BRISTOL TECHNOLOGY®

TransactionVision®

TransactionVision Planning Guide
Version 5.0.0

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

Printed January 13, 2006

This manual supports TransactionVision Release 5.0.0.

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

Part No. TV17060110

Contents

Chapter 1 TransactionVision Overview....1

TransactionVision Basics 1
TransactionVision Terms and Concepts.... 3
Additional TransactionVision Resources 5

Chapter 2 Introduction to TransactionVision Planning....9

Deployment Overview 9
Deployment Planning.... 11
Quick Tips for a Smooth Deployment 13
Assembing the TransactionVision Team.... 15

Chapter 3 Analyzer Host Sizing 17

Define the Scope 17
Sizing.... 19
Selecting TransactionVision Pilot Hardware & Software.... 20

Chapter 4 DBMS Configuration, Sizing, and Tuning 23

Disk Storage Requirements.... 23
DBMS Performance 24
DB2Test 27
OracleTest 27

Chapter 5 WebSphere Configuration....29

Websphere MQ Configuration 29
Websphere Application Server Configuration 29
WebSphere MQ Capacity Planning 30

Chapter 6 TransactionVision Configuration....33

Sensor and Communication Link Configuration.... 33
Analyzer Configuration.... 33
TransactionVision Installation 34

Chapter 7 Performance Testing and Tuning....39
Setting Objectives to Optimize Performance 39
TransactionVision's Process Steps.... 39
Data Collection Filters.... 40
Event Buffering 41
Summary 41
TransactionVision Performance Test 41
TransactionVision Performance Tuning.... 43

Chapter 8 Administration 45
Maintenance and Administration 45
Administration.... 45

Chapter 9 Customizing TransactionVision....47
TransactionVision Customizations.... 47
Custom Reports Flow Chart 48
Custom Report Steps 48

Chapter 1

TransactionVision Overview

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

Graphical analysis of business transactions enable you to:

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

TransactionVision Basics

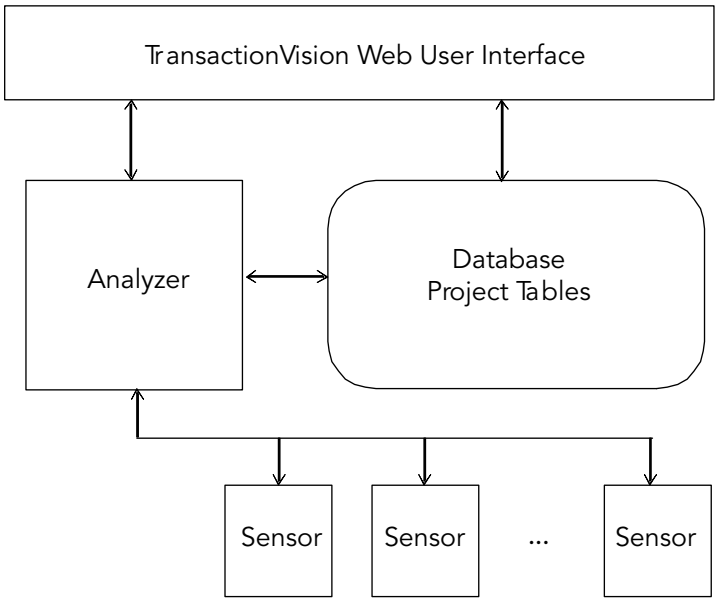
To understand the tasks required to plan you TransactionVision installation, you must understand the TransactionVision components, as well as some basic concepts.

Components

TransactionVision consists of three major components:

- Sensors
- Analyzers
- TransactionVision Web User Interface

The following diagram shows the relationship between these components:



Sensors

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

TransactionVision provides several types of Sensors:

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

For a complete description of TransactionVision Sensors, see the *TransactionVision Sensor Installation and Configuration Guide*.

Analyzer

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

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

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

Web User Interface

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

TransactionVision Terms and Concepts

To administer TransactionVision effectively, you must be familiar with the following terms and concepts:

- Communication links
- Event Collection
- Data collection filters
- Projects
- Database Schemas

Communication Links

Communication links enable a TransactionVision Sensor to communicate from a host on which an application is being monitored to the Analyzer.

Two communication paths must be defined for each communication link: one path for configuration messages from the Analyzer to the Sensors and another path for captured events from the Sensors to the Analyzer. When you define a communication link, you specify the name of the queue manager and queues the Sensor monitors for configuration messages and sends event messages to, as well as the queue managers and queues the Analyzer sends configuration messages to and retrieves event messages from.

Event Collection

The WebSphere MQ Sensor library implements all API entry points and has the same name as the standard library for the monitored technology (for example, `mqm.dll` for WebSphere MQ on Windows). It is installed in a different directory location, and your library search path is altered so that programs load the Sensor library at runtime instead of the standard technology library.

When a program running on the system where a Sensor is installed calls a WebSphere MQ API for the monitored technology, it actually calls the corresponding function in the Sensor library.

The Sensor function first generates a TransactionVision event, recording the API call and other details based on the data collection filters. It then invokes the actual API from the standard technology library. When the actual API returns, the Sensor function generates another TransactionVision event, this time representing the exit state of the function, and then forwards the return information to the calling program.

Important! For WebSphere MQ Sensors to intercept API calls from applications, the applications must link dynamically to the technology library as a shared library. Otherwise, Sensors cannot record events for the applications.

On the z/OS CICS platform, WebSphere MQ Sensors use the API-crossing exit mechanism provided by the CICS adapter of WebSphere MQ for z/OS. On z/OS batch and IMS platforms, applications are statically linked to, or dynamically invoke, TransactionVision stubs, which are replacements for the standard WebSphere MQ stubs.

A WebSphere API Exit Sensor is also available for other platforms.

The WebSphere Application Server servlet, JMS, and EJB Sensors use Java bytecode instrumentation to intercept servlet, JMS, and EJB API calls. At installation of the WebSphere servlet Sensor, the server JVM settings are modified to add a Sensor classloader plugin. This plugin intercepts servlet API calls and allows the TransactionVision Sensor to report them. The JMS API called are trapped using static bytecode

instrumentation and the instrumented jar file must be added to the application CLASSPATH before any other JMS jar file.

Projects

Event collection projects enable you to easily group and manipulate communication links, data collection filters, database schemas, and Analyzers as one entity. An event collection project is used by an Analyzer to define the communication links, the data collection filters, and the database schema that data will be written into. A project is assigned to a single Analyzer host for event analysis.

