BRISTOL TECHNOLOGY®

TransactionVision®

Security

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.

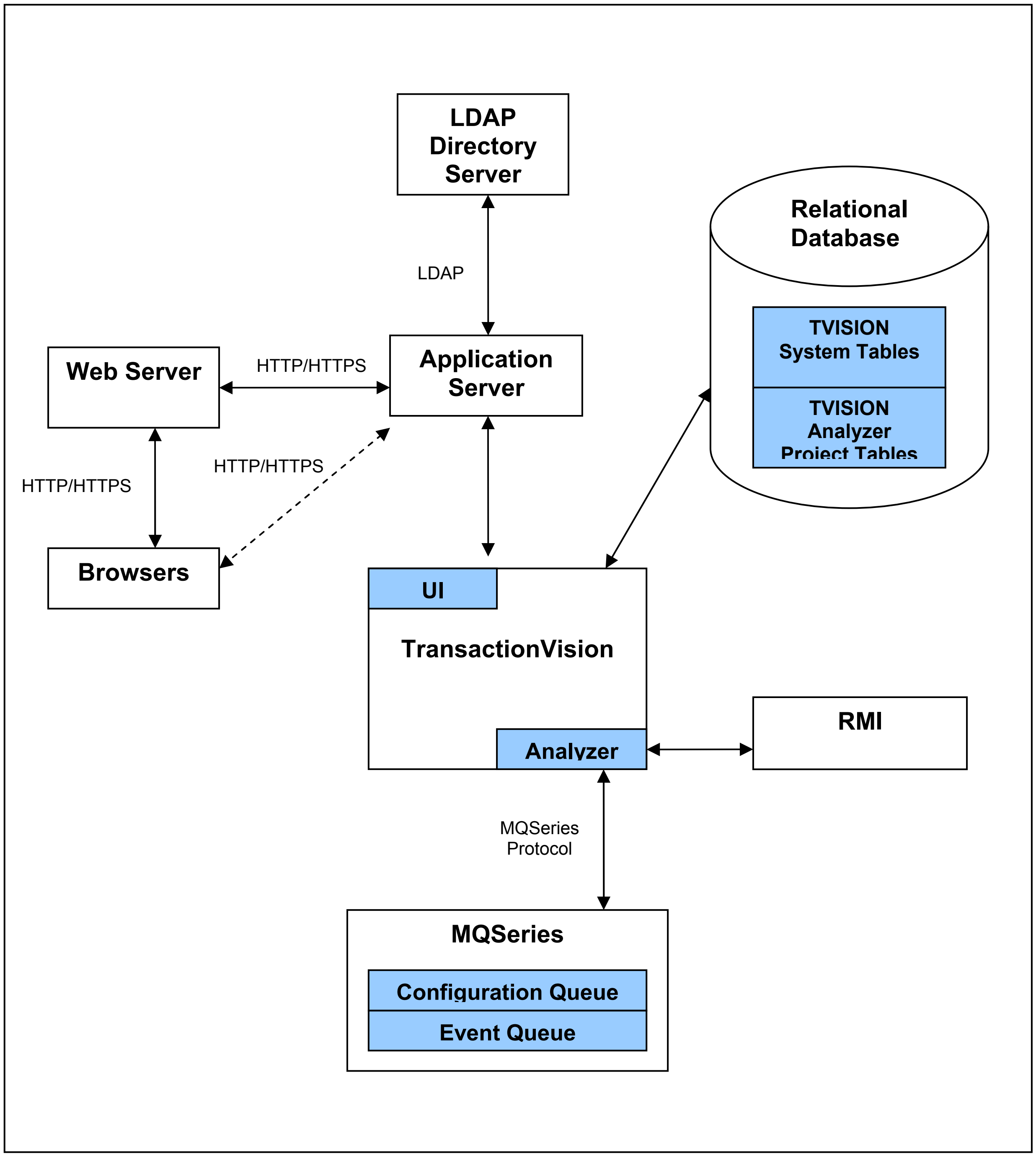
Part No. TV18060110

Contents

Introduction

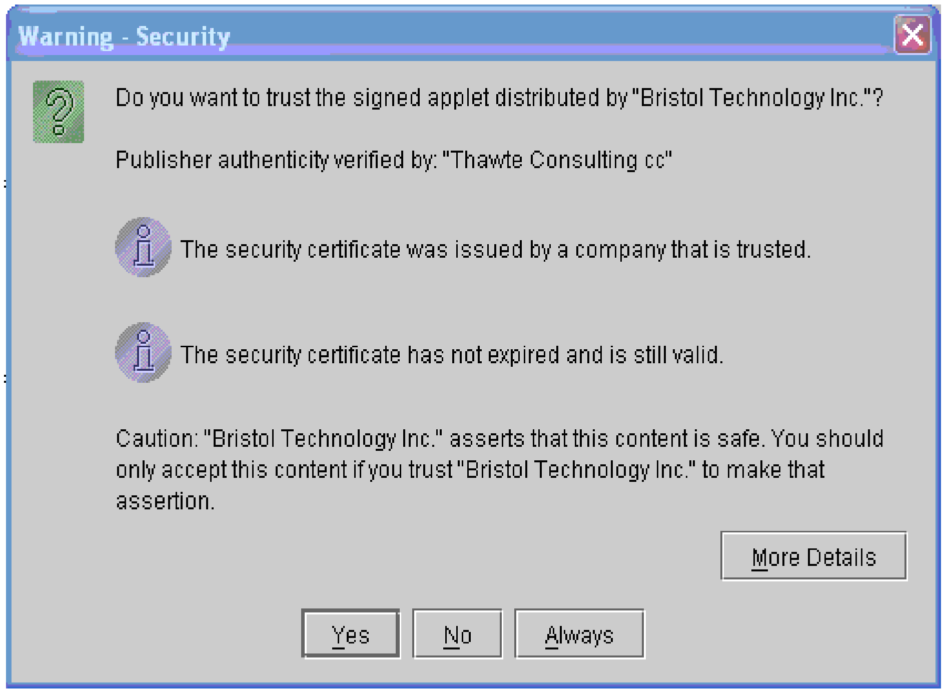
This document provides an overview of the security features and setup procedures of TransactionVision. These features and procedures ensure that data collected by TransactionVision is secure and accessible to the appropriate people. This document assumes that IBM HTTP Server, IBM WebSphere Application Server, IBM DB2, and IBM MQSeries are being used with TransactionVision. This document also contains general notes on these product security related issues. Make sure you refer to the respective product manuals for detailed information.

The following figure shows the relationship between TransactionVision components and other required software components.



Java Security Settings and TransactionVision Applets

TransactionVision's web component requires the ability to download and run Java applets. All TransactionVision applet JAR files are digitally signed with a root certificate from the Certificate Authority (Thawte). TransactionVision applets use signed certificates to insure execution of only trusted code. The first time a user opens up one of TransactionVision's applet views, they will see a security window opened by the browser asking if the user trusts the signed applet specified. Once an applet is verified from a trusted source, the applet can then be granted permission to perform operations that might ordinarily be forbidden. For example, this allows the TransactionVision applet to print, and copy to or receive things from the system clipboard, among other things.


Warning - Security

Do you want to trust the signed applet distributed by "Bristol Technology Inc."?

Publisher authenticity verified by: "Thawte Consulting cc"

The security certificate was issued by a company that is trusted.

The security certificate has not expired and is still valid.

Caution: "Bristol Technology Inc." asserts that this content is safe. You should only accept this content if you trust "Bristol Technology Inc." to make that assertion.

Advanced Java Security Configuration

