MLS DBMS Interoperability Study*

Rae K. Burns, AGCS, Inc.
Yi-Fang Koh, Raytheon Electronic Systems

ESC/AXS, Building 1704, Hanscom AFB, MA 01731
burns/koh@stars1.hanscom.af.mil

Interoperability among heterogeneous databases is a fundamental requirement of many emerging Department of Defense (DoD) systems. Often these systems also have requirements for Multilevel-Secure (MLS) operation, where data is labeled to reflect its sensitivity level (e.g., UNCLASSIFIED, SECRET, etc.). The Air Force Rome Laboratory MLS Database Management System (DBMS) Interoperability Study has surveyed the available Commercial-Off-The-Shelf (COTS) products supporting interoperability and tested several of them in a multilevel environment. We selected representative products and implemented test scenarios in the ESC/AXS Security Products Transition Analysis Facility (STAF). Our test environment included three commercial MLS DBMS products (Trusted ORACLE7, Informix Online/Secure, and Sybase Secure SQL Server) on several different MLS Operating System (OS) platforms. We also employed "system high" platforms running standard versions of the DBMS and OS products. We successfully moved data to and from the MLS databases using different COTS interoperability solutions. This paper describes our testing efforts and summarizes the lessons learned.

1. Introduction

The Multilevel Secure Database Management System Interoperability Study was initiated by Air Force Rome Laboratory to expedite the transition of trusted database technology into operational Air Force C4I environments. The overall goal of the study is twofold:

1) To examine the theoretical basis for heterogeneous trusted database interoperability, identify issues, and transition findings to the communities involved in future development (vendors, DoD users, and applicable standards groups).

2) To develop demonstrable examples of database interoperability that illustrate the advantages of MLS DBMS products in support of Air Force operational requirements for multilevel security.

The results of the first goal were documented in the interim report [1] and are based on an analysis of current and proposed interoperability approaches documented in the database research literature. To address the second goal, a multilevel database testbed was created at the Security Products Transition Analysis Facility at Hanscom Air Force Base. Within the testbed, COTS database products have been installed, including both MLS DBMS products and connectivity products. The products have been integrated together using a simple Air Base Status database as

* This work was sponsored by Air Force Rome Laboratory under the USAF ESC/AXS PRISM Program, contract numbers F19628-92-C-0006 and F19628-92-C-0008.

the underlying application. This paper presents the results of our testing and demonstrations of INTERSOLV Open Database Connectivity (ODBC), Oracle PL/SQL Extender (PLEX), Sybase OmniSQL, and PRAXIS OmniReplicator. The operating system platforms in the testbed include SunOS 4.1.3, Sun Solaris 2.4, Sun Trusted Solaris 1.1, and Santa Cruz Operations (SCO) Secureware CMW+ 3.0. Standard versions of ORACLE7 and Sybase SQL Server are installed on SunOS platforms and the MLS versions on Sun Trusted Solaris platforms. Informix Online/Secure is installed on the SCO CMW platform. The connectivity products were installed on different platforms, depending on product availability and test scenario configurations.

1.1 Study Approach

During the course of the Interoperability Study, we analyzed and screened a number of different COTS connectivity products, and selected representative products to integrate into the STAF testbed. Based on our analysis we identified four categories of COTS interoperability products: standards-based, vendor-specific, gateway, and replicator.

(1) Standards-based Solutions

We looked at two different standards-based interoperability approaches: Open Database Connectivity (ODBC) and Remote Data Access (RDA). Microsoft's ODBC interface [2] is one of the first implementations of the SQL Access Group (SAG) Call Level Interface (CLI) standard. ODBC is based on the X/Open and SAG CLI 1992 specification [3], defining a C or C++ programming language interface for standardized DBMS connectivity. We successfully used two different INTERSOLV ODBC products to access data in an MLS database.

