BRISTOL TECHNOLOGY®

TransactionVision®

# TransactionVision Sensor Installation and Configuration Guide

*Version 5.0.0*

39 Old Ridgebury Road | Danbury, CT 06810 | 203-798-1007

Printed January 30, 2006

This manual supports TransactionVision Release 5.0.

Part No. TV10060127

# Contents

Chapter 1

# TransactionVision Sensor Overview

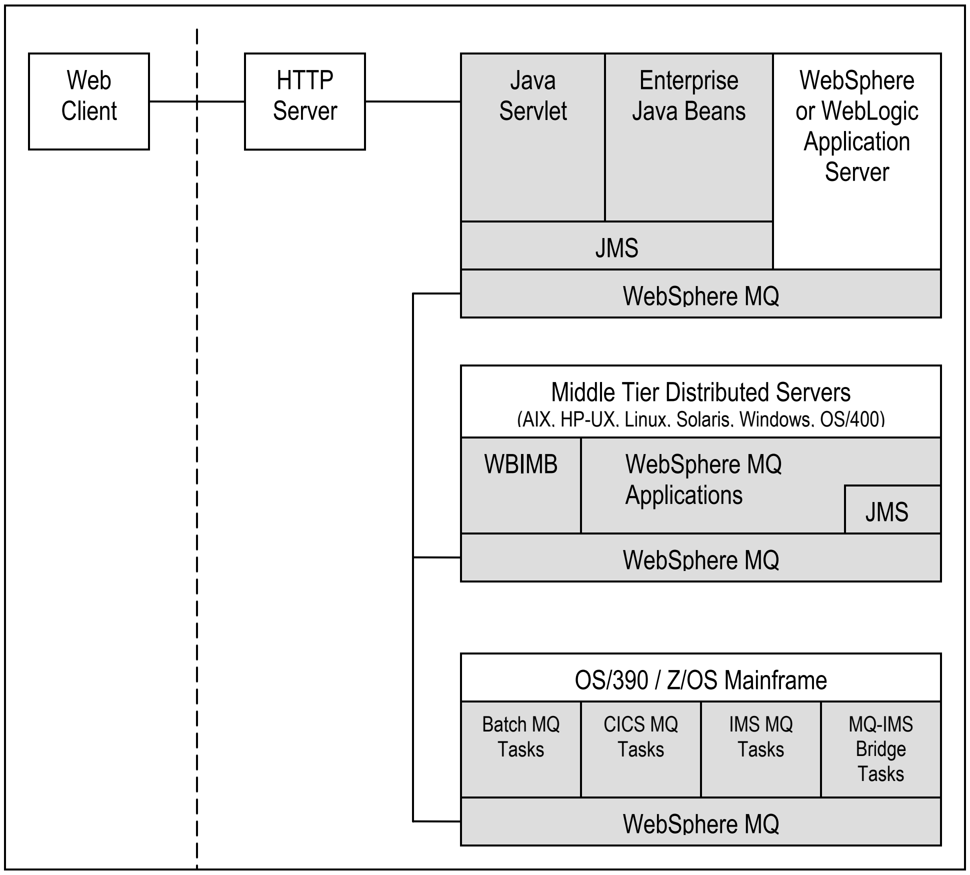
TransactionVision Sensors collect transactional events from the various applications involved in your distributed transactions. Sensors are lightweight libraries or exit programs that are installed on each computer in your environment. Each Sensor monitors calls made by supporting technologies on that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

## Available Sensor Types

TransactionVision provides several types of Sensors:

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

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



## 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 or TIBCO EMS events from standalone Java applications as well as from J2EE application servers. This Sensor tracks JMS interface methods such as send, receive, etc. These methods are tracked by instrumenting the JMS library to collect events at the entry and exit of each call.

### 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.

## Installation Overview

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. For instructions on installing and configuring the Analyzer, see the *TransactionVision Analyzer Installation and Configuration Guide*.
- 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. For instructions on installing and configuring the Web User Interface, see the *TransactionVision Web Application Installation and Configuration Guide*.
- TransactionVision Sensors. Install these packages on all hosts running applications that you wish to collect information about. This guide provides instructions for installing Sensors.

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

Before installing Sensors, 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). See the TransactionVision Release Notes for detailed software requirements.

## Upgrading from Previous Sensor Releases

The Java Servlet, JMS and EJB Sensors from TransactionVision 4.2.1 can be used with the TransactionVision 5.0.0 Analyzer. TransactionVision 5.0.0 Sensors may not be used with older versions of the TransactionVision Analyzer.

## Additional TransactionVision Resources

### Documentation Roadmap

This guide provides instructions for installing and configuring TransactionVision Sensors. In addition to this guide, the following documents are provided with TransactionVision:

- The *TransactionVision Planning Guide* provides information to help you plan the TransactionVision implementation in your environment.
- The *TransactionVisionWeb Application Installation and Configuration Guide* provides instructions for installing and configuring the TransactionVision web user interface. This file is also available from the TransactionVision Help menu.
- The *TransactionVision Analyzer Installation and Configuration Guide* provides instructions for installing and configuring the TransactionVision Analyzer, and setting up your database for the Analyzer. This file is also available from the TransactionVision Help menu.
- The *TransactionVision Administrator's Guide* provides instructions managing user accounts and communication links, configuring projects and data collection filters, and managing services and schemas. This file is also available from the TransactionVision Help menu.
- The *TransactionVision User's Guide* provides instructions using TransactionVision analysis views. This file is also available from the TransactionVision Help menu.

- The *TransactionVision Programmer's Guide* provides information for creating custom beans and reports for use with TransactionVision.
- The *TransactionVision Security Guide* provides an overview of the security features and setup procedures of TransactionVision. These features and procedures ensure that data collected by TransactionVision is secure and accessible to the appropriate people.

## 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

Chapter 2
# Installing Sensors on Distributed Platforms

To install TransactionVision Sensors on distributed (UNIX) platforms, perform the following steps for each type of Sensor. Note that TransactionVision provides unique packages for the Servlet/EJB, JMS, User Event, and WebSphere MQ Sensors.

## Installing Sensors

### Installation Files

The following table shows the installation file names for the TransactionVision packages 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 |
|---|---|
| AIX | tvision_common_500_aix.bff<br>tvision_sensor_jms_500_aix.bff<br>tvision_sensor_appserv_500_aix.bff<br>tvision_sensor_wmq_500_aix.bff<br>tvision_sensor_userevent_500_aix.bff |
| HP-UX | tvision_sensor_wmq_500_hpux11_tar.gz |
| Linux | tvision_common_500_linux.rpm<br>tvision_sensor_jms_500_linux.rpm<br>tvision_sensor_appserv_500_linux.rpm<br>tvision_sensor_wmq_500_linux.rpm<br>tvision_sensor_userevent_500_linux.rpm |
| Solaris | tvision_common_500_solaris_tar.gz<br>tvision_sensor_jms_500_solaris_tar.gz<br>tvision_sensor_appserv_500_solaris_tar.gz<br>tvision_sensor_wmq_500_solaris_tar.gz<br>tvision_sensor_userevent_500_solaris_tar.gz |

## Using the JMS Sensor with WebSphere Application Server 5.1

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 Steps

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.

   Unzipping/Untaring common package files ...
   Unzipping/Untaring Analyzer package files ...
   Unzipping/Untaring Web package files ...
   Unzipping/Untaring Application Server Sensor package
   files ...
   Unzipping/Untaring JMS Sensor package files ...
   Unzipping/Untaring WebSphere MQ Sensor package files
   ...
   Unzipping/Untaring User Event Sensor package files
   ...
   ```

   ***Important!*** The `"Unzipping…"` lines above are only for Solaris and HP-UX installations.

   ```
   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
6. TransactionVision User Event Sensor

99. All of above
q. Quit install

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

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

4. 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. For example, to install all Sensors from the menu above, type the following and press Return:

   ```
   3,4,5,6
   ```

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

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

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

***Important!*** If WebSphere 6.0 is shutdown during the installation while there is an active session connected to the WebSphere Application Server administration user interface, you will see the following message upon restart and logging back into the administration console:

```
The master configuration has been updated. You currently
have workspace conflicts with these modifications. To see
these updates you must save or discard your current
worskpace modifications.
```

If you choose to save your workspace modifications at this point, any changes that the TransactionVision utilities had just made during installation will be reversed. Instead, choose to discard your current workspace modifications. Any new modifications can be made once the master configuration is then reloaded.