The default behavior of the Java Plugin is to either run the applet as a trusted applet, or an un-trusted applet. To be run as a trusted applet, the user confirms the he trusts the signed applet as shown in the aforementioned dialog. When this is done, the plugin accords the trusted applet any permission that an executable running on that client machine would have. Depending on the situation, if more stringent security policies are required, the Java Plugin can be configured to give even trusted applets additional security restrictions. In most situations this should not be needed, but if such requirements are in place they can be configured as described below.

Additional security restrictions can be specified through your java.policy file located on a client machine. This file typically resides in `<Java InstallationDirectory>\jre\lib\security\java.policy`. This file can be customized to allow or deny access to a variety of system resources. The exact detail of all these permissions and their implications is outside the scope of this document, detailed information on the specifics can be found in `http://java.sun.com/j2se/1.3/docs/guide/security/permissions.html`. By default, signed applets do not use the policy file, even if present. In order to indicate that the Java Plugin should use of the local policy file, you must add a "usePolicy" entry in the policy file. For example the following policy file would override the default behavior of granting a trusted applet all permissions, and would only grant it whatever permissions are specified in the list.

```
grant {
           permission java.lang.RuntimePermission "usePolicy";
           ... additional permissions to grant ...

     }
```

Note that care should be taken in setting these permissions, as they can in some cases limit the functionality that the TransactionVision applet provides or prevent the applet from being able function at all.

Cookies

TransactionVision uses the J2EE session APIs to manage sessions and does not explicitly use cookies. The WebSphere Application Server can be setup to manage session id with the use of cookies. TransactionVision provides the option (non default) to save the user id and password and encrypted cookies. You can do this by editing `<TVISION_HOME>\config\ui\UI.properties`, and set the autologin flag to be true (i.e., autologin=true).

