWebLogic Scripts" on page 229.

#### Managed Server

If you plan to start a managed server with the **`startManagedWebLogic[.cmd|.sh]`** script, copy **`startManagedWebLogic.[cmd|sh]`** to **`tvStartManagedWebLogic.[cmd|sh]`** and modify the copied file to add the same environment settings as for monitoring an Admin Server. See "Modify startWebLogic and startManagedWebLogic Scripts" on page 229 for instructions on modifying these files.

*Important!* Add the classes at the beginning of CLASSPATH. For some domain types, the **`startManagedWebLogic`** script calls another WebLogic script. In these cases, modify the called script as well, as described in "Modify startWebLogic and startManagedWebLogic Scripts" on page 229.

If you plan to start a managed server via the Node Manager using the Admin Console, perform the following steps:

1. Select your server name under Servers in the left panel.
2. Select the Configuration tab in the right panel.
3. Select the Remote Start tab in the right panel.
4. Add the following jar files to the Class Path field:

<tvision_install>\java\lib\tvisionsensorservlet.jar
<tvision_install>\java\lib\weblogic.jar
<tvision_install>\java\lib\webservices.jar
<tvision_install>\java\lib\tvisionsensorjms.jar (only needed for JMS Sensor)
<tvision_install>\java\lib\com.ibm.mqjms.jar (only needed for JMS Sensor)
<tvision_install>\java\lib\xercesImpl.jar
<tvision_install>\java\lib\xmlParserAPIs.jar
<mqjava_install>\lib
<mqjava_install>\lib\com.ibm.mqjms.jar
<mqjava_install>\lib\fscontext.jar
<mqjava_install>\lib\providerutil.jar
<mqjava_install>\lib\connector.jar
<mqjava_install>\lib\jta.jar
<mqjava_install>\lib\com.ibm.mqbind.jar
<mqjava_install>\lib\com.ibm.mq.jar
<weblogic_install>\server\lib\webservices.jar
<weblogic_install>\server\lib\weblogic.jar

5. Add the following Java options to the Arguments field:

   -Dcom.bristol.tvision.home="<tvision_install>"
   -Dweblogic.classloader.preprocessor="com.bristol.tvision.sensor.servlet.WebLogicClassPreProcessor"
   -Dcom.bristol.tvision.sensor.disableApps="TransactionVision,console, wl_management_internal1, wl_management_internal2,uddi,uddiexplorer"

6. Click Apply at the end of the page.

7. Restart the server.

## Configure WebLogic for the TransactionVision Web User Interface

To configure WebLogic for the TransactionVision web user interface, you must perform the following tasks:

- Add the required environment setup to the server.
- Deploy the `tvision.ear` file.

### Add Environment Setup to Admin Server

To deploy a WebLogic Admin Server, copy **`startWebLogic.[cmd|sh]`** to **`tvStartWebLogic.[cmd|sh]`** and modify the copied file to add the following environment settings. See “Modify startWebLogic and startManagedWebLogic Scripts” on page 229 for instructions on modifying these files.

**Windows:**

```
set JAVA_OPTIONS= -Dcom.bristol.tvision.home="<tv_install_dir>"
%JAVA_OPTIONS%
set CLASSPATH=<path_of_jar_file_for_JDBC_driver>;%CLASSPATH%
```

**UNIX:**

```
JAVA_OPTIONS=" -Dcom.bristol.tvision.home="<tv_install_dir>"
$JAVA_OPTIONS"
CLASSPATH="<path_of_jar_file_for_JDBC_driver>:$CLASSPATH"
export CLASSPATH
```

*Important!* Add the classes at the beginning of CLASSPATH. For some domain types, the **`startWebLogic`** script calls another WebLogic script. In these cases, modify the called script as well, as described in "Modify startWebLogic and startManagedWebLogic Scripts" on page 229.

### Add Environment Setup to Managed Server

If you plan to start a managed server with the **`startManagedWebLogic[.cmd|.sh]`** script, copy **`startManagedWebLogic.[cmd|sh]`** to **`tvStartManagedWebLogic.[cmd|sh]`** and modify the copied file to add the same environment settings as for monitoring an Admin Server.