## Rebinding the WebSphere MQ Sensor on AIX

For the WebSphere MQ Sensor on the AIX platform, the installation calls the **`rebind_sensor`** script to relink the Sensor library. If you install a

WebSphere MQ support pack that modifies the WebSphere MQ libraries (`libmqm.a`, `libmqic.a`, `libmqm_r.a`, `libmqic_r.a`), you must run this script again in order for sensored applications to run correctly. For more information about this script, see Appendix A, "Utilities Reference."

## Uninstalling Sensors

To uninstall TransactionVision components, perform the following steps:

1. 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
   6. TransactionVision User Event 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 are using WebSphere Application Server and 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!*** For WebSphere Application Server, you must clean up the TransactionVision temporary cache and distribution directory after uninstalling the TransactionVision web user interface. WebSphere does not do this automatically, and any old files could cause a new installation to work incorrectly.

Chapter 3

# Installing Sensors on Windows

The TransactionVision Sensors for Servlet, EJB, JMS, User Event, and WebSphere MQ 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_500_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."

***Important!*** If WebSphere 6.0 is shutdown during the installation while there is an active session connected to the WebSphere Application Server administration user interface, you will see the following message upon restart and logging back into the administration console:

```
The master configuration has been updated. You currently
have workspace conflicts with these modifications. To see
these updates you must save or discard your current
worskpace modifications.
```

If you choose to save your workspace modifications at this point, any changes that the TransactionVision utilities had just made during installation will be reversed. Instead, choose to discard your current workspace modifications. Any new modifications can be made once the master configuration is then reloaded.

## Initial Installation

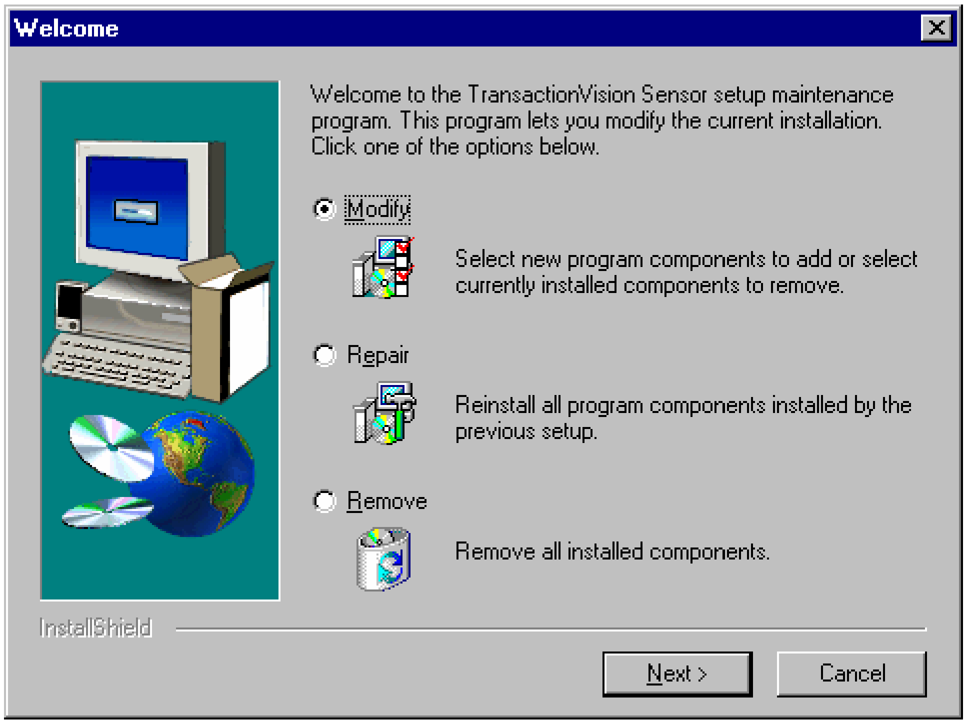
For an initial installation, the Setup Welcome screen is displayed.

1. On the Setup Welcome screen, click Next> to display the TransactionVision license agreement.
2. Click Yes to accept the license agreement. The User Information screen appears.
3. Enter your name and company name, then click Next>. The Destination Location screen appears.
4. To install the Sensors for Servlet, EJB, JMS, User Event, and WebSphere MQ, 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:


Welcome

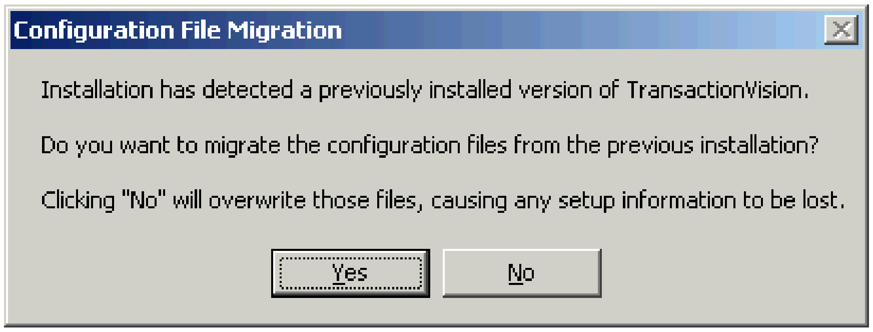
Welcome to the TransactionVision Sensor setup maintenance program. This program lets you modify the current installation. Click one of the options below.

● Modify
Select new program components to add or select currently installed components to remove.

○ Repair
Reinstall all program components installed by the previous setup.

○ Remove
Remove all installed components.

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:


Configuration File Migration

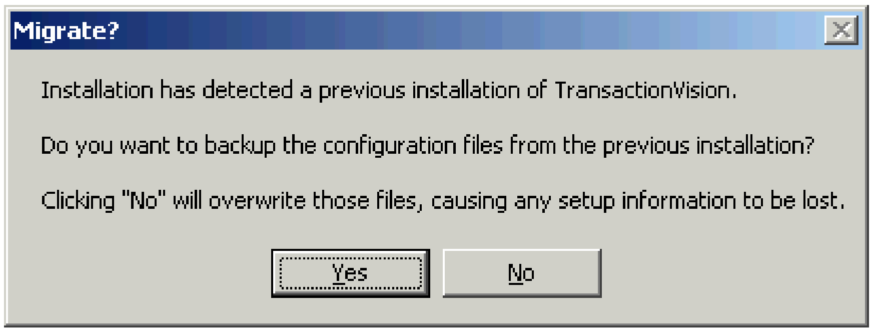
Installation has detected a previously installed version of TransactionVision.

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

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

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

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


Migrate?

Installation has detected a previous installation of TransactionVision.

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

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

Click Yes to create a backup copy or No to continue the installation without backing up configuration files. The installation wizard then installs the new version of 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\be asvc <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"
```

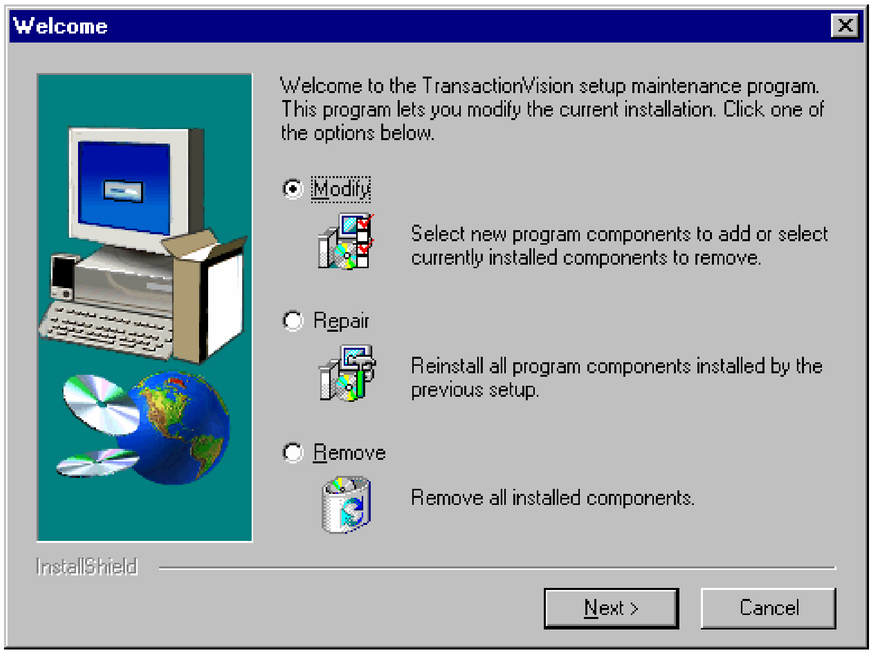
3. Prepend the following classes to the classpath, replacing <TVISION_HOME> with your TransactionVision installation path and <WMQ_HOME> with your WebSphere MQ installation path:

```
<TVISION_HOME>\java\lib\tvisionsensorservlet.jar;<TVI
SION_HOME>\java\lib\tvisionsensorejb.jar;<TVISION_HOM
E>\java\lib\tvjavaext.jar;<TVISION_HOME>\java\lib\web
logic.jar;<TVISION_HOME>\java\lib\webservices.jar;<TV
ISION_HOME>\java\lib\tvisionsensorjms.jar;<TVISION_HO
ME>\java\lib\com.ibm.mqjms.jar;<WMQ_HOME>\Java\lib;<W
MQ_HOME>\Java\lib\com.ibm.mqjms.jar;<WMQ_HOME>\Java\l
ib\fscontext.jar;<WMQ_HOME>\Java\lib\providerutil.jar
;<WMQ_HOME>\Java\lib\connector.jar;<WMQ_HOME>\Java\li
b\jta.jar;<WMQ_HOME>\Java\lib\com.ibm.mqbind.jar;<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_500_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:



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

[✓ Modify]

Select new program components to add or select currently installed components to remove.

Repair

Reinstall all program components installed by the previous setup.

Remove

Remove all installed components.

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.

## Uninstalling Sensors

To uninstall TransactionVision components, perform the following steps:

1. From the Start menu, choose Settings > Control Panel.
2. Double-click Add/Remove Programs.
3. Select the 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

# Installing Sensors 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 `sensor500.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 sensor500.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 sensor500.savf package resides

   ```
   RSTLICPGM LICPGM(3RBB9ES) DEV(*SAVF)
   SAVF(TVTMP/SENSOR500)
   ```

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.

Chapter 5
# Installing Sensors on z/OS

This chapter provides instructions for installing the forllowing TransactionVision Sensors on the IBM z/OS platform:

- CICS, WMQ Batch, and WMQ IMS
- WebSphere MQ CICS and WMQ-IMS Bridge

For additional requirements for RACF authorizations, firewall settings, and MIPS, see Appendix E.

## Installing the CICS, WMQ Batch, and WMQ IMS Sensor on IBM z/OS

To install these Sensors on the z/OS platform, perform the following steps, substituting a valid data set name high-level qualifier for &hlq, for example, TVISION.

Once you complete this installation procedure, see the *TransactionVision Administration Guide* for additional configuration instructions.

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 data set attributes on the quote command. For example:

   ```
   ftp> quote site fixrecfm 80 lrecl=80 recfm=fb blksize=3120
   ftp> bin
   ftp> put sld500.f1 '&hlq.sld500.f1'
   ftp> put sld500.f2 '&hlq.sld500.f2'
   ftp> put sld500.f3 '&hlq.sld500.f3'
   ftp> put sld500.mcs '&hlq.sld500.mcs'
   ```

   ***Important!*** If you receive a data set full error during this step (typically with SLD500.F2), use the following data set 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
```

Note that if you received the data set full error attempting to ftp the f2 file, the target data set has already been allocated and must be deleted before re-attempting the ftp in order for the new allocation parameters to take effect. If the target data set is not deleted, subsequent put commands will attempt to use the existing data set and get the same failure.

2. Use the TSO RECEIVE command to create the product distribution data sets 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.SLD500.F1') | DSN('&hlq.ASLD500.F1') |
| RECEIVE INDSNAME('&hlq.SLD500.F2') | DSN('&hlq.ASLD500.F2') |
| RECEIVE INDSNAME('&hlq.SLD500.F3') | DSN('&hlq.ASLD500.F3') |
| RECEIVE INDSNAME('&hlq.SLD500.MCS') | DSN('&hlq.ASLD500.SMPMCS') |

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

| **Data set** | **Member(s)** | **Description** |
|---|---|---|
| &hlq.ASLD500.F1 | ASLD500 | SMP/E JCLIN |
| &hlq.ASLD500.F2 | SLDMOD01-35 | Sensor modules |
| | MQCONNX | Dummy module for Web-Sphere 5.1 installations. |
| &hlq.ASLD500.F3 | DUMYCICS | Sample customization job |
| | DUMYIMS | Sample customization job |
| | 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 |
| | TVISIONM | Sample product startup procedure |
| | TVISIONR | Sample recovery procedure |
| | VERSION | Build number information |
| &hlq.ASLD500.SMPMCS | SMP/E MCS | |

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

4. A sample job is provided for each of the major installation steps. Modify the following members for your local requirements as directed by the member's commentary before submitting them in the following steps:

```
&hlq.ASLD500.F3(SLDALLOC)
&hlq.ASLD500.F3(SLDAPPLY)
&hlq.ASLD500.F3(SLDDDEF)
&hlq.ASLD500.F3(SLDDZON)
&hlq.ASLD500.F3(SLDGZON)
&hlq.ASLD500.F3(SLDRECV)
&hlq.ASLD500.F3(SLDTZON)
```

Optionally, you may copy and then modify these members.

You may perform a non-SMP/E install by customizing and running the SLDINSTL member, skipping the SMP/E steps (Steps 5-9), and resuming with Step 10. Also, skip Step 18. For an SMP/E install, perform all remaining steps. The SMP/E FMID for this installation is ASLD500.

5. 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 |

| Library | Description |
|---|---|
| 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 |

6. 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.
7. 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.

   ***Important!*** If you do NOT have CICS installed on your system, customize and submit the JCL DUMYCICS to generate dummy CICS modules. Then set the CICS high level qualifier (&CICSQUAL) to be the same as the target library high level qualifier (%tvlib). If you do NOT have IMS installed on your system, customize and submit the JCL DUMYIMS to generate dummy IMS modules. Then set the IMS high level qualifier (&IMSQUAL) to be the same as the target library high level qualifier (%tvlib).

   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. |
| 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. |

| Library | Description |
|---|---|
| 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. |

8. SMP/E RECEIVE the Sensor installation package product by customizing and running the SLDRECV member.
9. 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, SLDP-PCX, SLDPPSX, SLDPTCX, SLDPTDX, SLDPTSX, MQCONNX, SLDPMDR, SLDPMECB, SLDPMECQ, SLDPMECR, SLDPMECS, SLDPMEM, SLDPSTBB, SLDPSTBQ, SLDPSTBR, SLDPSTBS |
| &THLQ.SSLDINST | SLDACCPT, SLDALLOC, SLDAPPLY, SLDCICSD, SLDCRTQS, SLDDDDEF, SLDDZON, SLDGZON, SLDINSTL, SLDRECV, SLDTZON, VERSION |
| &THLQ.SSLDPROC | TVISION, TVISIONC, TVISIONM, TVISIONR |
| &THLQ.SSLMSAMP | No members in this release |
| &THLQ.SSLDAUTH | SLDPASM, SLDPBQM, SLDPCMD, SLDPCSC, SLDPCSI, SLDPCSM, SLDPITM, SLDPMON, SLDPSSS, SLDPTVM, SLDPMSI, SLDPMSM |

10. You may wish to review Chapter 11 in the *TransactionVision Adminisrator's Guide* before performing the remaining installation steps.
11. 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.
12. 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.

13. 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
    //
    ```

14. 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.

15. 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.

16. 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 11 in the *TransactionVision Adminisrator's Guide* for guidelines for customizing the startup procedures.
17. 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 11 in the *TransactionVision Adminisrator's Guide* for more information.
18. 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 CICS and WMQ-IMS Bridge Sensor on IBM z/OS

There is a single Sensor installation package for WebSphere MQ CICS and WMQ-IMS bridge Sensors on the z/OS. 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 WMQ-IMS bridge Sensor also requires additional steps, which are detailed in "Additional Setup for the WMQ-IMS Bridge Sensor" on page 38. If you do not intend to use the WMQ-IMS bridge Sensor, omit those steps.

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

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

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

The WebSphere MQ Sensor for z/OS CICS requires 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 these Sensors on z/OS, perform the following steps:

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

   ```
   ftp> quote site fixrecfm 80 lrecl=80 recfm=fb
   blksize=3120
   ftp> bin
   ftp> put slm500.f1 '&hlq.slm500.f1'
   ftp> put slm500.f2 '&hlq.slm500.f2'
   ftp> put slm500.f3 '&hlq.slm500.f3'
   ftp> put slm500.mcs '&hlq.slm500.mcs'
   ```

   ***Important!*** If you receive a data set full error during this step (typically with SLM500.F2), use the following data set 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
   ```

   Note that if you received the data set full error attempting to ftp the f2 file, the target data set has already been allocated and must be deleted before re-attempting the ftp in order for the new allocation parameters to take effect. If the target data set is not deleted, subsequent put commands will attempt to use the existing data set and get the same failure.