Client Communications Security

TransactionVision users access the TransactionVision web client via a web browser using an HTTP connection. There are many aspects to communication security, designed to protect client communication on a logical level to prevent unauthorized access to the server.

HTTPS

Complex security systems use many different protocols and algorithms for user authentication and data encryption. The lowest level security involves encryption standards, which are based on very complex algorithms, such as the RSA algorithms. The security protocols such as Secure Socket Layer (SSL) are one level higher.

The Secure Socket Layer (SSL) transfer protocol guarantees security during communication. When used with HTTP, the extensively accepted SSL secures the connection between the client and the server. One of the features of the SSL (HTTPS) transport mechanism is the reliance placed upon digital certificates. The use of such certificates provides the authentication and encryption required between peers over a secure connection. By default, WebSphere ships with a set of generic SSL certificates that allow global security to be configured and enabled quickly.

WebSphere HTTPS Setup

The TransactionVision web ui will run in either a HTTP or secure HTTPS environment, there is no need for any special configuration on TransactionVision's part. You can see what ports and determine where SSL is enabled through the WebSphere admin console by going to Application Servers-><your server>-> Web Container-> HTTP Transport. By default WebSphere runs a secure HTTPS connection on port 9443, with a pre-made certificate.   To generate the your own certificate, and further configure your WebSphere security environment refer to Chapter 10, "Administering WebSphere Security," in the IBM WebSphere Security Redbook (`http://www.redbooks.ibm.com/redbooks/pdfs/sg246573.pdf`).

WebLogic HTTPS Setup

WebLogic Server fully supports SSL communication. By default, WebLogic Server is configured for one-way SSL authentication. Using the Administration Console, you can configure WebLogic Server for two-way SSL authentication.

To use SSL when connecting to a WebLogic server application with your browser, you simply specify HTTPS and the secure port (default port number 7002) in the URL.

For example: https://*localhost*:7002/tv

Where *localhost* is the name of the system hosting the Web application.

WebLogic's online document http://e-docs.bea.com/wls/docs81/secintro/index.html and http://e-docs.bea.com/wls/docs81/secmanage/index.html provides detailed instructions for the setup.

TransactionVision Configuration Files Settings

All the TransactionVision configuration files are located under the `<TVISION_HOME>/config` directory. When the `<TVISION_HOME>/bin/` **TVisionSetupInfo.[sh|bat]** script is run, the location of the logs directory will be specified, where the TransactionVision log files will be found.

The TransactionVision application administrator, who should be the only person who has write access to the files, should own all TransactionVision configuration files. Generally speaking, only read permission should be granted to all the relevant TransactionVision groups.

The following table shows the suggested authorities for TransactionVision configuration files:

Directory	Transaction -Vision Web Application	Transaction -Vision Analyzer	Description
<TVISION_HOME>/config/codepage/*	read	read	Codepage configuration
<TVISION_HOME>/config/datamgr/*	read	read	Database connection configuration
<TVISION_HOME>/config/job/*	read	none	Job manager configuration
<TVISION_HOME>/config/ldap/*	read	none	Ldap connection configuration
<TVISION_HOME>/config/license/*	read	read	TransactionVision license
<TVISION_HOME>/config/logging/*	read	read	Logging policy
<TVISION_HOME>/config/popchart/*	read	none	Report display
<TVISION_HOME>/config/rmi/*	none	read	RMI policy
<TVISION_HOME>/config/services/*	none	read	Analyzer configuration
<TVISION_HOME>/config/setup/*	read	read	TransactionVision Setup configuration
<TVISION_HOME>/config/technologyconst/*	read	none	Various constant definitions
<TVISION_HOME>/config/typeconversion/*	read	none	Event details type conversion configuration
<TVISION_HOME>/config/ui/*	read	none	User interface configuration, including reports settings
<TVISION_HOME>/config/usermgr/*	read	none	User data configuration
<TVISION_HOME>/config/xdm/*	read	read	Various xdm definitions
<TVISION_HOME>/config/xmlschema/*	read	read	Various xsl definitions

Directory	Transaction -Vision Web Application	Transaction -Vision Analyzer	Description
<TVISION_HOME>/config/weblogic/*	read	none	Parameters used for TransactionVision setup related to WebLogic
<TVISION_HOME>/config/websphere/*	read	None	Parameters and scripts used for TransactionVision setup related to WebSphere Application Server
<user defined dir>/logs/*	write	write	TransactionVision logs

Note: `<TVISION_HOME>/config/datamgr/Database.properties` may contain a clear text DB2 username and password, so you must be especially careful when granting read permission of the files. For maximum security, assign the read authority only to the file owners and system administrator.

To define file access permissions for a file in Windows NT, perform the following steps:

1. Log in as the system administrator.
2. Open Windows Explorer.
3. Right-click on the properties or policy file and select Properties. The File Properties dialog opens.
4. Click the Security tab, and assign privileges to various user accounts or security groups for the file.