Data Collection Filters

Data collection filters assigned to a project determine the amount and type of information collected by each Sensor. Data collection filters specify criteria such as the following:

- Which technologies, hosts, programs, or APIs to collect information about
- Which CICS regions, transactions, and job names to collect information about
- Which queues or queue managers to collect information about
- Which servlets, WebSphere applications, WebSphere servers and URIs to collect information about
- What time range to collect information for
- The level of detail to be collected, such as API name only, API name and call arguments, or API name, call arguments, and data buffer segment. For the Servlet Sensor, the default data buffer size is 1K.

Database Schemas

The project schema defines the tables into which the events collected by Sensors are stored. When the Analyzer retrieves and processes events collected by Sensors, it places them into event related tables. By using schemas to partition event data by project, you can control access to event data collected by each project.

Additional TransactionVision Resources

Documentation Roadmap

This guide provides instructions for planning the TransactionVision implementation in your environment. In addition to this guide, the following documents are provided with TransactionVision:

- 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 *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 Sensor Installation and Configuration Guide* provides instructions for installing and configuring TransactionVision Sensors. 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

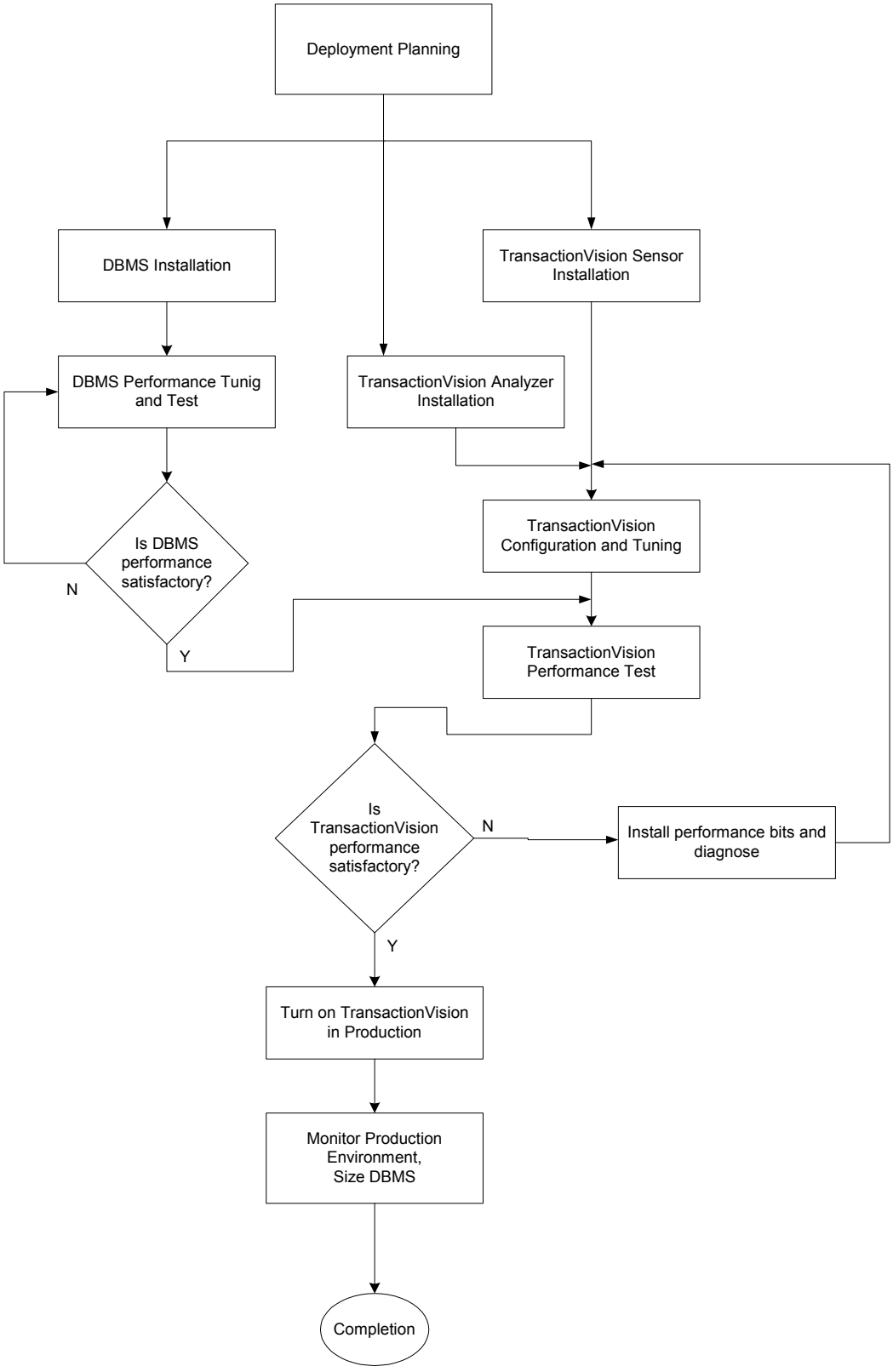
Chapter 2

Introduction to TransactionVision Planning

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

Deployment Overview

The following diagram shows a high level task flow depicting a typical deployment scenario. It provides an overview of the tasks required for a successful TransactionVision deployment. This guide describes each of these tasks in detail.


Deployment Planning
DBMS Installation
TransactionVision Sensor Installation
DBMS Performance Tunig and Test
TransactionVision Analyzer Installation
Is DBMS performance satisfactory?
N
Y
TransactionVision Configuration and Tuning
TransactionVision Performance Test
Is TransactionVision performance satisfactory?
N
Install performance bits and diagnose
Y
Turn on TransactionVision in Production
Monitor Production Environment, Size DBMS
Completion

Deployment Planning

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

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

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

Deployment planning should include collection of the following information:

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

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

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

Identify Servers to Monitor

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

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

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

Use this table to record the number of servers where the TransactionVision Sensor has been installed, which Sensor has been installed (WebSphere MQ, WebSphere Business Integration Message Broker, JMS, Servlet, EJB, CICS), the queue manager(s) the Sensor

connects to, whether a client or server connection is used and if z/OS is the platform, which CICS or IMS region the Sensor is installed under.

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

Dependent Software Installation

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

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

For details on DBMS configuration and tuning, see the *TransactionVision Analyzer Installation and Configuration Guide*.

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

Quick Tips for a Smooth Deployment

When planning your deployment, pay attention to the following tips for TransactionVision and your DBMS.