2. Use the TSO RECEIVE command to create the SMP/E input data sets 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.SLM500.F1’)` | `DSN(‘&hlq.ASLM500.F1’)` |
| `RECEIVE INDSNAME(‘&hlq.SLM500.F2’)` | `DSN(‘&hlq.ASLM500.F2’)` |
| `RECEIVE INDSNAME(‘&hlq.SLM500.F3’)` | `DSN(‘&hlq.ASLM500.F3’)` |
| `RECEIVE INDSNAME(‘&hlq.SLM500.MCS’)` | `DSN(‘&hlq.ASLM500.SMPMCS’)` |

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

| Data set | Member | Description |
|---|---|---|
| `&hlq.ASLM500.SMPMCS` | | SMP/E MCS file for Sensor |
| `&hlq.ASLM500.F1` | `ASLM500` | JCLIN for SMP/E |
| `&hlq.ASLM500.F2` | `DFHEAII` | Dummy module for CICS |
| | `MQCONNX` | Dummy module for MQ51 |
| | `SLMBCNFG` | Configuration queue table |
| | `SLMMOD01` | Sensor crossing exit program for z/OS CICS |
| | `SLMMOD02` | SLMC crossing exit management program for z/OS CICS |
| | `SLMMOD10` | WMQ-IMS bridge support module |
| | `SLMMOD11` | WMQ-IMS bridge support module |
| | `SLMMOD12` | WMQ-IMS bridge support module |
| | `SLMMOD13` | WMQ-IMS bridge support module |
| | `SLMMOD14` | WMQ-IMS bridge support module |
| | `SLMMOD15` | WMQ-IMS bridge support module |
| | `SLMMOD16` | WMQ-IMS bridge support module |
| | `SLMMOD17` | WMQ-IMS bridge support module |
| | `SLMMOD18` | WMQ-IMS bridge support module |
| | `SLMMOD19` | WMQ-IMS bridge support module |
| | `SLMYIOE0` | WMQ-IMS bridge support module |

| Data set | Member | Description |
|---|---|---|
| `&hlq.ASLM500.F3` | `DUMYCICS` | Sample customization job |
| | `DUMYIMS` | Sample customization job |
| | `SLMACCPT` | Sample JCL for SMP/E Accept step |
| | `SLMALLOC` | Sample JCL for TransactionVision SMP/E target and distribution library allocation |
| | `SLMAPPLY` | Sample JCL for SMP/E Apply step |
| | `SLMBCFGQ` | Sample job to change the default TransactionVision configuration queue name for the batch and IMS Sensors or the WMQ-IMS bridge Sensor |
| | `SLMCICSD` | Sample JCL for updating CICS CSD file with Sensor program definition |
| | `SLMCRTQS` | Sample JCL to create TransactionVision communi-cation queues |
| | `SLMDDDEF` | Sample JCL for creating SMP/E DDDEFs for TransactionVision |
| | `SLMDZON` | Sample JCL to create the SMP/E distribution zone |
| | `SLMGZON` | Sample JCL to create the SMP/E global zone |
| | `SLMINSTL` | Sample non-SMP/E install job |
| | `SLMRECV` | Sample JCL for SMP/E Receive step |
| | `SLMTZON` | Sample JCL to create the SMP/E target zone |
| | `TVISIONB` | Sample procedure to start the WMQ-IMS bridge Sensor control address space |
| | `TVISIOND` | Sample procedure to start the WMQ-IMS bridge Sensor event dispatcher address space |
| | `VERSION` | Text file containing TransactionVision build number information |

If any of the above data sets 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 data set, edit it appropriately for your local requirements, and use it to perform the installation.

The SMP/E FMID for this installation is ASLM500.

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

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.

   ***Important!*** If you do NOT have CICS installed on your system, customize and submit the JCL DUMYCICS to generate dummy CICS modules. Then set the CICS high level qualifier (&CICSQUAL) to be the same as the target library high level qualifier (%tvlib). If you do NOT have IMS installed on your system, customize and submit the JCL DUMYIMS to generate dummy IMS modules. Then set the IMS high level qualifier

(&IMSQUAL) to be the same as the target library high level qualifier (%tvlib).

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. |
| 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. |
| 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. |

| Library | Description |
|---|---|
| distribution zone DDEF ASLMPROC | Points to the ASLMPROC distribution library created in Step 4. |

7. 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 ASLM500.* data sets. For example, if the data sets were installed in USERNAME.ASLM500.* and your global CSI data set 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.ASLM500.SMPMCS,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SMPCNTL DD *
SET BDY(GLOBAL).
RECEIVE SELECT(ASLM500) RFPREFIX(USERNAME) SYSMODS LIST.
/*
```

8. 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, SLMC, SLMCSTUB, SLMXSRV, BTMQEXIT, SLMBCNFG |
| &THLQUAL.SSLMINST | SLMACCPT, SLMALLOC, SLMAPPLY, SLM-CICSD, SLMCRTQS, SLMDDDEF, SLMDZON, SLMGZON, SLMRECV, SLMTZON, SLMCPYDC, SLMCPYDI, VERSION, SLMINSTL |
| &THLQUAL.SSLMINCL | BTMQEXIT, BTTRACE |
| &THLQUAL.SSLMPROC | SLMEJCL, SLMLKSTB, SLMTJCL, SLMTRJCL, TVISIONB, SLMBCFGQ, TVISIOND |
| &THLQUAL.SSLMSAMP | SLMEXITS, SLMTSAMP |
| &THLQUAL.SSLMSRC0 | SLMEXITD |
| &THLQUAL SSLMAUTH | DFSYIOE0, SLMXCMD, SLMXCTL, SLMXINT, SLMXMON, SLMXRDQ, SLMXTKC, SLMXWRQ, SLMXXMC, SLMYIOE0 |

9. (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 SLMCICSD 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.
10. (CICS only) Place the CSQCAPX and SLMC load modules in a library within your DFHRPL concatenation. You can either copy the modules into an existing library already present in your DFHRPL concatenation or you can add the SSLMLOAD library to the DFHRPL concatenation.
11. (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).
12. 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. Chapter 6, “Configuring WebSphere MQ Sensors,” for more information about Sensor configuration and event queues.
13. 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 data sets used by CICS to supply terminal input and output. The input data set contains the CICS transaction names you want to run (SLMC in this case) and the output data set receives the terminal output from the transaction(s). Creating a sequential terminal definition requires you to assemble CICS terminal resource macro definitions using DFHAUPLE, set the TCT parameter in your SIT to enable reading of the TCT in which your assembled macros are placed, and add DD names to your CICS region's startup JCL to define the input and output data sets for the terminal.

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

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

```
SLMC\
CESF LOGOFF\
```

***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 WMQ-IMS Bridge Sensor

The WMQ-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 WMQ-IMS bridge Sensor, see Chapter 6, “Configuring WebSphere MQ Sensors.”

# 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 z/OS 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` |

| Platform | Environment Variable | Default Directory |
|---|---|---|
| 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.

On the HP-UX platform, you may have to use the `chatr` command to enable your application to pick up the Sensor library if the use of the `SHLIB_PATH` environment variable is not enabled or is not the first in the dynamic library search path for your application. You may use the `chatr` command to check the current settings of your application. In any case, make sure that `SHLIB_PATH` is enabled and is before the standard WebSphere MQ library path. See Chapter 10 in the *TransactionVision Administrator's Guide* for details.

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
```

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

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

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

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

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

### z/OS CICS

On the z/OS CICS platform, Sensors use the API-crossing exit mechanism provided by the CICS adapter of WebSphere MQ for z/OS.