*Important!* As with the Admin Server, these classes at the beginning of CLASSPATH. Any scripts called from startManageWebLogic must be updated as well.

If you plan to start a managed server via the Node Manager using the Admin Console, perform the following steps:

1. Select your server name under Servers in the left panel.
2. Select the Configuration tab in the right panel.
3. Select the Remote Start tab in the right panel.
4. Add the following jar files to the Class Path field:
   - Full path of the jar file for the JDBC driver class. For example: <db2_install>\java\db2java.zip
   - <weblogic_install>\server\lib\weblogic.jar
5. Add the following Java option to the Arguments field:

   -Dcom.bristol.tvision.home="<tvision_install>"
6. Click Apply at the end of the page.

7. Restart the server.

### Deploy tvision.ear File

To deploy the `tvision.ear` file with **WebLogic 8.1**, perform the following steps:

1. In the left panel, choose Deployments > Applications.
2. In the right panel, select the Deploy a new Application link.
3. Navigate to your `<tvision_install>\ui` directory and check the radio button next to "tvision.ear".
4. Click Target Application.
5. Check the server where you are deploying TransactionVision then click Continue.
6. Check one of the radio buttons in the "Source Accessibility" section.
7. Input "TransactionVision" in the Name: field of the Identify section.
8. Click Deploy. "TransactionVision" should show up under Deployments > Applications in the left panel.

## Modify startWebLogic and startManagedWebLogic Scripts

The following instructions are based on contents of the default **`startWebLogic`** and **`startManageWebLogic`** scripts created by the WebLogic Configuration Wizard. If you have customized these scripts, these instructions can only be used as reference.

The instructions depend on the domain type.

### WebLogic 8.1 on Windows

**Server Domain**

The **`startWebLogic.cmd`** calls the utility **`weblogic.Server`** at the end. Copy **`startWebLogic.cmd`** to **`tvStartWebLogic.cmd`** and add the required additional environment setup to the copied **`tvStartWebLogic.cmd`** before the following line:

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

The **startManagedWebLogic.cmd** calls the utility **weblogic.Server** at the end. Copy **startManagedWebLogic.cmd** to **tvStartManagedWebLogic.cmd** and add the required additional environment setup to the copied **tvStartManagedWebLogic.cmd** before the following line:

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

**Workshop Domain**

The **startWebLogic.cmd** calls the utility **weblogic.Server** at the end. Copy **startWebLogic.cmd** to **tvStartWebLogic.cmd** and add the required additional environment setup to the copied tvStartWebLogic.cmd before the following section:

```
if "%WLS_REDIRECT_LOG%"=="" (
     echo Starting WLS with line:
     echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%SERVER_CLASS%
     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
%SERVER_CLASS%
) else (
     echo Redirecting output from WLS window to %WLS_REDIRECT_LOG%
     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME% -
Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
%SERVER_CLASS%  1>&2 >"%WLS_REDIRECT_LOG%"
)
```

The **startManagedWebLogic.cmd** calls **startWebLogic.cmd** at the end. Copy **startManagedWebLogic.cmd** to **tvStartManagedWebLogic.cmd** and edit the copied **tvStartManagedWebLogic.cmd** to change the following line

```
call startWebLogic.cmd nodebug verbose nopointbase production
noiterativedev notestconsole %3 %4 %5 %6 %7 %8 %9
```

to:

```
call tvStartWebLogic.cmd nodebug verbose nopointbase production
noiterativedev notestconsole %3 %4 %5 %6 %7 %8 %9
```

**Portal Domain**

The **startWebLogic.cmd** calls the utility **weblogic.Server** at the end. Copy **startWebLogic.cmd** to **tvStartWebLogic.cmd** and add the required additional environment setup to the copied **tvStartWebLogic.cmd** before the following section:

```
if "%WLS_REDIRECT_LOG%"=="" (
     echo Starting WLS with line:
     echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%SERVER_CLASS%
     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
%SERVER_CLASS%
) else (
     echo Redirecting output from WLS window to %WLS_REDIRECT_LOG%
     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
%SERVER_CLASS%  1>&2 >"%WLS_REDIRECT_LOG%"
)
```