TransactionVision

1. **Don’t undersize!** Early in your planning process, determine the sizing of processor(s), processor speeds, system memory, log file(s) sizes (especially database transaction log file size) and storage space. Improper sizing could cause moderate to severe issues. Use the TVBlast test before installing TransactionVision on each and every environment (see Chapter 7). Contact Bristol to perform a proper and complete sizing exercise in all these areas with you.
2. **Disable the “Retry Event Package Delivery on Failure” option.** Set the Sensor “put event failure retry” option to an appropriate value according to the event collection strategy. For the safest setting, uncheck the “Retry Event Package Delivery on Failure” option. The option is set on by default to ensure that you don't lose any event messages. The side effect of this is that if you don't plan

your environment properly, it could clog the event queue, fill up queue manager storage space, slow down your application(s), etc. Setting the Retry option off is the safest option. However, unchecking the "Retry Event Package Delivery on Failure" option does leave open the possibility of not always getting every event message.

3. **Configure filters correctly.** Data collection filters assigned to communication links should have only the required MQ objects, application, hosts and APIs specified for collecting. MQGET calls with NO_MSG_AVAIL reason code should be filtered out. When monitoring WMQI brokers, make sure to filter out API calls to WMQI system queues MQGET with NO_MSG_AVAIL reason code.
4. **Make Configuration Message Expiry small.** Configuration Message Expiry should be set to a value small enough so that the amount of events generated by an "orphaned" configuration message can fit into the event queues without causing major production issues.
5. **Set event queue depth/storage correctly.** Event queue depth should be set to a value that matches the queue manager storage spaced based on the average event message size. It should be adequate to handle peak volume. It should not exceed the storage capability of the queue manager host system. You may wish to consider have a new queue manager created for TransactionVision Analyzer. To be certain of storage, plan and execute for maximum event message space. This is important especially when the event queue and TransactionVision dead letter queue are hosted by the production queue manager as running out of storage space may stop the queue manager completely.
6. **Protect the production environment.** A fail-safe method (such as running a crontab job) should be devised to make sure that if any of the TransactionVision components malfunction (such as raw event backlogging) the production environment will be protected.
7. **Configure the Sensor event packaging factor correctly.** Sensor event packaging factor should be set to a value suitable to the environment. This factor is usually set to 10 events per message.
8. **Modify the inetd Configuration file correctly.** Incorrect configuration of inetd is the most common cause for not being able to establish new MQ client connections. Making sure the configuration is correct (including spelling!) avoids this issue.

TransactionVision – z/OS

Enable Language Environment in CICS. The language environment must be enabled in the CICS region for C or the MQ listener will be disabled or possibly MQ connections will be disabled.

Databases

1. **Run database performance tests!** Most databases are tuned for reading without much attention paid to the database's write rate (insert rate). TransactionVision is very insert intensive. Run Bristol's database tests before installing TransactionVision on each and every environment (see Chapter 4).
2. **Set thread count accurately.** Analyzer thread count should be set to match the test results from the DBMS insert rate.
3. **Perform regular database performance maintenance.** DBMS performance maintenance – RUNSTATS (for DB2), or something similar for other databases—needs to be run on a regular basis (recommended daily) to ensure optimal performance.

Assembing the TransactionVision Team

Project Skills & Organization

In order to achieve the project objectives, Bristol Technology proposes forming a project team to evaluate TransactionVision. It is recommended that individuals with skills in the following areas be available during the project to assist with installation and configuration:

- Operating system on which a WebSphere Application Server will be installed
- Operating system on which the TransactionVision Sensors will be installed
- WebSphere MQ network configuration used by the TransactionVision environment
- WebSphere Application Server or other support web server
- IBM DB2 UDB Enterprise Edition or other supported database

The following additional skills may also be needed during installation and configuration:

- Application server technology
- Java Beans

During the evaluation phase, individuals representing the following skill sets should participate:

- Technical Architecture
- Middleware Development & Support
- Application Development & Support
- Business Process Analysis
- ROI Analysis
- Data Security / Information Integrity
- Java Programming
- Project Management

In large-scale projects, Bristol Technology recommends assigning a project manager responsible for the overall completion of the project. Other on-call skills may be required if challenges arise.

The Bristol Technology Team stands ready to supplement and assist your staff concerning any of these required skill sets. Bristol Technology will also provide performance-tuning assistance throughout the project.

Participant Roles

Bristol Technology utilizes a project team distribution list to keep everyone informed of project issues. Please provide the names, roles and contact information for individuals who will participate in the program. Be sure to also include the names of all individuals who must approve funding.

Participant's Role	Name	Title/Position	Phone	E-mail
Applicant				
Project Leader				
Middleware Architect				
Enterprise Architecture				
Production Support				
Application Development				
Data Security				
Database Admin				
WebSphere Developer				
Business Representatives				
Executive Sponsor				
Bristol Account Manager				
Bristol Sales Engineer				
Bristol Support				

Chapter 3
Analyzer Host Sizing

Define the Scope

We have found "scope creep" to be the number one issue facing most technology initiatives. It is therefore critical that scope be well defined at the beginning of the project. Bristol Technology has identified several key questions that facilitate an enhanced understanding and definition of scope.

Scoping Questionnaire

We begin by understanding the platforms and applications that use WebSphere MQ.

Note: You will typically find it easier to complete these tables if they are copied into an Excel spreadsheet.

Complete the following table for all WebSphere MQ applications on distributed platforms:

Application Name	Server Name	Server Mfg/Model	Operating System	# CPUs	Optional List of Queue Managers	Does The App Utilize WMQI? (Y/N)

Complete the following table for all WebSphere MQ applications on mainframe platforms:

Application Name	Mfg/Model	CICS	IMS	OTMA Bridge?	Batch	Is WebSphere MQ Usage Based Pricing (MULC) utilized?	Optional List of Queue Mgrs	Does The App Utilize WMQI? (Y/N)

The information provided in the previous tables represent the total universe of platforms with processes that can be tracked by TransactionVision.

The next step is to understand which set of applications form a business process that can readily be utilized in a pilot. Identify your business processes and their associated applications in the following table:

Business Process Name	Application Name	Server or Mainframe Name

With the potential pilot business processes in mind, please consider the following issues.

- What is your organization's current and planned middleware architecture and how does it compare to the available pilot applications/processes?
- Will MQSI be used in production or during the pilot?
- What is the availability of key staff?
- What role should the business community play?
- What is the end-to-end transaction time today? What message rate should be set as a hurdle? Does the hurdle rate need to be reached

during the pilot? Can the pilot environment support the required message rate?