The International Standards Organization (ISO) Open Systems Interconnection (OSI) RDA standard [4] defines the message format for sending SQL queries to a DBMS and receiving data from the DBMS. The National Institute of Standards (NIST) has supported efforts to promote the standard, but currently there are few COTS products that implement the RDA standard. The major MLS DBMS vendors currently only support proprietary message formats and do not provide RDA interfaces to their products. Consequently, for the Interoperability Study, we did not perform any testing with RDA-compliant products.

(2) Vendor-specific Solutions

Several DBMS vendors support interfaces that facilitate interoperability but do not provide a general purpose solution. We used an Oracle Federal tool, Oracle PLEX, to integrate data from a Sybase Secure SQL Server database into a Trusted ORACLE7 application.

(3) Gateway Solutions

Gateway products generally map the SQL schema from one DBMS onto an equivalent schema in another DBMS, giving the user transparent read and/or write access to a foreign data source. We tested the Sybase OmniSQL gateway product in two different configurations. In one configuration, we retrieved data from two different MLS databases; in the other, we loaded data from multiple single-level databases into a central multilevel database.

(4) Replicator Solutions

Replication supports the automatic updating of remote databases based on changes made to another source database. Our experiments with PRAXIS OmniReplicator included replication among MLS databases as well as replication from a single-level database into a multilevel database.

1.2 Example Database

Since the scope of this study was limited to interoperability issues, we chose a simple application for testing each connectivity product. The Air Base Status database contains information about the facilities and runways of several air bases and indicates the current status of the base and each of its runways (e.g., whether it is operational or not). We created this database on each of the platforms in the testbed and populated it with a small amount of data on air bases in the Persian Gulf. We then designed tests for the interoperability products that retrieved and updated the status information using different operational scenarios.

1.3 Product and Configuration Limitations

Because of our focus on available COTS technology, we were limited in the level of assurance achievable in our test configurations. The MLS DBMS products we used have been evaluated (or are being evaluated) at B1. The Compartmented-Mode Workstation (CMW) platforms we employed are also fundamentally B1 class systems. We limited our test scenario accreditation range to CONFIDENTIAL and SECRET, with some releasability compartments, in order to demonstrate the MLS DBMS and connectivity products in an appropriate risk environment.

None of the COTS connectivity products we used have been evaluated nor were they targeted for use in a multilevel context. Their use imposes additional limitations since none of the products were able to directly interpret sensitivity labels. To retrieve sensitivity labels, we created views within the multilevel databases that automatically converted the sensitivity label to a character string. By accessing these views instead of the base tables, the sensitivity labels were made available to the connectivity products as *advisory* labels. (The labels can only be advisory since the COTS connectivity products are not *trusted* to manage sensitivity labels.) For database updates, a connectivity product was run as a single-level process; consequently, all updates were labeled by the MLS DBMS with the sensitivity label associated with that process.

The remainder of this report describes the COTS solutions and our experiences installing, configuring, and demonstrating them in the STAF multilevel database testbed.

2. Open Database Connectivity

The ODBC interface allows a user to write a single application to access databases managed by different DBMS products. SQL statements can be included directly in the source code or can be constructed dynamically at run time. The underlying communication with the DBMS is completely transparent to the application.

ODBC architecture includes four components as illustrated in Figure 1. An **Application** uses the **ODBC Application Programming Interface (API)** to call ODBC functions that submit SQL statements and retrieve data. The **Driver Manager** loads drivers on behalf of an application, then the **Driver** for a specific DBMS processes ODBC function calls and submits the SQL statements to the designated **Data Source**. We successfully integrated two ODBC products to access different data sources:

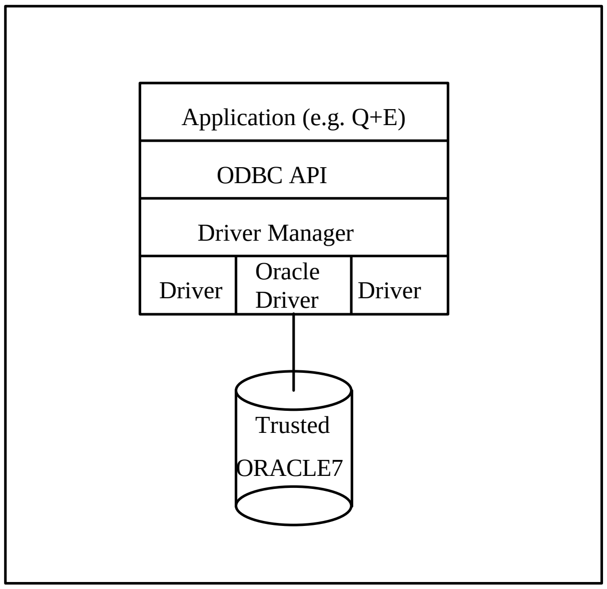

Figure 1. ODBC Configuration

INTERSOLV Q+E for Windows: Q+E is a query and reporting application that uses ODBC drivers to provide access to a large number of different database products [5]. We installed Q+E and the ODBC Driver Pack [6] on a Windows NT platform. We also installed Oracle SQL*Net for Windows to provide connectivity to Oracle databases. After configuring both the ODBC.INI file and the local Oracle configuration files to refer to the Trusted ORACLE7 database, we were able to retrieve information and generate customized Q+E reports that included both the data and sensitivity labels.

INTERSOLV ODBC for UNIX: The ODBC drivers for UNIX were installed on a Sun Solaris 2.4 system. The Oracle ODBC driver was linked with the standard ORACLE7 client libraries for Solaris. We then wrote a C program that used ODBC functions to connect to the Trusted ORACLE7 database on a Sun Trusted Solaris 1.1 CMW. The program simply retrieved Air Base Status information (including sensitivity labels) and displayed it interactively to the user. Since we did not have client libraries for any of the other MLS DBMS products on Sun Solaris, we did not test the other MLS databases.

The specific configuration parameters for the UNIX and Windows environments are documented in the ODBC Interoperability Report [7] along with the C source code developed for the UNIX testing. The ODBC API does provides a DBMS-independent interface that can be used to access MLS databases.

3. Oracle PLEX

Oracle PLEX is a vendor-specific interoperability solution that allows an Oracle application to access foreign data sources [8]. It provides a set of functions that extend the capability of Oracle’s Programming Language/Structured Query Language (PL/SQL) to communicate with an *application server* that performs operations outside the scope of a traditional database application. The PLEX product provides a number of program development tools to build both the application server and the PL/SQL modules used to communicate with the server. For our interoperability test, we developed an application server that retrieved data from a Sybase Secure

SQL Server database and displayed it using Oracle PL/SQL routines in a Trusted ORACLE7 database. The application server used the Sybase DB-Library API to retrieve data from the Air Base Status database. While we were successful in accessing a remote database using PLEX, the solution was fairly complex and did not provide a generic interoperability solution, as documented in the Oracle PLEX Interoperability Report [9]. However, for access to non-database information, ORACLE PLEX provides a viable basis for using Oracle's PL/SQL, rather than C, for application development.

4. OmniSQL Gateway

The Sybase OmniSQL Server allows an application using the Sybase Open Client API to access databases managed by other DBMS products[10]. Information about the other databases is stored locally in an OmniSQL database as mappings from the local environment to the remote environments. Both user identifier mappings and table definition mappings are maintained by OmniSQL. When a local request is made, OmniSQL uses the mapping information to access the appropriate data sources.

There were two different test scenarios established for the OmniSQL interoperability testing. First, the OmniSQL Server was installed on a SunOS system and used to combine data from a Trusted ORACLE7 database and a Sybase Secure SQL Server database. In the second scenario, multiple instances of the OmniSQL Server were run on a Sun CMW at different sensitivity labels and used to update a central Sybase SQL Server database.