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

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

1. Edit thlqual.SSLMPROC(SLMBCFGQ) and change as follows:
   (a) Change the value of the SET CONFIGQ statement to specify the desired configuration queue name.

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

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

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

For more information about the WebSphere MQ-IMS bridge Sensor, see "Using the WebSphere MQ-IMS Bridge Sensor" on page 57.

## IBM z/OS CICS

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

If the program SLMCNFQ can be executed by the Sensor via an EXEC CICS LINK, the Sensor does so, passing SLMCNFQ a CICS comm area large enough to hold a configuration queue name (48 bytes). The comm area initially contains the default configuration queue name (TVISION.CONFIGURATION.QUEUE). When 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 z/OS 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 z/OS CICS WebSphere MQ Sensor tries to load the program SLMBCNFG and gets the configuration queue name. For instructions on generating this program, see “Generating SLMBCNFG..”

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 Queues 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 z/OS batch and IMS; that is:

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

### Using Separate Configuration Queues for Each CICS Region

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

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

   ```
   DEFINE PROGRAM(SLMBCNFG) GROUP(BTITV)
      DESCRIPTION(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 not collected by the API Exit Sensor and are set to fixed values of MQCC_OK and MQRC_NONE, respectively. The event time for MQDISC events is set to the before-MQDISC function invocation time.
- The API Exit Sensor collects some TransactionVision internal events generated from 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 the `qm.ini` file, which applies to individual queue managers. The `mqs.ini` file is typically located in the directory `/var/mqm`. The `qm.ini` file is typically in the directory `/var/mqm/qmgrs/<qmgr_name>`. A stanza consists of a section header followed by a colon, which is then followed by lines containing attribute/value pairs separated by the "=" character. Note that the same attributes may be used in either mqs.ini or qm.ini.

### 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."

- `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. On OS/400, this file name is `ALLOW.MBR`.
- `exit_sensor.deny` defines which programs will not be monitored. All other programs will be monitored. On OS/400, this file name is `DENY.MBR`. This file is shipped with the WebSphere MQ Sensor and contains some default programs from which events are not collected by the API Exit Sensor, such as the WebSphere MQ command server.

These files are located at the top level TransactionVision installation directory. For example, on Solaris if TransactionVision is installed at /opt/TVision, these two files exist in the /opt/TVision directory. On OS/400, these files are in /qsys.lib/tvsensor.lib/EXITSENSOR.FILE.

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

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

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

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:

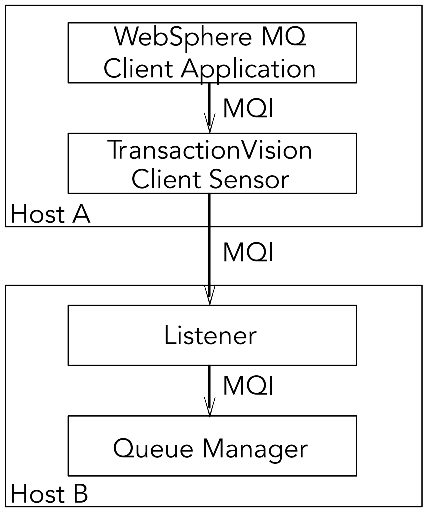
- There are two locations for WebSphere MQ samples on the Windows platform. If you run the samples under WebSphere MQ\bin, you must copy the sample executables (amqsput, amqsget, etc.) to a different directory to enable them to load the Sensor library instead of the standard WebSphere MQ library. This 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:


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

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

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

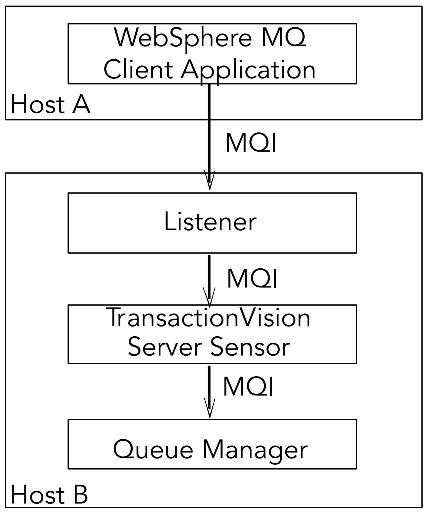
The trace information generated will be based on the client application context. This means information such as program name, program instance, and host name, will be related to the client application directly.

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

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

## Centralized Monitoring

Centralized monitoring of the WebSphere MQ listener program is only supported using the API Exit Sensor and is not supported with the library sensor. The following diagram illustrates how the Sensor works in a centralized monitoring environment:


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

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.

## 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 60.

   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

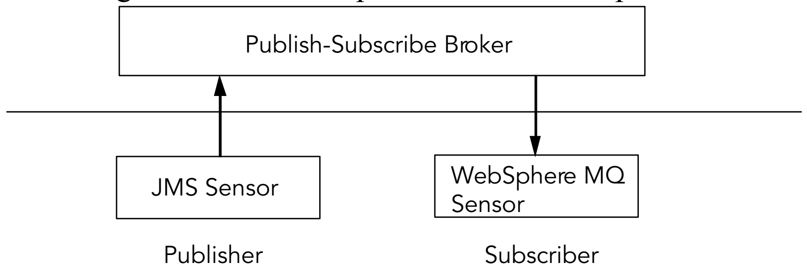
You may now insert any number of TransactionVision Sensor trace and failure nodes into any message flows through the Message Brokers Toolkit. Remember that any changes to the configuration repository must be deployed to the appropriate brokers.

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. |

| Element | Required? | Attributes | Description |
|---|---|---|---|
| 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 z/OS 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 z/OS 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 z/OS IMS job, specify the PSB name, transaction name, region ID, job name, IMS ID and IMS type. |

## Optional Attributes for the Proxy Element

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

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

## Configuring the User Interface

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

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 JMS, TIBCO EMS, 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 9, “Configuring Sensor 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. For WebSphere, the following prompt appears:

```
Does your target Application Server use a different
configuration instance from the default WebSphere install
directory? [y]:
```

Answer **y** if your WebSphere Application Server has a separate configuration instance. A WebSphere configuration instance can be duplicated using the **wsinstance** command. If you answer **y**, you will be prompted to enter the directory location of the duplicate configuration instance:

```
Enter the location of your WebSphere Application Server
configuration instance [c:\temp\wsin]:

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

## JMS Provider Prompts

```
Retrieving Sensor configuration parameters...

The following configuration parameters are required for
the TransactionVision Sensor to correctly retrieve its
configuration messages.

Please choose the JMS provider that you want to monitor:

        1.  WebSphere MQ
        2.  TIBCO EMS
        3.  All the above JMS providers.

Your selection of JMS provider [3]:
```

Enter the number corresponding to the JMS provider to be monitored. If you want to monitor both IBM WebSphere MQ JMS and TIBCO EMS JMS activities, select option **3**.

```
Please choose the JMS provider used for sending events to
the Analyzer:

        1.  WebSphere MQ
        2.  TIBCO EMS

Your selection of JMS provider [1]:
```

Enter the number corresponding to the  JMS provider that will be used to provide message communication between JAVA Sensors and the TransactionVision Analyzer. This setup step only appears when you choose to monitor both JMS providers (WebSphere MQ and TIBCO EMS) at the same time. When only one JMS provider is monitored, the messaging system will be set automatically to use the JMS provider you are monitoring.