- Can a few critical platforms be used for the pilot, or must every potential platform be evaluated?
- Can one business process be used for the pilot?
- Does the pilot environment have sufficient HW/SW to conduct the test?
  - Does the pilot environment have the prerequisite software?
  - How many messages must be captured and stored during the test? Does the pilot environment have enough disk space to store the messages?

Final considerations should include the hardware requirements for the pilot. Bristol Technology can provide a hardware sizing recommendation for the pilot based on the information provided in the Sizing section. After carefully considering these issues, you will be prepared to make an informed selection of the pilot business processes.

After a pilot environment is selected, an overall schematic of the application architecture should be drawn and a description of the process should be documented.

Sizing

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

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

Bristol can also provide sizing for the complete roll-out if you provide the information requested in this table regarding the production environment.

In a pilot environment a single instance of the TransactionVision Analyzer is usually deployed. In a production environment, however, deploying several Analyzers might be advantageous. Deploying several Analyzers requires partitioning data. In a partitioned environment, the information requested in **Error! Reference source not found.** should be provided for each data partition.

There are two alternatives for partitioning data.

1. TransactionVision's data connection filters can be used to partition data. In order to take advantage of TransactionVision's filters, events captures by TransactionVision need to contain enough information to enable the data filters to partition data. For example, the MQ header often contains information that can be used to identify specific business applications. Organizations are encouraged to explore this alternative more fully with Bristol Engineering.
2. Alternatively, an organization's system architecture can be used to partition data. Many system architectures require a different message route for each business application. In these environments, data can be partitioned based on the company-wide system architecture.

A key aspect of sizing for the TransactionVision Analyzer is the number of messages per second and the average message size, including the user buffer. The following table is used as one factor to calculate the sizing for your environment.

Hardware environment:	16-way IBM eServer p670 (16 1.1GHz CPUs)
Software environment:	AIX 5.1, DB2 EE 7.1

Benchmarks with 1,000,000 messages containing 1,024 bytes of user data generated this TransactionVision Analyzer throughput model for the IBM p670.

1,259	messages per second
75,000	messages per minute
4,500,000	messages per hour
108,000,000	messages per day

Selecting TransactionVision Pilot Hardware & Software

Determining the appropriate platform(s) for TransactionVision requires an understanding of TransactionVision's major components: the sensors and the analyzer.

Sensors are deployed on platforms running the applications to be monitored. Only one sensor is required for each platform.

The Analyzer is a WebSphere application that collects and analyzes messages and stores them in a RDBMS database.

The hardware requirements for the Analyzer and Web application components increase as the number of messages and users increases. In a limited pilot, all components can usually run on a single Windows or Linux workstation that is fast. A specific sizing recommendation can be

provided based upon the volume information provided in the tables completed during the scope step.

Chapter 4
DBMS Configuration, Sizing, and Tuning

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

Disk Storage Requirements

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

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

(Average message size + 13K Byte) * Event Rate * Number of Collection Points * Event Retention Time = Storage Size

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

((2 + 13)K Bytes * 720,000 * 2 * 4 = 90G Bytes).

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

DBMS Performance

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

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

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

1. Use separate DBMS instances for separate projects – if the data can be partitioned then separate DBMS instances on different hosts can be employed to achieve parallelism. This setup requires setting up multiple instances of TransactionVision.
2. Use RAID disks for table space containers. RAID disks provide parallel I/O at hardware level.
3. Separate table space containers and log file directories. *Log files* (DB2 term, *Rollback Segment* for Oracle) holds uncommitted database operations and usually is highly utilized during database insert/update. For this reason it should have its own container on physically separated disks, and preferably on RAID disks.
4. Stripe table spaces. If parallel I/O is not available at the hardware level, DBMS can be set up to span a tablespace across multiple disks, which introduces parallelism at the DBMS space management level.
5. Large-scale deployment may require an even higher degree of parallelism at the DBMS level, such as using DB2 EEE (Parallel Edition) database.

There are many other database parameters that may impact the performance of TransactionVision. These parameters are listed in the TransactionVision Administrator's Guide and need to be examined one-by-one to make sure they are optimized to the specific DBMS system. Some of these parameters (DB2) are listed below for reference:

- APP_CTL_HEAP_SZ
- MAXAPPLS
- APPLHEAPSZ
- DB2_RR_TO_RS (DB2 variable)

There is also some benefit when the tablespaces used by TransactionVision is managed by the database directly (DMS or DMS RAW tablespaces).

Testing DBMS and Diagnosing Performance bottleneck

Bristol provides an independent tool (*DB2Test*) that can be used to test the performance of DBMS relevant to TransactionVision (especially the record insert rate). The tool is written in Java and should be run where the TransactionVision Analyzer will be installed.

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

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

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

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

1. Disk I/O usage for all involved physical disks (tablespaces and log

files), especially I/O busy percentage.

2. CPU usage, including wait time percentage.

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

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

Database Test

Prerequisites

- Java Runtime Environment 1.3.1 or above
- DBMS server or client installed
- JDBC 2.0 driver is set in the CLASSPATH environment variable. For DB2 the driver is *db2java.zip*. For Oracle the driver is *classes12.jar*.
- A test structure is set up in the database. Use the following SQL statements to create the database structure :

DB2

*CREATE TABLE **SCHEMA_NAME**.RAW_EVENT (event_id INTEGER NOT NULL , event_status INTEGER NOT NULL, event_time TIMESTAMP NOT NULL, commlink_secs INTEGER NOT NULL, commlink_msecs INTEGER NOT NULL, client_secs INTEGER NOT NULL, client_msecs INTEGER NOT NULL, event_data BLOB(10M) NOT NULL, overflow_id INTEGER, CONSTRAINT PK_RAW_EVENT PRIMARY KEY (event_id)) **<IN TABLESPACE>**;*

*CREATE SEQUENCE **SCHEMA_NAME**.SEQ_RAW_EVENT_EVENT_ID;*

Please replace ***SCHEMA_NAME*** with a suitable schema name in the test environment. If not using the default table space USERSPACE, <***IN TABLESPACE***> should be specified with the actual tablespace name.