To set configuration file permissions in UNIX, perform the following steps:

1. Login as system administrator
2. Run **chmod** to grant the appropriate access rights to different TransactionVision groups. Recommended settings are `rw--r--` for `/datamgr/database.properties`, and `rw-r--r--` for all other configuration files.
3. If necessary, use **chgrp** to change file groups and **chown** to change the ownership of the files.

TransactionVision User Rights

Securing TransactionVision application resources requires protecting the resources on an application level and exercising the security features of the runtime platform (authentication and authorization). TransactionVision uses a Lightweight Directory Access Protocol (LDAP) server to authenticate users and store permissions for what TransactionVision operations a user is or is not allowed to perform.

There are two aspects of directory security that affect how TransactionVision interacts with the directory server: authentication and authorization. Authentication is the process of validating a user's identity. Authorization is the granting of access rights to authenticated users. Clients can either be authenticated by one of several methods or connect to the directory server anonymously. Permissions, or access rights, can be set for the entire directory content, directory entries, or even data items within an entry.

User Authentication with the LDAP Server

TransactionVision web component uses a Lightweight Directory Access Protocol (LDAP) server to authenticate users, and to store permissions relating to what a user is or isn't allowed to do. Using LDAP gives users with existing LDAP infrastructure an easy way to integrate with TransactionVision's user management.

The TransactionVision application reads the configuration file `<TVISION_HOME>/config/ldap/Ldap.properties`, and constructs the function calls in JNDI to query from the LDAP server. When a user goes to the login page, TransactionVision prompts for a user name and password. It then invokes the JNDI call to authenticate the user name and password against an LDAP server.

TransactionVision authenticates to an LDAP server by attempting a bind operation. A connection between the TransactionVision and the server is established if the bind is successful. After TransactionVision binds to the directory server, it uses account information stored there to authenticate users attempting to log in. It is always more secured to connect to the LDAP server using Secure Socket Layer (SSL).

How to Enable SSL with Directory Server

The following steps describe how to setup SSL communication using IBM's Directory Server. First of all, you need to setup up Directory Server to use SSL. In order to use SSL with you must have installed IBM's GSKIT tools (ikeyman key management tool - this is an optional package that comes with and IBM HTTP Server) to generate a certificate and keystore for the LDAP server. Once the key database and certificate are generated, enable SSL on the LDAP server and specify the key database and certificate name there. For IBM Server, this can be done through the Directory Server Web Administration tool under Server Administration > Manage Security Properties. For platforms without the LDAP server administration tool, manual editing of the ibmslapd.conf configuration file is required. Please see your IBM documentation for more details. The default SSL port is 636.

The Sun Cryptographic extensions provided within the java JDK are used by default. There are other third party JSSE packages available; the 'CryptProvider' property in the `<TVISION_HOME>/config/ldap/Ldap.properties` file can be used to point TransactionVision to a different Cryptogaph Service Provider if so desired.

Next is the setup of the certificates on the TransactionVision side. Since Java uses a different format from the key database created for IBM Directory Server. We need to create another key storage. From the IBM key management utility, select Extract Certificate, and export the certificate you created for IBM Directory Server using the base 64 encoding option. This will create a file you can then import. 'keytool' is the utility used by JSSE to create certificates. Use the -import option, to import the certificate, giving it the same name as in the IBM Directory Server definition. (For example: keytool -import -alias <certName> -file <file.arm> -keystore <where you want key store to be located>). If it is the first time, you will be asked to set a password, and this password will be required to make further changes on subsequent accesses to this keystore. The CertificateTrustStore property in the ldap.property file should be set to point to wherever you create this keystore. This property sets the javax.net.ssl.trustStore property to point to this file.

The final step is to set UseSSL property to true, and SSL communication is enabled. The LDAPServerName property needs to be changed to point to the SSL port you defined before. This is an example of `<TVISION_HOME>/config/ldap/Ldap.properties`:

```
LDAPServerName=ldap://hostname:636
ContextFactory=com.sun.jndi.ldap.LdapCtxFactory
UserDirectoryLocation=ou=xxx,o=xxx,c=xx
GroupDirectoryLocation=ou=xxx,o=xxx,c=xx
UseSSL = true
CertificateTrustStore=<keystore path>/<keystore>
CryptProvider=com.sun.net.ssl.internal.ssl.Provider
DemoMode=false
```

See http://www.sun.com/software/solutions/blueprints/1200/ldap-security.pdf and http://www-3.ibm.com/software/network/directory/server/index.html for more details about user authentication with LDAP.

User Authorization

Once a user is properly authenticated, the TransactionVision security management component retrieves access privilege information and user rights to control the views and data available to the user from the LDAP server. This information is stored within the user's current session object.