### IBM WebSphere MQ Messaging System:

```
NOTE: If any JMS applications (within an application server
or stand alone) being monitored with TransactionVision use
a server connection to a different queue manager than
configured here, it is highly recommended to configure the
Sensor to use a client connection as there are limitations
in WebSphere MQ when using server connections to multiple
queue managers.  This limitation could result in
TransactionVision or even the JMS application failing.

Configuration queue manager [configuration_qm_name]:
shuangypc.tv1.manager
```

Enter the name of WebSphere MQ queue manager

```
Configuration queue [TVISION.CONFIGURATION.QUEUE]:
```

Enter the name of WebSphere MQ queue

```
Type of Connection to Configuration Queue Managers for
Server Connection, c for Client Connection (s/c): c
```

Whether use server connection or 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 is located
[configuration_qm_host]: shuangypc
```

The host where WebSphere MQ queue manager resides

```
Port number client uses to connect to configuration queue
manager [1414]: 1415
```

The message listener port of WebSphere MQ queue manager

```
Name of channel client uses to connect to configuration
queue manager [configuration_qm_channel]: SHUANGYPC.TV1.CHL
```

The global client connection channel name.

## TIBCO EMS as Messaging System:

```
NOTE: The following TIBCO EMS parameters are used by the
Sensor to connect to the TIBCO EMS Server for reading
filter configuration messages and writing events. The User
name and password may be left blank if your TIBCO server is
not setup with authorization enabled.
```

```
  TIBCO EMS Server Host Name []: shuangypc
```

Enter the host name of TIBCO EMS server.

```
  TIBCO EMS Server Port Number [7222]:
```

Enter the port number of TIBCO EMS server.

```
  User Name []: admin
```

Enter the name of the user that has to permission to operate on configuration queue and event queue, if authentication is enabled on TIBCO EMS server.

```
  Password []:
```

Enter the password for the user, if specified.

```
  Configuration queue [TVISION.CONFIGURATION.QUEUE]:
```

Enter the queue name where the Sensor retrieves configuration messages delivered by the Analyzer.

***Important!*** You must setup the administration information of all monitored TIBCO EMS servers so that the JMS Sensor can correctly resolve the TIBCO routing and bridging relation between TIBCO EMS queues and TIBCO EMS topic.

```
The TransactionVision Sensor optionally collects TIBCO EMS
server configuration information to resolve TIBCO  EMS
objects,  such as bridges and routes.  To do this, user
name and password need to be entered for each TIBCO EMS
server being monitored.

Please select one of the following options:

        1.  Add a TIBCO EMS server
        2.  Edit a TIBCO EMS server (disabled)
        3.  Remove a TIBCO EMS server (disabled)
        4.  List TIBCO EMS servers (disabled)

        5.  Finished entering TIBCO EMS servers

Please select TIBCO EMS Server option [1]:
```

Menu item 2, 3, and 4 are enabled when there is at least one TIBCO EMS sever information provided in the configuration.

### Add a TIBCO EMS Server

```
TIBCO server name: shuangypc
```

Enter the TIBCO server named defined in `tibemsd.conf`.

```
TIBCO EMS Server Host Name: shuangypc
```

Enter the name of the host where the TIBCO server resides.

```
TIBCO EMS Server Port Number [7222]:
```

Enter the listen ports of the TIBCO EMS server

```
TIBCO server admin user name: admin.
```

Enter the name of the user that has “all” administrative permission.

```
TIBCO server admin user password:
```

Enter the password of the user that has “all” administrative permission.

### Edit a TIBCO EMS Server

```
All TIBCO servers administration information entered

        1. shuangypc
```

```
Please select the server you want to Edit: 1

TIBCO server name [shuangypc]:
```

Enter the TIBCO server named defined in `tibemsd.conf`.

```
TIBCO EMS Server Host Name [shuangypc]:
```

Enter the name of the host where the TIBCO server resides.

```
TIBCO EMS Server Port Number [7222]:
```

Enter the listen ports of the TIBCO EMS server.

```
TIBCO server admin user name [admin]:
```

Enter the name of the user that has “all” administrative permission.

```
TIBCO server admin user password []:
```

Enter the password of the user that has “all” administrative permission.

### Delete TIBCO EMS Server

```
All TIBCO servers administration information entered

        1. shuangypc

        2. mcnealy

Please select the server(s) you want to remove. Multiple
servers can be listed here and separated by ",": 2
```

Select the number corresponding to the TIBCO server you wish to remove. To remove multiple servers, separate the numbers with a comma (for example, "1,2").

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

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.

```
TIBCO EMS installation location [C:\tibco\ems]:
```

Enter the TIBCO EMS installation path. This setup step only shows up when TIBCO EMS is used.

```
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]:
cgServer
```

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.

```
File name of WebLogic domain startup script
[startWebLogic.cmd]:
```

Enter the file name of the WebLogic server startup script.

```
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.

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 in your Application
Server? (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, EJB 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".
```

***Important!*** To monitor a WebLogic Admin Server, use the **tvStartWebLogic.[cmd|sh]** script created by SensorSetup instead of the **startWebLogic.[cmd|sh]** supplied by WebLogic. Similarly, to monitor a WebLogic Managed Server, use the **tvStartManagedWebLogic.[cmd|sh]** script instead of **startManagedWebLogic.[cmd|sh]**.

## 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.

```
TIBCO EMS installation location [C:\tibco\ems]:
```

Enter the TIBCO EMS installation path. This setup step only shows up when TIBCO EMS is used.

```
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.

The following menu appears the first time you run SensorSetup:

```
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 ...

The following WebSphere profiles are available on this
host:

  1. default
  2. Profile2
```

The previous question appears only if you are using WebSphere 6 and multiple profiles are present. Otherwise, the following question is displayed:

```
The following WebSphere servers are available on this host:

  1. cell: testpc, node: testpc, server: server1
  2. cell: testpc, node: testpc, server: server2

Plese select the WebSphere server by number: 1
```

If you run TVisionSetupInfo at a later time to change any information, the following menu appears:

```
Your current WebSphere server setting is:
            cell: testpc, node: testpc, server: server1
Would you like to change this? (y/n) [n]:
```

To change information, enter **y** to display the initial configuration menu. Select the desired server. To use the same information, enter **n**.

```
The following WebSphere virtual hosts are available on this
host:

  1. admin_host
  2. default_host

Please select the WebSphere virtual host by number: 2
```

If TVisionSetupInfo cannot obtain this information from WebSphere, you will be required to input the cell name, node name, server name, and virtual host strings manually.

```
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

Do you want to automatically set up Sensors on your
Application Server? (y/n) [y]: y
```

Reply ‘y’ if you want to automatically instrument the required jar file and modify the web server runtime environment to install the TransactionVision java sensor. Reply ‘n’ will end the SensorSetup script.

```
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 Steps 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 D, "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 the *TransactionVision Analyzer Installation and Configuration Guide*.

## 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. Make sure that the deployment manager and node manager are shutdown during the installation.
2. Install the TransactionVision Sensor on all hosts where clustered application servers reside.
3. 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.

   In SensorSetup, for the question to restart the application server, enter **n** to not restart the application server at this point.

   ```
   The WebSphere Application Server dmCell/app1Node/Web1
   where the TransactionVision Sensor has just been
   added must be restarted in order for the
   TransactionVision Sensor to work correctly. Do you
   want to restart this application server? (y/n) [y]: n
   ```
4. 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. On WebSphere 6.*, `<WAS_INSTALL>` refers to your profile directory.

   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 for WebSphere 5.*, 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.

For WebSphere 6.*, copy the following files:

- Copy `<SERVER_PROFILE_DIR>/config/cells/hostcNetwork/nodes/HostA/servers/AppServerA/server.xml` from the machine HostA TO **`<DMGR_PROFILE_DIR/config/cells/hostcNetwork/nodes/HostA/servers/AppServerA/server.xml`**
- Copy `<SERVER_PROFILE_DIR>/config/cells/hostcNetwork/nodes/HostB/servers/AppServerB/server.xml` from the machine HostB TO **`<DMGR_PROFILE_DIR>/config/cells/hostcNetwork/nodes/HostB/servers/AppServerB/server.xml`**

where `SERVER_PROFILE_DIR` is the profile containing the application server installation and `DMGR_PROFILE_DIR` is the profile containing the deployment manager installation.

5. Restart the Deployment Manger.
6. 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>
```

## Enabling Third-Party Instrumentation

Conflicts may arise between TransactionVision instrumentation and instrumentation performed by third-party applications.