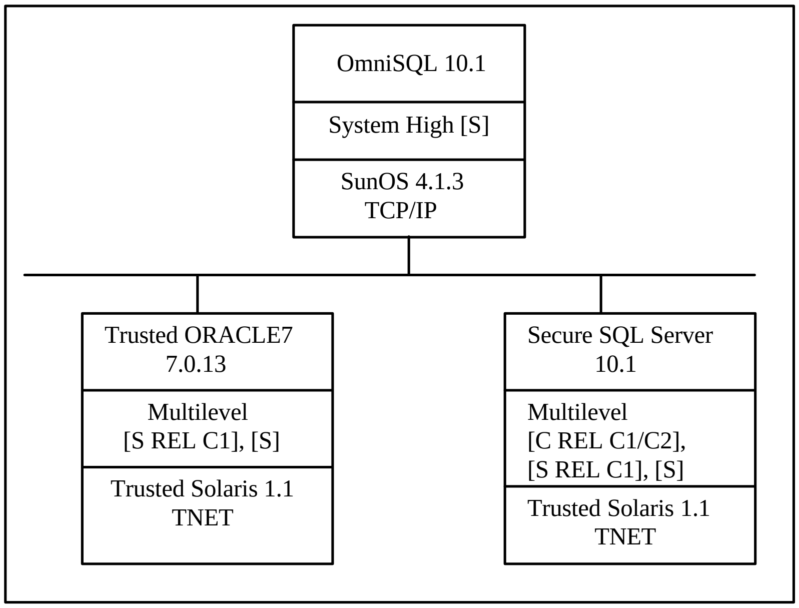

Figure 2. OmniSQL SunOS Environment

Figure 2 illustrates the configuration for the first scenario. To combine data from the Trusted ORACLE7 database with data from the Sybase Secure SQL Server database, we created a stored procedure in the OmniSQL database that referred to both of the MLS databases. We first had to map the MLS data (e.g., views with the sensitivity labels converted to character string datatypes) to locally defined OmniSQL tables. OmniSQL provides a utility to automatically generate the required mapping specification from the table definition in the remote database [11]. However, we had difficulty using the utility because it did not expect the extra sensitivity label column that the MLS DBMS products append to each table. (While Trusted ORACLE7 only appended the label column to base tables, Sybase Secure SQL Server appended it to views as well.) The problems we encountered are documented in the OmniSQL Interoperability Report [12]; however, we were able to successfully combine the data from the MLS databases after dealing with the sensitivity label problems.

For the second scenario, the OmniSQL Server software was installed on a Sun CMW. We set up multiple OmniSQL Servers to retrieve the Air Base Status data at three different sensitivity levels from three different databases (standard Sybase, Trusted ORACLE7 and ORACLE7) and update a central multilevel Sybase Secure SQL Server Air Base Status database.

Three different sensitivity labels were used in this testing: [C REL CNTRY1/CNTRY2], [S REL CNTRY1], and [S]. An OmniSQL database and a server instance were required at each sensitivity level in order to communicate with the remote database at a single level. In addition to the table mappings required for this scenario, we set up an OmniSQL stored procedure at each level. The stored procedure first retrieved the requested status information and then used that data to update the status information in the central multilevel database. Figure 3 illustrates the configuration for this scenario.