Client access to TransactionVision data is coordinated between the user and session management components. These components determine the subset of views and the associated operations that can be invoked by individual users. TransactionVision employs a view-based security policy for defining an individual user's access rights. This information is set up and maintained in the LDAP server.

If it hasn't already been done, the TransactionVision objects and schema definitions must be added to you LDAP server. This is usually done by the TVisionSetup scripts, but can also be manually installed by running the `<TVISION_HOME>/bin/CreateLdapEntries[sh.bat]` script. This will create the default Transaction vision objects in the LDAP schema and create four predefined LDAP groups to the LDAP server, which are User, Operator, Developer and Administrator respectively. It should only ever be necessary to run this once per LDAP server.

Authorization	User	Operator	Developer	Admin
View user data	X	X	X	X
Create/modify project queries	X	X	X	X
Start/stop data collection		X	X	X
Modify all project settings			X	X
View all reports	X	X	X	X
View TransactionVision page "View"	X	X	X	X
View TransactionVision page "Home"	X	X	X	X
View TransactionVision page "Current Project"	Queries only	X	X	X
View TransactionVision page "Reports"	X	X	X	X
View TransactionVision page "Help"	X	X	X	X
View TransactionVision page "Administration"				X

The above four default LDAP groups generated by CreateLdapEntries script show an example of the different types of privileges that you can assigned to a TransactionVision security group, you may modify these LDAP groups or create different ones to define your organization's own security levels in the LDAP server according to your company security policy. The way to give a user TransactionVision permissions is to assign a user, or group of users, to a TVAuthorization object. This object contains all the group attributes indicating what the user is allowed to do. A user can be assigned to multiple TVAuthorization objects; in addition to any he might be assigned because of the groups he is a member of.

Each user existing within the LDAP lookup directory who wants to access to TransactionVision web component, a user attribute, named TVAuthorityGroup, would have to be added to his entry by the administrator. This attribute would name the full distinguished name (DN) of the TransactionVision security group that this user belongs to. The group attributes of the security groups are defined under TransactionVision group directory, and each group could be configured by the administrator to contain any of the permutations of permissions listed in the table below. A TVAuthorization object specifies different security group privileges, a list of accessible projects (if permissions indicate user can only access certain projects), a list of accessible queries (if query permissions are enabled), a list of accessible reports, and a list of accessible views.

Group Attribute	Description
TVAuthCollectionState	If set to 'true', user is allowed to start and stop collection.
TVAuthUserData	If set to 'true', user is allowed to view user data.
TVAuthModifyProject	If set to 'true', user is allowed to modify project settings (communication links, collection filters).
TVAuthModifyQueries	If set to 'true', user is allowed to modify and create queries.
TVAuthModListedProj	If set to 'true' and TVAuthModifyProject isn't enabled, a user may make changes to a project, but only if it is has been entered in his project list.
TVAuthOnlyListedQueries	If set to 'true', user will only be able to use the queries entered in the list of allowable queries.
TVAuthAccessListedProjects	If this value is set to true, a user will only be able to see projects specified in his project list. If it is set to false, a user has read access to all projects.
TVAuthQueryList	List of queries the user is allowed to use.
TVAuthProjectList	List of projects the user is allowed to access.
TVAuthViewList	List of views the user is allowed to access.
TVAuthReportList	List of report groups this person is allowed to access. ‘allGroups’ specifies all reports may be viewed.

You may refer to TransactionVision Administration Guide for more details regarding how to setup different TransactionVision security levels in the LDAP server.

Connecting the LDAP Server

TransactionVision uses the LDAP protocol to talk to the LDAP server. How TransactionVision interacts with an LDAP server is configured through `<TVISION_HOME>/config/ldap/Ldap.properties` file. This configuration file contains 8 entries, by running `<TVISION_HOME>/bin/TVisionSetupInfo.[sh|bat]` script, users will be prompted about the information of LDAPServerName, UserDirectoryLocation, GroupDirectoryLocation and DemoMode settings. Here is the brief description of each entry defined in the Ldap.properties configuration file:

LDAPServerName

The LDAPServerName property identifies the directory server you wish to connect to. It directly maps to attribute of java.naming.provider.url, used in creating an initial JNDI context. The value of the property should contain a LDAP URL string (e.g. "ldap://hostname:portnum")

ContextFactory

The ContextFactory property contains the java class name to be created as the initial context factory, depending on which implementation you wish to use. The value of the property should be the fully qualified class name of the factory class that will create an initial context. If you are using WebSphere you're likely to want to use IBM classes and so would set this to 'com.ibm.jndi.LDAPCtxFactory'. If using TomCat, you might want to use 'com.sun.jndi.ldap.LdapCtxFactory'.