### WebSphere Application Server

To allow a third-party product's instrumentation to co-exist with TransactionVision, add the JVM property `com.bristol.tvision.classloader.preprocessor` (or `com.bristol.tvision.classloader.postprocessor`) to point to thethe third-party class which implements the `com.ibm.websphere.classloader.ClassLoaderPlugin` interface.

The class pointed to by `com.bristol.tvision.classloader.preprocessor` is processed before TransactionVision instrumentation. The class pointed to by `com.bristol.tvision.classloader.postprocessor` is processed after the TransactionVision process is complete.

### WebLogic

To allow a third-party product's instrumentation to co-exist with TransactionVision, change the WebLogic startup script to add your preprocessor class to the JVM property `weblogic.classloader.preprocessor`.

Separate multiple class names with a comma. The classes are processed in the order listed. If you want a class preprocessor to process after TransactionVision's, add it after the TransactionVision preprocessor, as in the following example:

```
-
Dweblogic.classloader.preprocessor="com.bristol.tvision.sen
sor.servlet.WEbLogicClassPreProcessor,full.path.to.3rdparty
```

`.preprocessor"`

The third-party class preprocessor should implement the `weblogic.utils.classloaders.ClassPreProcessor` interface.

Chapter 9

# Configuring Sensor Logging

## Log Files

By default, all TransactionVision Sensors 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 z/OS operator console log, or the OS/400 user job log.
- 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 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 Sensor log file `sensor.log`, reaches 10 MB in size, it is renamed `sensor.log.1` and a new `sensor.log` file is created. If you change the configuration so that there are two backup files, the following events take place when `sensor.log` reaches 10 MB:

1. `sensor.log.2` is removed if it exists.

2. `sensor.log.1` is renamed `sensor.log.2`.
3. `sensor.log` is renamed `sensor.log.1`.
4. A new `sensor.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 Sensor is doing internally. It is used mainly to troubleshoot problems and should **not** be turned on in production environments.

You can enable trace logging in TransactionVision Sensors to debug Sensor configuration issues. Trace information for the WebSphere MQ Sensor is logged to the UNIX system log, the Windows event log, the z/OS operator console log, or the 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.

## 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_s
   er verA.properties
   ```

   For serverB, set the JVM property as follows:

   ```
   com.bristol.tvision.sensor.properties=sensor/Sensor_s
   er verB.properties
   ```

   For stand-alone programs, this JVM property is set on the command line when the JVM is invoked, as follows:

   ```
   java -
   Dcom.bristol.tvision.sensor.properties=sensor/Sensor_
   se rverA.properties
   ```

   For WebSphere, set this JVM property using the Administration 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/<custo
   m 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.

# Appendix A
# Utilities Reference

## 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.

## 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 both WebSphere MQ JMS and TIBCO EMS JMS | Instruments those files required for your Java messaging service provider. |
| 2. Instrument Servlet Sensor | Instruments only those files required for the Servlet Sensor. |
| 3. Instrument EJB Sensor | Instruments only those files required for the EJB Sensor. |
| 4. Add JMS Sensors to WebSphere Application Server | Add the JMS Sensor in addition to any other Sensors already installed in the Application Server. |
| 5. Add Only Servlet Sensor to WebSphere Application Server | Adds only the Servlet Sensor to the Application Server. Any other previously installed Sensors are removed from the Application Server. |
| 6. Add Only Servlet and EJB Sensors to WebSphere Application Server | Adds both the Servlet and EJB Sensors to WebSphere Application Server. Any other previously installed Sensors are removed from the Application Server. |
| 7. Add All Sensors to WebSphere Application Server | Adds the Servlet, EJB, and JMS Sensors to WebSphere Application Server. |
| 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. |
| 9. 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. |

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 2 to instrument the Servlet Sensor

- Option 7 to install all Sensors
- Option 9 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 both WebSphere MQ JMS and TIBCO EMS JMS | Instruments those files required for your Java messaging service provider. |
| 2. Instrument Servlet Sensors | Instruments those files required for the Servlet Sensor. |
| 3. Instrument EJB Sensor | Instruments only those files required for the EJB Sensor. |
| 4. Add JMS Sensors to WebLogic Application Server | Add the JMS Sensor in addition to any other Sensors already installed in the Application Server. |
| 5. Add Only Servlet Sensor to WebLogic Application Server | Adds only the Servlet Sensor to the Application Server. Any other previously installed Sensors are removed from the Application Server. |
| 6. Add Only Servlet and EJB Sensors to WebLogic Application Server | Adds both the Servlet and EJB Sensors to the Application Server. Any other previously installed Sensors are removed from the Application Server. |
| 7. Add All Sensors to WebLogic Application Server | Adds the Servlet, EJB, and JMS Sensors to Application Server. |
| 8. Remove All Sensors from WebLogic Application Server | Removes the Servlet, EJB, and JMS Sensors from Application Server. |
| 9. Remove EJB Sensor from WebLogic Application Server | Instructs you to restart the admin server iwth `tvStartWebLogic.cmd`. |

## Syntax:

```
SensorAdmin
```

## SensorSetup

### Location:

```
TVISION_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 or TIBCO EMS
- The installation directory of 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 or the TIBCO EMS server name, host name, port number, user name, and password

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
```

## 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
# Configuration Files

The TransactionVision setup utilities save Sensor 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.

## 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. |
| ConfigurationQMHost | The host name of the configuration queue manager. |
| ConfigurationQMPort | The listener port number of the configuration queue manager. |
| ConfigurationQM-Channel | The channel name of the configuration queue manager. |
| ConnectionRetryDelay | The delay in milliseconds between connection attempts when the connection to the configuration queue manager is lost. The default is 1000. |
| 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. |
| ConfigurationRe-trieveInterval | The time interval in milliseconds to check the configuration queue for new configuration messages. The default value is 10000. |
| 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

## 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-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.

Appendix C

# TransactionVision Changes to WebSphere Application Server

A number of changes are made to WebSphere Application Server during TransactionVision Sensor 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`.

## 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)
- `<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/fscontext.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 D

# Configuring Application Servers for TransactionVision

Typically, you can use the TransactionVision **`SensorSetup`** and **`SensorAdmin`** utilities to configure your application server for the TransactionVision 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 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 the TransactionVision Sensor

### 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/SensorSetupto 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)`
   - `<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/fscontext.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"/>
```