Oracle

*CREATE USER "**SCHEMA_NAME**" PROFILE "DEFAULT" IDENTIFIED BY "SCHEMA_NAME" DEFAULT TABLESPACE "**USERS**" TEMPORARY TABLESPACE "TEMP" QUOTA UNLIMITED ON USERS ACCOUNT LOCK;*

*GRANT "CONNECT" TO "**SCHEMA_NAME**";*

*CREATE TABLE **SCHEMA_NAME**.RAW_EVENT (event_id INTEGER NOT NULL, event_status INTEGER NOT NULL, event_time DATE NOT NULL,*

commlink_secs INTEGER NOT NULL, commlink_msecs INTEGER NOT NULL, client_secs INTEGER NOT NULL, client_msecs INTEGER NOT NULL, event_data BLOB, overflow_id INTEGER, CONSTRAINT PK_RAW_EVENT PRIMARY KEY (event_id));

*CREATE SEQUENCE **SCHEMA_NAME**.SEQ_RAW_EVENT_EVENT_ID;*

Please replace ***SCHEMA_NAME*** with a suitable schema name in the test environment. If not using the default table space ***USERS***, ***USERS*** should be replaced with the actual tablespace name.

DB2Test

Location:

com.bristol.tvision.admin.DB2Test

Description:

Measures DB2 database INSERT performance.

The utility inserts sample event data into the RAW_EVENT table of the specified schema. Before running the test, create a new project schema with CreateSqlScript (which can be deleted after running the test).

Syntax:

```
DB2Test databaseName user passwd schema eventCount
eventSize threadCount {-commit n}{-jdbcBatch}
```

Options:

Option	Description
`databaseName`	Name of the DB2 Database that contains the schema to be used for the test.
`user`	DB2 user name
`passwd`	DB2 password
`schema`	TransactionVision schema in which sample event data will be saved.
`eventCount`	Number of events to generate.
`threadCount`	Number of threads to use to generate events.
`-commit n`	Executes every *n* insert statements as part of a single batch operation. Default is to commit each insert individually.
`-jdbcBatch`	Use JDBC standard batching.

OracleTest

Location:

com.bristol.tvision.admin.OracleTest

Description:

Measures Oracle database INSERT performance.

The utility inserts sample event data into the RAW_EVENT table of the specified schema. Before running the test, create a new project schema with CreateSqlScript (which can be deleted after running the test).

Syntax:

```
OracleTest databaseName host port user passwd schema
eventCount eventSize threadCount [-VARCHAR | -BLOB |
-LONGRAW] {-commit n} {-jdbcBatch} {-OracleBatch} {-thin}
{-parallel} {-url URL}
```

Options:

Option	Description
`databaseName`	Name of the Oracle database that contains the schema to be used for the test.
`host`	Name of host system on which Oracle server exists
`user`	Oracle user name
`passwd`	Oracle password
`schema`	TransactionVision schema in which sample event data will be saved.
`eventCount`	Number of events to generate.
`eventSize`	Size of event user data buffer (default is 1024 bytes)
`threadCount`	Number of threads to use to generate events.
`-VARCHAR`	Use this option if the RAW_EVENT table has been created with a VARCHAR column definition.
`-BLOB`	Use this option if the RAW_EVENT table has been created with a BLOB column definition.
`-LONGRAW`	Use this option if the RAW_EVENT table has been created with a LONGRAW column definition.
`-commit n`	Executes every *n* insert statements as part of a single batch operation. Default is to commit each insert individually.
`-jdbcBatch`	Use JDBC standard batching.
`-OracleBatch`	Use Oracle update batching.
`-thin`	User thin client driver. Default is to use oci client driver.
`-parallel`	Use the Oracle INSERT PARALLEL option instead of the standard INSERT INTO.
`-url URL`	By default, OracleTest will use an appropriate JDBC URL for thin or oci client drivers. However the default may be overwridden by specifying the JDBC URL here.

Chapter 5
WebSphere Configuration

Websphere MQ Configuration

The following needs to be checked for the WMQ objects created for TransactionVision:

1. Event queue Queue manager. It is best to have the actual event queue hosted on a queue manager other than the production queue manager, so that event queue related issues (such as running out of disk storage due to event backlog) will not affect normal production functions.
2. Event queue depth should be set to a value that matches the queue manager storage space based on the average event message size. It should be adequate to handle the peak volume.
3. Event queue storage should match the event queue depth based on the average event message size. It should not exceed the storage capability of the queue manager host system. This is important especially when the event queue and TransactionVision dead letter queue are hosted by the production queue manager as running out of storage space may stop the queue manager completely.
4. Event queue message persistency property should match the event collecting policy.

Websphere Application Server Configuration

The Websphere Application Server should be hosted on a host system with enough processing power, especially if reports are going to be run for TransactionVision.

The Websphere Application Server should also have good access to the DBMS server. Running Websphere Application Server on a system with a

slow link to the DBMS will result in very poor response time for the end user.

WebSphere MQ Capacity Planning

This chapter provides a methodology for determining how TransactionVision will impact the number of messages handled by a given system. Then, these adjusted numbers can be used with the information in the *MQSeries Planning Guide* (CSQZAB03) to calculate storage needs, etc.

Calculating TransactionVision's Impact on Message Volume

Use the following formula to determine the impact of TransactionVision monitoring on message volume:

impact factor = 1 + (*application efficiency factor* / *event packaging number*)

Variable	Description
impact factor	The impact of TransactionVision monitoring on message volume. This number can be multiplied times the existing message volume to show how much message volume will be increased with TransactionVision monitoring. Note that when TransactionVision is not actively being used to monitor a system, there is no impact on message volume, so this factor becomes one.
application efficiency factor	A number from 1 to 5 approximating how efficiently the applications use the WebSphere MQ APIs. It is the ratio of the number of WebSphere MQ APIs called to the number of messages handled. In the worst case scenario, each MQPUT or MQGET will be accompanied by an MQCONN, MQOPEN, MQCLOSE and MQDISC. In this case, the efficiency of the application would be 5, since approximately five APIs are called for every message processed. At the other end of the spectrum, if an application were to call MQCONN, MQOPEN and then MQPUT 1,000 messages and MQCLOSE, MQDISC, the total number of APIs is 1,004. Dividing this by the number of messages processed (1,000) gives us a ratio of 1.004. Estimating this ratio is necessary to determine how many events TransactionVision will generate, and hence how many event messages TransactionVision will send.

Variable	Description
event packaging number	The parameter set in the data collection filter telling the Sensor to pack a certain number of events into one event message. Note that if event packaging is not enabled, then this factor has a value of 1.

This impact factor may have to be calculated independently for different applications which are being monitored, since these applications may have different efficiency factors.