UserDirectoryLocation:

The UserDirectoryLocation property specifies to the TransactionVision web application what path to lookup users under for authentication, and retrieval of TransactionVision authorities. If the administrator wished to use an existing user hierarchy, he can use this property to integrate with the existing directory structure. When a user logs in, his user name is looked up with the UserDirectoryLocation to perform authentication. TransactionVision creates by default a user location under "ou=Users,ou=TVision,o=bristol,c=us" if the administrator doesn't wish to use an existing user directory. In order to be given TransactionVision permissions, users in this directory must have a TVAuthorityGroup attribute to name the full DN of the security group that this user belongs to.

GroupDirectoryLocation

Similar to the UserDirectoryLocation, this property specifies where the TransactionVision web application to lookup user groups. A security group is considered to be a definition with one or more uniqueMember attributes (such as the GroupOfUniqueNames class) containing the list of users, and TVAuthorization group attributes indicating what permissions to give this group.

UseSSL

If UseSSL is true, SSL will be enabled on the directory server connection by setting Context.SECURITY_PROTOCOL to 'SSL'.

CertificateTrustStore

The CertificateTrustStore property points to the key store (created by keytool) containing the trusted certificate for the LDAP server. The system property javax.net.ssl.trustStore is set to this value.

CryptProvider

The CryptProvider property specifies the Crytographic Service Provider you wish to register as set by java.security.Security.addProvider(). If you are using the standard JSSE package, you can leave it at the default value of com.sun.net.ssl.internal.ssl.Provider.

DemoMode

If this property is specified, instead of using LDAP server, a set of 4 predefined demo users are enabled. Those 4 users are, Admin, Developer, Operator and User respectively. They have privileges that default to their corresponding security group. This setting can be used to demo the product without needing an LDAP server installed.

Managing the TransactionVision Analyzer

The TransactionVision Analyzer communicates with TransactionVision sensors and processes event data collected by the sensors into meaningful analysis. The Analyzer runs as a windows service on windows and as a daemon on Unix. TransactionVision uses RMI (Remote Method Invocation) to communicate with the Analyzer on a specified port (default is 1099). The Web component of TransactionVision as well as the command line utilities then use RMI to communicate to the Analyzer in order to control its behavior, initiating service shutdown, allowing collection to be started, stopped, and getting status information. For more details on how to issue commands to the Analyzer see the Administration guide.

TransactionVision defines RMI security policy in the file `<TransactionVision Install Directory>/config/rmi/RMI.policy`. By default, it grants the permission java.security.AllPermission. Most significant in this is that it allows remote access to the Analyzer to any user that has access to a TransactionVision install. An Administrator of TransactionVision should be aware of this, if more secure access to the Analyzer is desired, a combination of other options can be taken to control access. The simplest mechanism to control access to an Analyzer from the host it is running on, is file permissions for the TransactionVision install. These permissions can be adjusted as needed. For example, on Unix, you should only allow read/write/execute permission to either the owner of the install, or those in a designated group that is authorized to manage TransactionVision. To control access to the Analyzer from remote machines, the RMI.policy file previously mentioned can be used. By granting java.net.SocketPermission you can control from what machine remote access to the analyzer is allowed. The below template can be used to achieve this. At minimum the RMI.policy file must grant socket permission to the machine on which the Application Server is running, otherwise it will not be possible to control the Analyzer from the web interface.

```
grant {

   permission java.net.SocketPermission "webserver:*", "accept, connect, listen, resolve";
   permission java.net.SocketPermission "remotepc:*", "accept, connect, listen, resolve";
   permission java.net.SocketPermission "localhost:*", "accept, connect, listen, resolve";
   permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete, execute";
   permission java.net.NetPermission "*";
   permission java.awt.AWTPermission "*";
   permission java.util.PropertyPermission "*", "read, write";
   permission java.lang.reflect.ReflectPermission "*";
   permission java.lang.RuntimePermission "*";
   permission java.security.SecurityPermission "*";
   permission java.io.SerializablePermission "*";

};
```

In this example, the machines 'webserver' and 'remotepc' and the local machine the Analyzer is running on are the only machines from which commands to the Analyzer would be accepted.

Outside of 'java.net.SocketPermission', the other permissions represent a minimum set of permissions required by the Analyzer itself, it is not recommended that these be changed. Additional information about the specifics of these permissions can be found at `http://java.sun.com/j2se/1.3/docs/guide/security/permissions.html`.

TransactionVision Database Security

The objectives of securing your database are:

- Preventing unauthorized access to classified data by anyone without a business need to know.
- Preventing unauthorized users from committing mischief by maliciously deleting or tampering with data.
- Monitoring user access of data through auditing techniques.