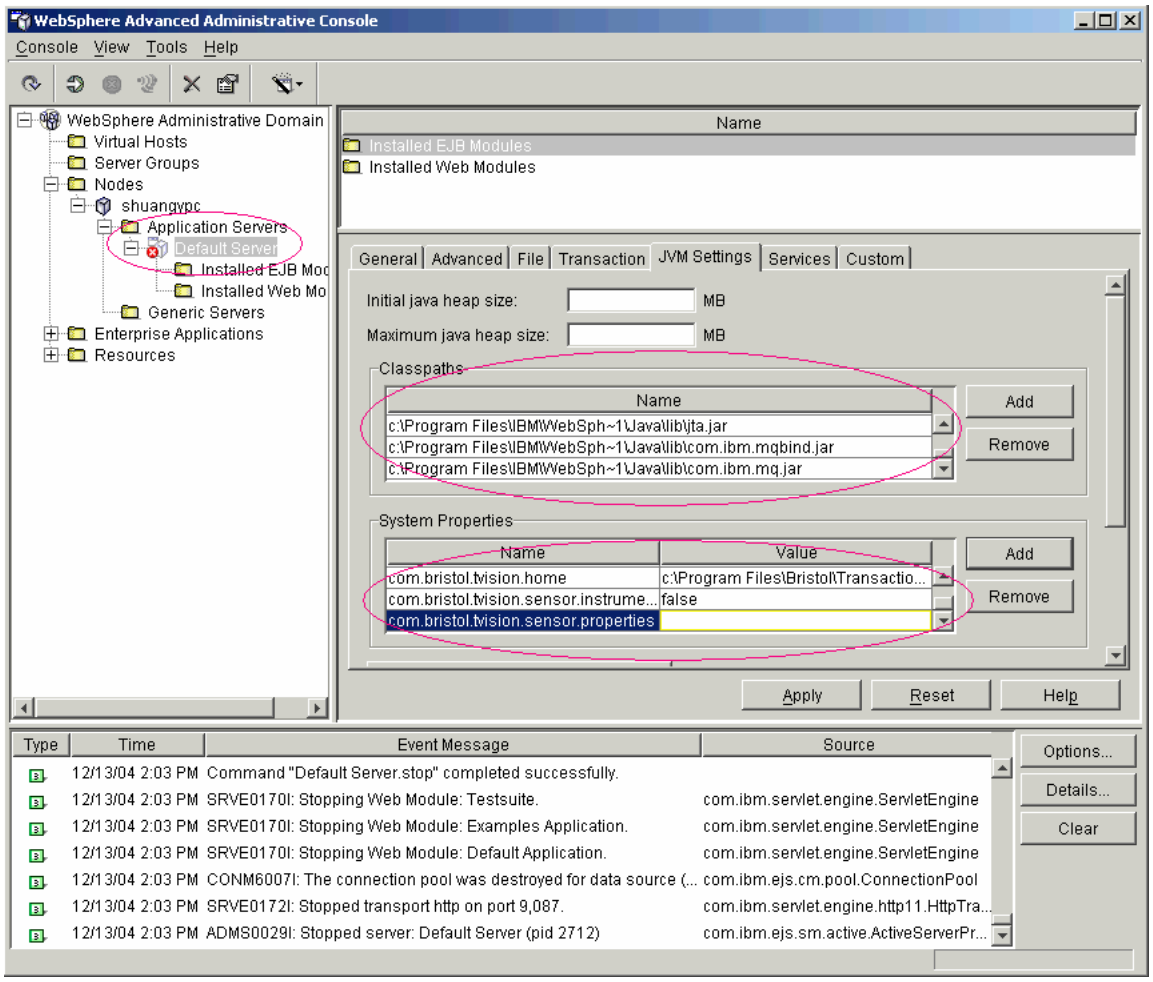
   In “Interceptor X,” the X is a unique number one higher than the existing Interceptor values in your `server.xml` file. For example, Interceptor_18, and Interceptor_19 could be the two names you give if 18 and 19 are not currently in use.
10. Restart the server for settings to take effect.

## 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/SensorSetupto 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:



Initial java heap size: MB

Maximum java heap size: MB

Classpaths

| Name |
|---|
| c:\Program Files\IBM\WebSph~1\Java\lib\jta.jar |
| c:\Program Files\IBM\WebSph~1\Java\lib\com.ibm.mqbind.jar |
| c:\Program Files\IBM\WebSph~1\Java\lib\com.ibm.mq.jar |

System Properties

| Name | Value |
|---|---|
| com.bristol.tvision.home | c:\Program Files\Bristol\Transactio... |
| com.bristol.tvision.sensor.instrume... | false |
| com.bristol.tvision.sensor.properties | |

| Type | Time | Event Message | Source |
|---|---|---|---|
| | 12/13/04 2:03 PM | Command "Default Server.stop" completed successfully. | |
| | 12/13/04 2:03 PM | SRVE0170I: Stopping Web Module: Testsuite. | com.ibm.servlet.engine.ServletEngine |
| | 12/13/04 2:03 PM | SRVE0170I: Stopping Web Module: Examples Application. | com.ibm.servlet.engine.ServletEngine |
| | 12/13/04 2:03 PM | SRVE0170I: Stopping Web Module: Default Application. | com.ibm.servlet.engine.ServletEngine |
| | 12/13/04 2:03 PM | CONM6007I: The connection pool was destroyed for data source (... | com.ibm.ejs.cm.pool.ConnectionPool |
| | 12/13/04 2:03 PM | SRVE0172I: Stopped transport http on port 9,087. | com.ibm.servlet.engine.http11.HttpTra... |
| | 12/13/04 2:03 PM | ADMS0029I: Stopped server: Default Server (pid 2712) | com.ibm.ejs.sm.active.ActiveServerPr... |

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`
- `$WAS_HOME/lib/xerces.jar`
- `$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.instrument.ejb | false |
| com.bristol.tvision.sensor.disableApps | 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 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/SensorSetupto 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 112 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;
<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:
<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 112.

### 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 112 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 112.

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)
<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.tvis
ion.sensor.servlet.WebLogi cClassPreProcessor"
-Dcom.bristol.tvision.sensor.disableApps="Transaction
Vision,console, wl_management_internal1,
wl_management_internal2,uddi,uddiexplorer"
```

6. Click Apply at the end of the page.
7. Restart the server.

## 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.polic
y"
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.polic
y" 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.polic
y" %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.polic
y" %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.polic
y" %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.polic
y" %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.polic
y" %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.polic
y" %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.polic
y" %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.polic
y" %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.poli
cy" 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.polic
y ${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.poli
cy" ${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.poli
cy" ${SERVER_CLASS}  1>&2 >"${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.polic
y ${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.poli
cy" ${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.poli
cy" ${SERVER_CLASS}  1>&2 >"${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.polic
y ${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.poli
cy" ${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.poli
cy" ${SERVER_CLASS}  1>&2 >"${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.polic
y ${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.poli
cy" ${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.poli
cy" ${SERVER_CLASS}  1>&2 >"${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
```

Appendix E

# Additional OS/390 Settings

This appendix lists RACF authorizations, firewall settins, and MIPS requirements for TransactionVision on the OS/390 platform.

## RACF Authorizations

RACF authoriazations are highly customized to the user's environment and hence there are no specific requirements for the TransactionVision Sensors. Use the programs and transactions listed in the following tables to help determine the required RACF authorizations.

### Fro the TransactionVision CICS Sensor

| Transaction | Needs to Run | Needs to Access WebSphere MQ | Needs to Run as a CICS Global User Exit |
|---|---|---|---|
| SLDS | Yes | No | No |
| SLDM | Yes | No | No |
| SLDP | Yes | No | No |
| SLDC | Yes | No | No |
| SLDD | Yes | No | No |

| Transaction Program | Needs to Run | Needs to Access WebSphere MQ | Needs to Run as a CICS Global User Exit |
|---|---|---|---|
| SLDPCSX | Yes | No | No |
| SLDPCMX | Yes | No | No |
| SLDPCCX | Yes | No | No |
| SLDPDSX | Yes | No | No |
| TVISION | Yes | No | No |
| TVISIONC | Yes | Yes | No |

| Exit Program | Needs to Run | Needs to Access WebSphere MQ | Needs to Run as a CICS Global User Exit |
|---|---|---|---|
| SLDPTCX | Yes | No | Yes |
| SLDPPSX | Yes | No | Yes |
| SLDPICX | Yes | No | Yes |
| SLDPTDX | Yes | No | Yes |
| SLDPPCX | Yes | No | Yes |
| SLDPFCX | Yes | No | Yes |
| SLDPTSX | Yes | No | Yes |

### For the TransactionVision CICS WMQ Sensor

| Transaction | Needs to Run | Needs to Access WebSphere MQ | Needs to Enable/Disable CICS Crossing Exit |
|---|---|---|---|
| SLMC | Yes | No | Yes |

| Transaction Program | Needs to Run | Needs to Access WebSphere MQ | Needs to Enable/Disable CICS Crossing Exit | Needs to be Accessed From CICS Crossing Exit |
|---|---|---|---|---|
| CSQCAPX | Yes | Yes | No | Yes |
| SLMC | Yes | Yes | Yes | No |
| SLMBCNFG | Yes | Yes | No | No |

### For the TransactionVision IMS WMQ Sensor

The TransactionVision IMS WMQ Sensor is run from within the IMS WMQ application that is being monitored. This is done by link editing a TransactionVision stub against the IMS WMQ application. Therefore, there may be some RACF authorizations to be performed.

The following libraries are used an may also need to be RACF authorized based on the local RACF authorization scheme.

| Libraries | APF Authorized | In DFHRPL |
|---|---|---|
| SSLDLOAD | No | Yes |
| SSLDAUTH | Yes | No |

## Firewall Settings

Since the TransactionVision Sensors use WebSphere MQ to communicate with the TransactionVision Analyzer, no firewall settings are required. All communication is configured through WebSphere MQ.

## MIPS Required

There are no specific MIPS requirements for the TransactionVision Sensor. Sensors are achitected to be active only when required; even then, they use very little resources to achieve the required functionality.

# Index