Let's consider an example. Suppose TransactionVision is being used to monitor an application running in batch mode overnight. This application runs on z/OS and reads data from some internal datasets and pushes that data out using messages to a variety of other servers. On average the application sends out 2,500 messages spread evenly over 25 different queues. The application is written efficiently - there is only one MQCONN and MQDISC needed and then an MQOPEN/MQCLOSE pair for each queue. Each batch of 1,000 messages is also committed or rolled back using MQCMIT or MQBACK.

An efficiency rating for this application can be calculated as follows:

2,500 (msgs) + 2 (MQCONN/MQDISC) + 50 (MQOPEN/MQCLOSE) + 25 (MQCMIT) = 2577 apis

2,577 apis / 2,500 msgs = 1.03 *application efficiency factor*

Now, we can apply the *application efficiency factor* to the number of messages sent and calculate different *impact factors* based how we set the event packaging.

Event Packaging Number	Impact Factor
1	`1+(1.03/1) = 2.03`
10	`1+(1.03/10) = 1.103`
25	`1+(1.03/25) = 1.0412`

Since this is a very efficient application, the TransactionVision impact can be minimized quite well with a reasonable packaging number. Simply multiply the *impact factor* times the number of messages processed, 2,500 in this case, and we find that the total message traffic has increased to approximately 2,603 using the a packaging number of 25.

The *impact factor* can be used in the standard capacity planning formulae in the *MQSeries Capacity Planning Guide* which depend on either total messages or messages/second figures. Simply multiply the *impact factor* times the current messages or messages/second figure to get a new result which takes into account the increased traffic due to TransactionVision.

Note that the use of this factor only accounts for increased message volume, but does not account for the message size.

Message Size

Message size has an impact on the TransactionVision data. If messages are small, only several thousand bytes for example, then the relative size of API overhead (message descriptors, put message options, etc.) tracked by TransactionVision is large. If the messages themselves are large, however, the relative size of overhead is small, but all of the content of the messages is being duplicated. In cases where very large messages (greater than 500K, for example) are sent, it may be desirable to use data ranges in the data collection filter to have TransactionVision collect only a portion of the data. This minimizes strain on the storage requirements of the WebSphere MQ system.

Data Collection Filters

The TransactionVision impact formula presented in this appendix considers the scenario where the broadest data collection filter is applied—every WebSphere MQ API is captured. The impact factor can be drastically reduced by defining more specific data collection criteria. For example, a simple data collection filter might capture only MQGET and MQPUT1 APIs. In this case, the *application efficiency factor* will be reduced to 1, since no extraneous APIs will be captured.

Since the data collection filter allows for great flexibility, it is difficult to create a formula to calculate the effect of a given data collection filter. Instead, the *impact factor* may be considered a “worst case" scenario which may be reduced through the use of appropriate data collection filters.

Chapter 6
TransactionVision Configuration

Sensor and Communication Link Configuration

1. Data collection filters assigned to communication links should have only the required MQ objects, applications, hosts and APIs specified for collecting. MQGET calls with NO_MSG_AVAIL reason code in general should be filtered out.
2. When monitoring WMQI brokers, make sure to filter out API calls to WMQI system queues MQGET with NO_MSG_AVAIL reason code.
3. Sensor event packaging factor should be set to a value suitable to the environment. This factor usually is set to 10 events per message.
4. Sensor “put event failure retry” should be set to an appropriate value according to the event collecting strategy. For the safest setting, uncheck the “Retry Event Package Delivery on Failure” option.
5. Thread count assigned to each communication link should match the test results from the DBMS insert test.

Analyzer Configuration

The following parameters should be checked for the Analyzer:

1. Make sure the DBMS schema has been tuned up and tested (see Chapter 4).
2. Configuration Message Expiry should be set to a value small enough so that the amount of events generated by an “orphaned”

configuration message can fit into the event queues without causing major production issues.

3. Analyzer thread count should be set to match the test results from the DBMS insert test.

TransactionVision Installation

Sensor Installation

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

Analyzer Installation

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

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

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

There are two alternatives for partitioning data.

- TransactionVision's data collection filters can be used to partition data. In order to take advantage of TransactionVision's filters, events captured by TransactionVision need to contain enough

information to enable the data filters to partition data. For example, the WebSphere MQ header often contains information that can be used to identify specific business applications. Refer to the *TransactionVision Administrator's Guide* for details on using data collection filters and communication links.

- Alternatively, an organization's system architecture can be used to partition data. Many system architectures require a different message route for each business application. In these environments, data can be partitioned based on the company-wide system architecture.

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

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

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

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

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

Web User Interface Installation

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

- It has sufficient processing power to support the WebSphere or WebLogic application server and the number of users that are

expected to use TransactionVision. Also, consider whether any report or analysis jobs are being planned to be deployed.

- All prerequisites are met. Refer to the release notes for required software versions.
- The machine has network access to the machine running the TransactionVision Analyzer. Many times it may be sufficient to run the web user interface and Analyzer on the same machine.
- You must have administrative access to the machine and WebSphere or WebLogic.

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

TransactionVision Setup

The *TransactionVision Installation and Configuration Guides* walk you through the process of setting up the TransactionVision product. This involves enabling the Sensors and setting up database access from the Analyzer and web component.

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

Communication Link Setup

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

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

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

complete the following tables to define our communication links.

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

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

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

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

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

User Access

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

The users of TransactionVision are as follows:

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

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

Chapter 7

Performance Testing and Tuning

This section discusses strategies to minimize the overhead on transactional performance due to the installation and active operation of TransactionVision. When appropriately configured, TransactionVision should enable you to collect all the required analytical information about your systems and still meet your transactional service levels.

Setting Objectives to Optimize Performance

Most organizations establish service level standards for a complete, end-to-end-business process. This business process can include application logic, information retrieval from one or more data sources, database inserts, deletes and updates, data transformation and middleware messaging. The contribution of any one of these steps toward the end-to-end business process performance depends on the overall business process design. For example, a process that is solely responsible for routing messages will rely significantly more on middleware performance than a business process that needs to access a database and performs application logic.

Our experience indicates that the vast majority of business processes do more than just flow through middleware – they also usually involve accessing data, performing application logic and often data transformation. Therefore, performance overhead should be evaluated based on the entire end-to-end business process. Once you have identified the end-to-end business process that you need to optimize, the performance overhead of TransactionVision can be evaluated and optimized.

TransactionVision's Process Steps

When actively collecting data, TransactionVision performs three (3) tasks after an application makes a WebSphere MQ API call. Specifically, TransactionVision will:

1. Inspect the content of the call against the data collection filter criteria
2. If the message matches the filter, TransactionVision makes a copy of the message content and calls MQPUT to send the event asynchronously back to the TransactionVision database
3. Pass the complete message to the WebSphere MQ library

After passing the call to the MQ library, the application resumes its normal flow by executing the actual API call.