TransactionVision defines two different schemas: the system schema (or TVISION schema), and the project schema. The system (TVISION) schema stores non-events related information, while project schema stores project events collected by TransactionVision sensors.

Script "`<TVISION_HOME>/bin/CreateSqlScript.[sh|bat] –c –e –n –s`" is required to run to create the system (TVISION) schema in your TransactionVision database. After that, you can start working with TransactionVision and create TransactionVision project schemas from either the web application or CreateSqlScript. Generally speaking, one project is associated with one project schema, but users can share project schema among different projects if necessary.

TransactionVision reads the configuration file `<TVISION_HOME>/config/datamgr/Database.properties`, and starts the communication with DB2. The database username/password can be specified in those properties files for the database connection. In this case, make sure that read/write permissions of the properties files are granted only to the system administrator and appropriate security groups.

A more secure way to connect to DB2 is for the TransactionVision web application and the Analyzer to run from a user account that has read/write access to the required database tables. Add a DB2 user account to the database and grant appropriate authorities and privileges to the user. In this case, comment out the username and password field from the `<TVISION_HOME>/config/datamgr/Database.properties` file, and run a client connection to the database directly from the user account.

Assign read and write access to system tables and project tables for the TransactionVision web application. The Analyzer also requires write access to both project and system tables.

If the TransactionVision web application is the only application running on the machine, apart from what we have mentioned above, you may also consider using a JNDI database connection, which specifies the DB2 user name and password pair in WebSphere Server. To enable a JNDI database connection for TransactionVision, perform the following steps:

1. Open your WebSphere Administration Console, select WebSphere Administrative Domain->Resources->JDBC Drivers-> Db2JdbcDriver->Data Sources, create a new data source for TransactionVision web application, and enter all the required information like JNDI name, database name, user id and password.

2. Change the Implementation Classname of Db2JdbcDriver to COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource.

3. Save and exit WebSphere Administration Console.

4. Edit `<TVISION_HOME>/config/datamgr/Database.properties` and add the following two lines:

   connection_type=JNDI
   jndi_url={JNDI Name you specified in WAS}

5. Restart WebSphere and the TransactionVision application.

DB2 Authentication

A security facility outside DB2, either part of the operating system or a separate product, performs user authentication in DB2. If you are using an operating system which lacks security mechanisms, such as Windows 98, configure your environment for the DB2 server to rely on a more secure system to provide the sufficient level of security.

Some third-party products can be used to add one more layer of security to your environment. DB2 can work with these external security initiatives to protect a transaction or analytic environment.

Once a user is successfully authenticated by the system, DB2 remembers the user's identity and all other relevant security information, including the user's group list. DB2 must validate the user with a SQL authorization name or authority ID, which can be the same as the user ID or a mapped value. This connection information is kept until the user disconnects the database communication.

Since authentication can be managed either by the operating system or by third-party products, DB2 provides various authentication options that you can define in the database manager configuration file. DB2 uses the authentication parameter to determine how and what authentication should use. The authentication parameter can be set for four categories: Server, Client, DCE, and Kerberos.

For the details of different settings for database manager configuration file, see http://www7b.boulder.ibm.com/dmdd/library/techarticle/zikopoulos/0102_zikopoulos.html.

DB2 Authorization

Once DB2 authenticates a user, it proceeds to the second DB2 layer of security, which is DB2 authorization. DB2 obtains the authorization information about the user, and then assigns the relevant authorities and privileges to the user, such as the database operations the user can perform and which database resources the user can access or modify.

In response to each user request, DB2 may use more than one authorization check, depending on the objects and operations concerned. DB2 authorization is implemented by DB2 facilities. DB2 system catalogs hold a list of the privileges associated with each authenticated user. It compares the recorded privileges of the authenticated user's authorization name and the groups the user belongs to. Based on the result, DB2 then determine the permissions to grant the user.

Users require the correct privilege to perform any operations or access any data objects. The DB2 authorization process grants or revokes these privileges. Every DB2 privilege can be granted or revoked from a specific user, a specific group, or more than one group.

You may also consider using DB2 Access Control Method, which generates different subsets of the contents of information (packages) in DB2. A user can view or access only authorized data. Every database resources can work with this access control method. A good data access monitoring and control method highly improves the security performance by preventing malicious or careless unauthorized accesses to the data.

TransactionVision Configuration and Event Queue Setup Security

Secure WebSphere MQ communication should fulfill the following objectives:

- Non-repudiation: The author of a message cannot deny authorship of the message.
- Integrity: It can be verified that the message has not been modified in any way during its way to the receiver.
- Authentication: The message comes from where it claims to come.
- Confidentiality: A message can only be read by the receiver.

TransactionVision sensors are agents that reside in the space of the monitored process and collect information on WebSphere MQ calls.