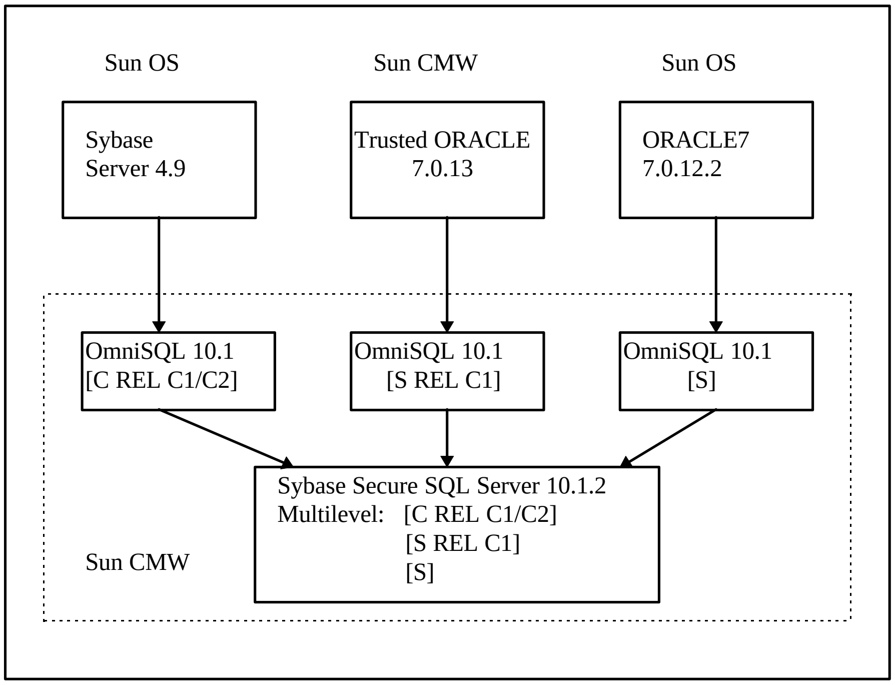

Figure 3. OmniSQL Multilevel CMW Environment

During the OmniSQL Server testing, several interoperability problems surfaced that involved the network configuration definitions on the Sun CMW platforms. The first problem was caused by the Trusted ORACLE7 SQL*Net listener process. The listener process executes with privilege and runs at the *lowest* level of data within the database. Our first scenario was accessing tables at the *highest* level from a single-level untrusted SunOS system. This caused several different problems, all of which were resolved by small modifications to the Sun CMW network host configuration file (TNETRHDB). We had similar problems with Sybase Secure SQL Server and floating information labels which were also solved by minor changes to the TNETRHDB file.

Details on these problems and their solutions are documented in the OmniSQL Interoperability Report [12].

Within both of these test environments, we were able to demonstrate that the OmniSQL Server can be used successfully to retrieve and update data from databases managed by dissimilar MLS DBMS products without violating the overall system security policy.

5. OmniReplicator

The PRAXIS OmniReplicator is a replication server designed to work with heterogeneous databases [13]. The tables and columns to be replicated are specified by an administrator using an OmniReplicator application on a PC. The administrator application connects to the source database, creates tables for use by the OmniReplicator, stores the replication configuration as defined by the administrator, and creates triggers to capture the updates to be replicated. When OmniReplicator processes run on the source platform, they uses the tables within the source database to control and monitor the replication activities. For communication to the target database, OmniReplicator relies on the SequeLink product from INTERSOLV. SequeLink transforms the update statements into messages and transmits them to the target database.

