Database Management Group

Enterprise Computing Services

DB2 Security, Access, and Performance for Distributed Database

POLICY/PROCEDURE NUMBER: 001016-02 **Approved by:** Edward J. Hively

SUBJECT: Security, Access, and Performance issues for DB2 Universal Database for z/OS.

Introduction: Securing and protecting our client's data is our primary concern by providing information access to legitimate parties while preventing access by unknown and unauthorized parties. The core conflicts have been access type and controlling potential adversaries from abusing secured data.

Purpose/Scope: This policy has been established to protect the integrity of the State of Nebraska data from fraudulent use and/or abuse resulting from unauthorized access. The five fundamental objectives are: authorization [(1)], authentication [(2)], accountability [(3)], privacy [(4)] and data integrity [(5)] (see Section III for definitions).

- Section I describes Distributed Database Access issues.
- Section II discusses additional security concerns.
- Section III provides the five fundamental objective definitions.

Section I. Distributed Database Access.

A. Security, Access, and Performance Issues.

Distributed (e.g. client-server or n-tier) applications require remote database access across different network topologies. This network linkage introduces potential issues that must be addressed by Management, Application Development, and our client's. We are now working outside our secure mainframe (i.e. host) environment and we must "think" security when we design and implement distributed applications.

For those developing distributed applications these issues must be addressed:

- Client-side security controls are ineffective; desktop security is usually weak. If it exists it should not be relied upon as an effective security mechanism.
- Generic application programming interfaces (API's) such as ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity) add a new 'dynamic' complexity into security, access, and performance issues which should be considered when developing applications.
- ODBC access requires dynamic SQL execution. Dynamic SQL requires a DB2 prepare, describe, and bind of all statements, which places locks on the catalog and directory tables for the duration of the executed statements. This is an iterative process; therefore, it is repeated each time dynamic SQL is executed.
- Dynamic SQL requires the end users to have DML (i.e. select, insert, update, and delete) privileges on DB2 tables to access the relational data. This increases the administration costs of security and impacts the performance of the Database Management System (DBMS) overall.

- Use of "system" or "surrogate" user id's to run the application causes many problems. You lose the ability to track the activities of individual users within the database in regards to security, auditing, and performance.
  - ✓ This architecture increases the security exposure for all individuals who can gain access to the application server itself since the user id and password must be stored with the applications server so it may access the remote DBMS data on behalf of the user. This is a serious security consideration given that there is some information that should be secured from both internal and external users.
- Dynamic SQL cannot be controlled and, therefore, cannot be guaranteed that it happened.
  - ✓ It is difficult to debug applications programs since SQL displayed in error logs may be difficult to recreate given its dependencies on the application logic.
- Dynamic SQL requires bind authority and execution authority to be given to the person who actually submits the dynamic SQL.
- Dynamic SQL places locks on both system and application db2 objects (i.e. tablespaces, tables, and indexes).
  - ✓ This increases the likelihood that those applications that utilize dynamic SQL must be segregated from transactional applications that can achieve significantly higher throughput.
- Dynamic SQL causes major performance problems over static SQL since dynamic must execute hundreds of run-time calls (static SQL is the preferred method of coding along with calling Stored Procedures).
- Dynamic SQL introduces accountability issues since there is no assurance that any transaction that takes place can be subsequently proven.
- Distributed Relational Database Architecture (DRDA) connections that utilize TCP/IP have very few security controls in comparison to connections using SNA protocols. Therefore, access to our client's data is expose to unauthorized access.

B. Resolution.

Having stated some of the Security, Access, and Performance Issues in Section A, there are legitimate reasons for opening up the DB2 for z/OS Enterprise system to distributed database (i.e. client/server) access.

Therefore, the following criteria must be met before we can allow this type of access.

- For client-side security the Enterprise should use middle tier to authenticate (e.g. LDAP, e-Directory) when possible.
  - ✓ All inbound client connections will be authenticated against RACF. This may be accomplished by utilizing e-Directory; which in turn will refer incoming LDAP requests for DB2 access to the z/OS RACF server.
- Distributed Relational Database Architecture (DRDA) connections that use secure protocols are still a good viable method.
  - ✓ DRDA connections will 'in bound' user ids to be translated to "No Log" id.
  - ✓ "No Log" Id's are Id's that individuals can use to authenticate within the group but cannot log onto the mainframe server to gain access to DB2 for z/OS data.
- Applications that execute in client/server environment will be required to execute the 'set current sqlid' statement to pick up authority to application data. This assures that the Id executing is authorized.

- ❖ The 'set current sqlid' statement will be set to a RACF secondary group id; this ensures the "No Log" id is a member of the groups authorized to access the requested resource(s). The secondary group id will maintain the privilege to access the DB2 for z/OS data.

- ❖ Distributed Relational Database Architecture (DRDA) connections that utilize TCP/IP have very few security controls. Therefore, access to our client's data is exposed to unauthorized access. Application developers should ensure that our business clients understands and are willing to accept this risk.

- ❖ Those applications desiring to utilize TCP/IP must find a secure method (i.e. e-Directory or LDAP) to protect the DB2 for z/OS data.

Section II. Additional Security Concerns.

To make our environment secure, the data provided in the DB2 for z/OS enterprise must be determine 'to be' of value and then apply the appropriate level of technical and procedural security. To wait for our data to be unavailable (i.e. through a denial of service attack) is unthinkable. Management has to decide how to protect the enterprise information from the different paths of access. Previously, batch and CICS were the only forms of accessing this data. Today, with distributed database access and the Internet the platforms and access concerns bring more challenges to the Database Management Systems (DBMS).

The main areas to address when considering going to distributed database application are:

- Database/Application Security design considerations.
- User Id and passwords to access this data.
- Access control built into the application.
- User authentication to the database.
- Encrypting the information in the database when there is a need for confidentiality.
- Separate Security Administration should be considered.
- Third Party Database Security Products should include features built in to the design.

Bottom Line: As we explore more avenues to provide our customers with access to their data. The enterprise must implement security controls in the end-to-end database environment. This means improve security coverage by access, authentication, authorization, confidentiality, and privacy controls.

Section III. Appendix.

A. Definitions.

(1) Authorization – Assurance that the person or computer at the other end of the session is permitted to do 'what they ask for'.

(2) Authentication – Assurance that the resource (person or computer) at the other end of the session 'really' is what it claims to be.

(3) Accountability – Assurance that any transaction that takes place can subsequently be proven to have taken place. Both the 'sender' and the 'receiver' agree that the exchange took place.

(4) Privacy – Assurance that sensitive information is not visible to any eavesdropper.

(5) Data Integrity – Assurance that the information that arrives is the same as when it was sent.