Based on this sequence of events, there are two main factors that dictate the performance impact of TransactionVision: Data Collection Filters and Event Buffering.

Data Collection Filters

The Data Collection Filters determine which messages and how much of each message is saved to the database. Data Collection Filters can be set based on several parameters including: Host Name, User Name, Program Name, Time, CICS Region, CICS Transaction, Job Name, API, API Return Code, QueueManager, MQSI Broker, MQSI Message Flow, IMS Region Type, IMS Region Identifier, IMS Identifier, IMS Transaction, and IMS PBS. By tuning these parameters, filters that limit the volume of messages collected by TransactionVision are created.

The data collection filter parameters that most significantly affect system performance are related to the MQ series User Data Buffer section. TransactionVision can be configured to collect API names only, API name plus API header information or API, Header and User buffer.

Parameters can also be set to collect only critical sections of the user buffer. For example parameters can:

- Set the Data Collection Filters to collect only APIs with failure or warning return codes.
- Set Data Collection Filters to collect only the first 100 bytes of each message user buffer.

TransactionVision can even be configured to measure and identify its own performance bottlenecks. For example, TransactionVision performance can be evaluated through a post-collection java bean that measures end-to-end transaction performance at the end-point of a request/reply system. If performance exceeds the desired service level, additional sensors can be activated automatically to collect all the data that will help determine the source of the problem.

Event Buffering

An additional performance tuning mechanism provided by TransactionVision is event buffering. By default TransactionVision calls MQPUT each time a message matches a Data Collection Filter. The Event Packaging parameter buffers multiple messages at the sensor and packages them into a single MQPUT call. For example, event buffering can enable TransactionVision to send one TransactionVision message for every ten application messages.

Summary

By utilizing the data collection, configuration and operational strategies discussed above, you should be able to collect all required information about your systems and meet your transactional service level objectives. Additionally, you may find that your new end-to-end transaction analysis capabilities provided by TransactionVision more than justify any incremental computer resources and capacity utilized. Bristol's Global Services organization stands ready to assist customers with setup and configuration to enable you to achieve maximum value and performance.

TransactionVision Performance Test

Prior to activating TransactionVision, it is recommended to run the test application (**tvblast**) provided by Bristol to verify that the underlying infrastructure that is running TransactionVision can handle the volume generated by the production applications.

This test should be run in the same environment (or a mirrored environment) where the production application will run. The test result should be used to check whether the system has been tuned up to the same level of similarly configured systems.

The test should be run to generate events with the same size expected in production, and at a similar event rate. The whole event collecting and analyzing path will be tested this way. During the test, system resources across all involved platforms should be monitored for any potential issues.

Prerequisites

- WMQ server installed
- Sensor installed on the test platform (supported platforms are Windows NT 4.0/2000, Solaris, AIX and HP-UX)
- TransactionVision Analyzer installed and configured for collecting events from the test platform
- One available queue defined at the queue manager to be tested

Usage

PUT format

```
tvblast –c count –b message_size  –W wait_interval -t
trace_level -p print_iteration -u unit_of_work -s
status_report queue_manager queue
```

Where:

count	number of total messages to write
message_size	size of message user buffer in bytes (default is
1000)	
wait_interval	amount of time in millisecond to wait before proceed to the next message
trace_level	control how detailed the trace information is. Usually set to 2.
print_iteration	control how often (in messages) the trace message is given
unit_of_work	control how many messages are included for each unit of work
status_report	control how often (in messages) a statistics report is generated
queue_manager	the name of the queue manager the program connects to
queue	the name of the queue that is to be used for the test

GET format

```
tvblast –c count –b buffer –r -C –W wait_interval -t
trace_level -p print_iteration -u unit_of_work -s
status_report queue_manager queue
```

Where:

count	number of total messages to read
buffer	size of the receiving buffer
wait_interval	amount of time in millisecond to wait before proceed to the next message
trace_level	control how detailed the trace information is. Usually set to 2.
print_iteration	control how often (in messages) the trace message is given
unit_of_work	control how many messages are included for each unit of work
status_report	control how often (in messages) a statistics report is generated
queue_manager	the name of the queue manager the program connects to
queue	the name of the queue that is to be used for the test

* *tvblast -x* shows explanation of all options.

Examples

If the production environment has a transaction rate of 5 transactions/second with two data collecting points (MQPUT and MQGET) and average message buffer size is 2000 bytes. The following

test can simulate the production scenario in a 10-minute test using two tvblast instances for writing and reading:

Message rate: 5 message/sec for each tvblast instance. Set the message wait interval to 200ms (*-W 200*).

Message size: 2000 bytes. Set the buffer size to 2000 (*-b 2000*).
Test duration: 10 minutes. Set the total message count to 3000 messages (*-c 3000*).
Unit of work: assuming each message is in its own unit of work (*-u 1*).

tvblast -c 3000 -b2000 -t 2 -p 50 -u 1 -W 200 -s 300 test_manager test_queue
tvblast -c 3000 -b2000 -C -r -t 2 -p 50 -u 1 -s 300 test_manager test_queue

Note: The two tests should be run simultaneously. The data collection filter should be set to collecting MQPUT and MQGET in general.

TransactionVision Performance Tuning

If the performance test results show that the system does not perform as expected, further investigation is required to identify the bottleneck.

The first place to look is the performance data from the test, particularly the system resource utilization data. There may be resource bottlenecks (CPU utilization on the Analyzer hosts, disk I/O utilization on the DBMS hosts, etc.) that can be immediately recognized from this data.

If there is no obvious source for the under performance, then a diagnostic test needs to be performed to break down the event processing time. A TransactionVision build with the performance diagnosis function enabled needs to be installed and the performance test needs to be repeated. TransactionVision will generate detailed performance information that is helpful in pinpointing the cause of the performance issue.

Chapter 8
Administration

Maintenance and Administration

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

- DBMS storage space management - an effective mechanism needs to be devised to control the storage space used by TransactionVision data. This may include a regular data backup and deletion schedule. A policy needs to be in place to decide what data needs to be backed up. This can be decided based on what the data is being used for. While every scenario may be different, some examples might be to save complete event data which may be used for short term diagnostics and may be preserved for just a week. Alternatively, message data may be used for auditing and may need to be preserved for years. Statistical data may need to be preserved for a long period to observe trends in volume and response time.
- DBMS performance maintenance - RUNSTAT (for DB2) needs to be run on a regular basis to ensure optimal performance.
- Adding and changing projects, schemas, users, queries, filters, Sensors and communication links. Refer to the *TransactionVision Administristration Guide* for details on these tasks.