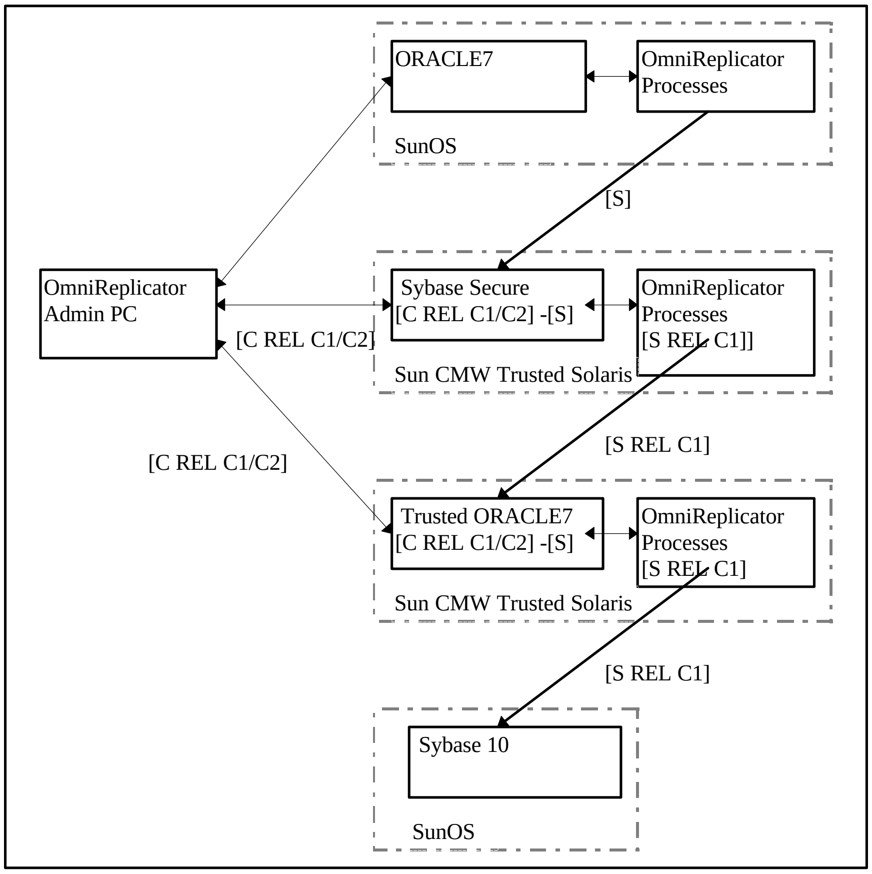

Figure 4. OmniReplicator Configuration

For our interoperability test, we set up a cascading configuration, as illustrated in Figure 4. The standard version of ORACLE7 is the source database. When updates are made to it, they are replicated to the Secure Sybase database at [S]. These updates to the Sybase Secure database cause replication to the Trusted ORACLE7 database. We used the Sybase *trusted trigger* feature to perform a downgrade from [S] to [S REL C1] prior to replicating the data to Trusted ORACLE7. We designated the OmniReplicator triggers as trusted and authorized them to "writedown" to [S REL C1] when they stored data. As indicated in Figure 4, the Sybase Secure OmniReplicator processes execute at [S REL C1], not at [S], to read and process the replication information stored by the trusted triggers. In turn, updates to the Trusted ORACLE7 database cause replication to a standard Sybase database.

The installation of the SequeLink and OmniReplicator software in a multilevel environment is fairly complex [14]. The SequeLink product is intended to be installed by the **root** user, however we modified the install script and installed it without privilege. In addition, when a remote user tries to connect to the SequeLink Server, SequeLink attempts to validate the user's login ID and password. However, on the CMW systems, the **/etc/passwd** file is not the same as on a regular UNIX system. The **/etc/passwd** file does not contain encrypted passwords, so SequeLink initially rejected remote connections. To get around this problem, we put an entry in the **/etc/passwd** file for the account we were using for our replication testing; this solved the problem without creating any errors with the CMW user authentication. Finally, since we needed to connect at several different labels, we created a multilevel directory to store the SequeLink log files.

For the Sybase version of SequeLink, we also had to modify the SequeLink stored procedures that retrieve datatype information. We removed the sensitivity datatype since the OmniReplicator Administrator software on the PC did not correctly interpret the sensitivity datatype. We also had to modify the scripts that create the OmniReplicator tables within the source Sybase database. These modifications were extensive, since the creation of multilevel tables in Sybase requires extra parameters on the CREATE TABLE statements. Finally, we had a problem with the use of the **sa_role** feature. The OmniReplicator software assumes there is a Sybase user, **rpdbo**, who is authorized for the sa_role. However, in Secure Sybase, the role must be *enabled* before it is effective, so the operations requiring the sa_role failed. We finally managed to overcome this problem by changing database ownership and privileges so that the use of the sa_role was unnecessary. We had fewer problems with Trusted ORACLE7, but still had to make a couple of changes to the installation scripts.