The **startManagedWebLogic.cmd** calls **startWebLogic.cmd** at the end. Copy **startManagedWebLogic.cmd** to **tvStartManagedWebLogic.cmd** and edit the copied **tvStartManagedWebLogic.cmd** to change the following line:

```
call startWebLogic.cmd nodebug nopointbase production
```

to:

```
call tvStartWebLogic.cmd nodebug nopointbase production
```

**Integration Domain**

The **startWebLogic.cmd** calls the utility **weblogic.Server** at the end. Copy **startWebLogic.cmd** to **tvStartWebLogic.cmd** and add the required additional environment setup to the copied **tvStartWebLogic.cmd** before the following section:

```
if "%WLS_REDIRECT_LOG%"=="" (
     echo Starting WLS with line:
     echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%SERVER_CLASS%
     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
%SERVER_CLASS%
) else (
     echo Redirecting output from WLS window to %WLS_REDIRECT_LOG%
     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
%SERVER_CLASS%  1>&2 >"%WLS_REDIRECT_LOG%"
)
```

The **startManagedWebLogic.cmd** calls **startWebLogic.cmd** at the end. Copy **startManagedWebLogic.cmd** to **tvStartManagedWebLogic.cmd** and edit the copied **tvStartManagedWebLogic.cmd** to change the following line:

```
call startWebLogic.cmd nodebug verbose nopointbase production
noiterativedev notestconsole %3 %4 %5 %6 %7 %8 %9
```

to:

```
call tvStartWebLogic.cmd nodebug verbose nopointbase production
noiterativedev notestconsole %3 %4 %5 %6 %7 %8 %9
```

**Platform Domain**

The **startWebLogic.cmd** calls the utility **weblogic.Server** at the end. Copy **startWebLogic.cmd** to **tvStartWebLogic.cmd** and add the required additional environment setup to the copied **tvStartWebLogic.cmd** before the following section:

```
if "%WLS_REDIRECT_LOG%"=="" (
     echo Starting WLS with line:
     echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%SERVER_CLASS%
     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
%SERVER_CLASS%
) else (
     echo Redirecting output from WLS window to %WLS_REDIRECT_LOG%
     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%WLS_PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"
%SERVER_CLASS%  1>&2 >"%WLS_REDIRECT_LOG%"
)
```

The **`startManagedWebLogic.cmd`** calls **`startWebLogic.cmd`** at the end. Copy **`startManagedWebLogic.cmd`** to **`tvStartManagedWebLogic.cmd`** and edit the copied **`tvStartManagedWebLogic.cmd`** to change the following line:

```
call startWebLogic.cmd nodebug verbose nopointbase production
noiterativedev notestconsole %3 %4 %5 %6 %7 %8 %9
```

to:

```
call tvStartWebLogic.cmd nodebug verbose nopointbase production
noiterativedev notestconsole %3 %4 %5 %6 %7 %8 %9
```

### WebLogic 8.1 on UNIX

#### Server Domain

The **`startWebLogic.sh`** calls the utility **`weblogic.Server`** at the end. Copy **`startWebLogic.sh`** to **`tvStartWebLogic.sh`** and add the required additional environment setup in the copied **`tvStartWebLogic.sh`** before the following line:

```
${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
weblogic.Server
```

The **startManagedWebLogic.sh** calls the utility **weblogic.Server** at the end. Copy **startManagedWebLogic.sh** to **tvStartManagedWebLogic.sh** and add the required additional environment setup in the copied **tvStartManagedWebLogic.sh** before the following line:

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

**Workshop Domain**

The **startWebLogic.sh** calls the utility **weblogic.Server** at the end. Copy **startWebLogic.sh** to **tvStartWebLogic.sh** and add the required additional environment setup in the copied **tvStartWebLogic.sh** before the following section:

```
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
     echo "Starting WLS with line:"
     echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy
${SERVER_CLASS}"
     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
${SERVER_CLASS}
else
     echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME} -
Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
${SERVER_CLASS}  1>&2 >"${WLS_REDIRECT_LOG}"
fi
```