Administration

The following tasks need to be planned for TransactionVision administration:

1. DBMS storage space management – an effective mechanism needs to be devised to control the storage space used by TransactionVision data.

2. DBMS performance maintenance – RUNSTAT (for DB2) needs to be run on a regular basis (recommended daily) to ensure optimal performance.
3. TransactionVision runtime environment itself needs to be monitored to make sure things like TransactionVision log files do not run out of space
4. A fail-safe method (such as running a crontab job) should be devised to make sure that if any of the TransactionVision components malfunction (such as raw event backlogging) the production environment will be protected.

Chapter 9

Customizing TransactionVision

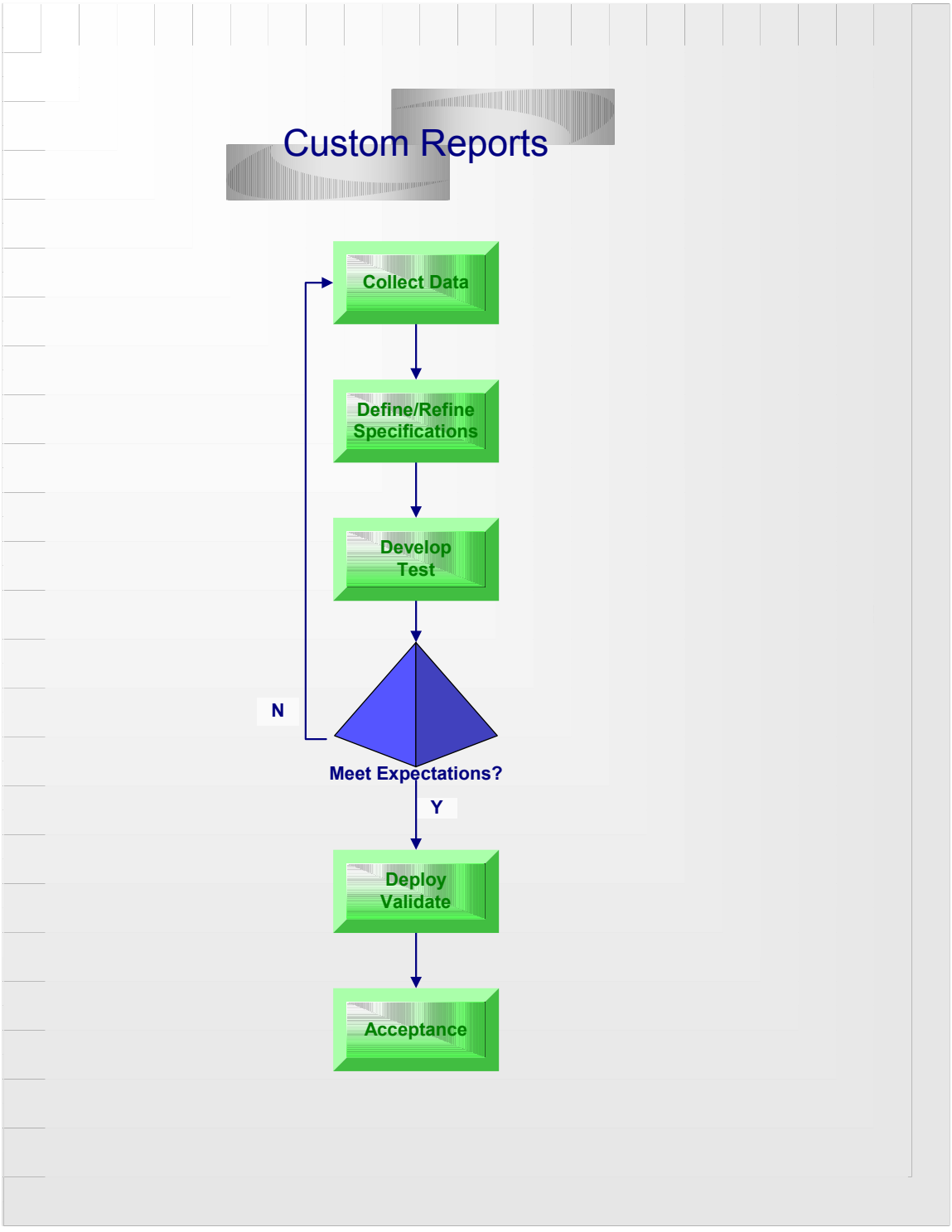
TransactionVision Customizations

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

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

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

Custom Reports Flow Chart



Custom Report Steps

The process of creating custom reports generally consists of the following steps:

1. **Collect TransactionVision data from customer environment.** This step usually involves installing, configuring and running TransactionVision in some customer environment. The customer environment can be development environment, QA environment or production environment, as long as the actual (or mirrored) transaction flow can be captured by TransactionVision. The captured data can be used as the base for demonstrating TransactionVision capability, developing requirement for custom reports and later testing the reports. This step may be done during the Proof-of-Concept stage.
2. **Define/Refine custom report requirement specification**. Once customer has better understanding of the data collected by TransactionVision and how the data can be used to solve technical and business issues, a specification shall be developed by the customer to formalize the requirement for the custom reports. The specification should cover the logic for generating the reports, detailed user interface description (possibly with screen mock-ups or samples) if applicable, the manner the reports will be invoked (scheduled, interactive, etc) and response time and any relevant topics. Bristol will provide an estimate for the amount of work that is involved to implement the reports.
3. **Develop and test custom reports.** Once the specification is finalized, work can start to implement the logic to create the reports. The work will be conducted at Bristol. The work may involve both front end (user interface) and back end (data analysis) programming. Bristol will conduct the development in house and test the reports against the data collected from customer environment.
4. **Review reports.** Once the reports have been tested, the customer will have an opportunity to review the reports. If the reports does not meet all customer expectation, or based on the reports the customer has developed additional requirement, we can go back to step 2 to refine the requirement specification.
5. **Deploy and validate report.** Once the customer has approved the reports, they can be deployed to the customer environment for final validation. Sometimes re-configuration of the reports is necessary if the final deployment environment is different from the environment where the initial test data were collected. A user acceptance test usually is conducted after the deployment to make sure that both the content of the reports and the response time of the reports are acceptable to the customer.