When an application calls a WebSphere MQ API, it actually invokes a TransactionVision sensor interceptor function instead of the standard function. An interceptor function yields program control to the sensor for TransactionVision related processing and then forwards the call on to the real WebSphere MQ function.

The sensor manages the configuration and event queues defined in the TransactionVision communication network. This management function includes examining received configuration messages on the configuration queue, removing expired messages, retrieving newly arrived messages, and processing persistent event messages on the event queue.

The mechanism used by WebSphere MQ to transfer messages between a client and a queue manager or two adjacent queue managers is called a channel. A security channel exit is used to authenticate the partner's message channel agent (MCA) when two MCAs connect to one another. The exit is invoked at MCA initiation and termination, and at channel startup after initial data negotiation, but before any user data is exchanged.

The WebSphere MQ channel security exit offers MQ node to MQ node security, which is completely independent of application design. It is activated or deactivated by changing a channel definition. Every channel security exit applies a pair of public and private keys for verifying itself during an initial handshaking phase with its partner. The channel security exit on the client application's side operates the same way.

All MQSeries security related information is stored in a central security server database (CSSD). This database stores all queue manager names in case of MQ node to MQ node communications, and application names in case of client applications connected to a queue manager in record. It also contains all public and private keys and all channel names applied in MQSeries communications using the channel security exit.

A channel exit requires the public key of its partner to verify its partner. It retrieves the public key of its partner either from an accompanying public key file or, if the public key file does not exist, by using WebSphere MQ communication from the CSSD.

WebSphere MQ channel security exits should fit into a system that may contain firewalls, application gateways, organizational policies and procedures. If you have one WebSphere MQ server connected to another WebSphere MQ server in the outside world, use a dedicated machine as an MQ application gateway. The partner on the other side should do the same, and both partners should agree on the channel security exit to use.

If you are using an WebSphere MQ client application, consider whether each user has its own unique pair of public and secret key files or if a user must show a user ID and password each time he logs in. Instead of allowing client applications communicate directly to WebSphere MQ, it should be better to design an application to run as a client under a Web browser and communicate with a Web server, which is connected to the MQ Server.

It is possible to establish multiple security contexts between the same pair of WebSphere MQ system principals, allowing parallel channels to use the security exit.

In the WebSphere Version 5 products, there are two methods of providing security:

- DCE security, and
- The Object Authority Manager (OAM) facility

TransactionVision supports the OAM security facility.

Object Authority Manager (OAM) facility

In WebSphere MQ for UNIX systems and Windows NT, authorization for using MQI calls, commands, and access to objects is provided by the OAM, which is enabled by default. Access to WebSphere MQ entities is controlled through MQ user groups and the OAM. The setmqaut and dspmqaut commands enable administrators to grant or revoke authorizations as required.

By default, the OAM controls access to queue-manager resources. It is automatically installed and enabled for each queue manager you create, unless you specify otherwise. The OAM exploits the security features of the underlying operating system. In particular, the OAM uses operating system user and group IDs. Users can access queue manager objects only if they have the required authority.

Through OAM you can control the following:

- Access to WebSphere MQ objects through the MQI. When an application program attempts to access an object, the OAM checks that the user ID making the request has authorization for the operation requested. In particular, queues, and the messages on queues, can be protected from unauthorized access.
- Permission to use PCF commands.

Different groups of users may be granted different kinds of access authority to the same object. For example, for a specific queue, one group may be allowed to perform both put and get operations; another group may be allowed only to browse the queue (MQGET with browse option). Similarly, some groups may have get and put authority to a queue, but are not allowed to alter or delete the queue.

By default the OAM is enabled. To disable it, set the operating system variable MQSNOAUT before creating the queue manager. However, if you do this you cannot, in general, restart the OAM later. A better approach is to enable the OAM and ensure that all users and applications have access through an appropriate group or user ID.

For more information about OAM security, please refer to the IBM MQSeries Information Center.

References

- http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246520.pdf
- http://java.sun.com/products/jdk/1.3/docs/guide/security
- http://java.sun.com/j2se/1.4/docs/guide/plugin/developer_guide/security.html
- http://www.sun.com/software/solutions/blueprints/1200/ldap-security.pdf
- http://java.sun.com/products/jdk/1.1/docs/guide/rmi/
- http://www7b.boulder.ibm.com/dmdd/library/techarticle/zikopoulos/0102_zikopoulos.html
- http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/SG245306.html
- http://www-3.ibm.com/software/ts/mqseries/library/manualsa/csqzae05/csqzae0503.htm
- http://www-3.ibm.com/software/network/directory/server/index.html
- http://java.sun.com/products/jdk/1.4/docs/tooldocs/solaris/rmid.html
- IBM LDAP Implementation Cookbook
- TransactionVision Administrator's and User's Guides
- IBM MQSeries Information Center