Once the replication parameters were correctly setup using the OmniReplicator Administrator, we were able to cascade the replication of Air Base Status information from the original source through the intermediate multilevel databases. The use of a trusted trigger within the Sybase Secure database supported an automated downgrade. Additional features of OmniReplicator could be employed to replicate data to different target databases based on data sensitivity labels and to sanitize data as it is replicated.

6. Lessons Learned

The overall conclusion of this Interoperability Study is that COTS connectivity products can successfully be used to support Air Force operational requirements in a multilevel environment. Since the COTS DBMS products currently do not provide *high assurance* solutions, the accreditation range for an operational system will necessarily be limited.

We encountered two fundamental problems with sensitivity labels. First, each of the MLS DBMS products define the sensitivity label column differently, both in column name and datatype. In general, COTS connectivity products do not recognize the sensitivity label datatype and cannot interpret it. An SQL standard for sensitivity labels would greatly facilitate interoperability using

COTS products. The standard would primarily need to address the label *datatype* and how the label appears to client applications (e.g., as a character string or as an internal label to be interpreted by the client software).

A second problem with sensitivity labels is that they can only be advisory. If there were an appropriate infrastructure established to deliver a sensitivity label along with the data to a *trusted* component on a client system, then a trusted application could display the data and sensitivity labels. The infrastructure could involve extensions to trusted networking (e.g., a security API as proposed by the Trusted Systems Interoperability Group (TSIG)) or could be based on digital signatures (e.g., where the MLS DBMS would sign both the data and its label before returning it to a client application). In order to have trusted sensitivity labels, some additional infrastructure must be developed.

The other problems we encountered had to do with the trusted networking parameters and the security environment on the MLS platforms. Configuring a single MLS platform is complex and difficult. To correctly configure a heterogeneous MLS network of any size is even more difficult. Our problems with the Sun CMW network configuration parameters were the result of some subtleties involved with privileged software accessing single-level hosts. Further standardization of multilevel network protocols, configuration parameters, and label translation capabilities would substantially improve the administrator's ability to configure a secure heterogeneous network that supports database interoperability.

Future work in MLS database interoperability should address both the sensitivity label issues and the trusted networking infrastructure. In addition, as higher assurance operating systems and database management systems are developed, interoperability using high assurance MLS database servers should be pursued. A high assurance database server could support a wider accreditation range and could be accessed from both system high platforms and low assurance B1 and CMW platforms.

7. References

1 Interoperability of Trusted, Heterogeneous, Autonomous, Distributed (THAD) DBMS Interim Report, Security Products Transition Analysis Facility, ESC/AXS, Hanscom AFB, 2/22/95.

2 Microsoft ODBC 2.0 Programmer's Reference and Software Development Kit (SDK) Guide, Microsoft Press, 1994.

3 X/Open CAE Specification, Structured Query Language (SQL), (ISBN:1-872630-58-8 C201), August 1992,.

4 X/Open CAE Specification: SQL Remote Data Access, (ISBN: 1-872630-98-7 C307), 1994.

5 Q+E User's Manual, INTERSOLV, January 1995.

6 Data Direct ODBC Drivers, INTERSOLV, February 1995.

7 ODBC Interoperability Report, Security Products Transition Analysis Facility, ESC/AXS, Hanscom AFB, February, 1996.

8 Oracle PLEX Installation and User's Guide, Oracle Federal, August, 1994.

9 Oracle PLEX Interoperability Report, Security Products Transition Analysis Facility, ESC/AXS, Hanscom AFB, February 1996.

10 OmniSQL Server System Administration Guide, Sybase Inc., March, 1993.

11 OmniSQL Server Class and Utilities Reference Manual, Sybase Inc., November 1993.

12 OmniSQL Interoperability Report, Security Products Transition Analysis Facility, ESC/AXS, Hanscom AFB, February, 1996.

13 OmniReplicator Concepts and Facilities, Praxis International, Inc., 1995.

14 OmniReplicator Interoperability Report, Security Products Transition Analysis Facility, ESC/AXS, Hanscom AFB, July 1996.