The **startManagedWebLogic.sh** calls **startWebLogic.sh** at the end. Copy **startManagedWebLogic.sh** to **tvStartManagedWebLogic.sh** and edit the copied **tvStartManagedWebLogic.sh** to change the following line:

```
./startWebLogic.sh nodebug verbose nopointbase production
noiterativedev notestconsole $3 $4 $5 $6 $7 $8 $9
```

to:

```
./tvStartWebLogic.sh nodebug verbose nopointbase production
noiterativedev notestconsole $3 $4 $5 $6 $7 $8 $9
```

**Portal Domain**

The **startWebLogic.sh** calls the utility **weblogic.Server** at the end. Copy **startWebLogic.sh** to **tvStartWebLogic.sh** and add the required additional environment setup in the copied **tvStartWebLogic.sh** before the following section:

```
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
     echo "Starting WLS with line:"
     echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy
${SERVER_CLASS}"
     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME} -
Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
${SERVER_CLASS}
else
     echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME} -
Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
${SERVER_CLASS}  1>&2 >"${WLS_REDIRECT_LOG}"
fi
```

The s**tartManagedWebLogic.sh** calls **startWebLogic.sh** at the end. Copy **startManagedWebLogic.sh** to **tvStartManagedWebLogic.sh** and edit the copied **tvStartManagedWebLogic.sh** to change the following line:

```
./startWebLogic.sh nodebug nopointbase production
```

to:

```
./tvStartWebLogic.sh nodebug nopointbase production
```

**Integration Domain**

The **startWebLogic.sh** calls the utility **weblogic.Server** at the end. Copy **startWebLogic.sh** to **tvStartWebLogic.sh** and add the required additional environment setup in the copied **tvStartWebLogic.sh** before the following section:

```
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
     echo "Starting WLS with line:"
     echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy
${SERVER_CLASS}"
     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
${SERVER_CLASS}
else
     echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
${SERVER_CLASS}  1>&2 >"${WLS_REDIRECT_LOG}"
fi
```

The **`startManagedWebLogic.sh`** calls **`startWebLogic.sh`** at the end. Copy **`startManagedWebLogic.sh`** to **`tvStartManagedWebLogic.sh`** and edit the copied **`tvStartManagedWebLogic.sh`** to change the following line:

```
./startWebLogic.sh nodebug verbose nopointbase production
noiterativedev notestconsole $3 $4 $5 $6 $7 $8 $9
```

to:

```
./tvStartWebLogic.sh nodebug verbose nopointbase production
noiterativedev notestconsole $3 $4 $5 $6 $7 $8 $9
```

**Platform Domain**

The **`startWebLogic.sh`** calls the utility **`weblogic.Server`** at the end. Copy **`startWebLogic.sh`** to **`tvStartWebLogic.sh`** and add the required additional environment setup in the copied **`tvStartWebLogic.sh`** before the following section:

```
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
     echo "Starting WLS with line:"
     echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy
${SERVER_CLASS}"
     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
${SERVER_CLASS}
else
     echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Dweblogic.ProductionModeEnabled=${WLS_PRODUCTION_MODE}
-Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy"
${SERVER_CLASS}  1>&2 >"${WLS_REDIRECT_LOG}"
fi
```

The **`startManagedWebLogic.sh`** calls **`startWebLogic.sh`** at the end. Copy **`startManagedWebLogic.sh`** to **`tvStartManagedWebLogic.sh`** and edit the copied **`tvStartManagedWebLogic.sh`** to change the following line:

```
./startWebLogic.sh nodebug verbose nopointbase production
noiterativedev notestconsole $3 $4 $5 $6 $7 $8 $9
```

to:

```
./tvStartWebLogic.sh nodebug verbose nopointbase production
noiterativedev notestconsole $3 $4 $5 $6 $7 $8 $9
```

# Index

## A



## B



## C

# D



# E



# F



# I



# J



# L



# M

# O



# P



# R



# S

# T



# U



# V



# W