Security Target

for

Trusted RUBIX Version 5.0

Multilevel Security

Relational Database Management System

Version 1.4.8

September 30, 2004

Prepared By

Mitretek Systems, Inc.

3150 Fairview Park Drive South

Falls Church, VA 22042-4519

Prepared For

Infosystems Technology, Inc.

7700 Leesburg Pike, Suite 402
Falls Church, VA 22043
TEL 703-448-0002

Table of Contents

List of Tables

List of Figures

1 Introduction

The target of evaluation (TOE) is Trusted RUBIX 5.0, which is a Relational Database Management System (RDBMS) product that provides security features of Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Security Auditing for high levels of trust. Infosystems Technology, Inc. (ITI) is the developer of the Trusted RUBIX product.

This document is a Security Target (ST) that provides a basis for the evaluation of Trusted RUBIX 5.0. The ST will describe the following topics and express the rationale for the derived security objectives, security requirements, and the TOE security functions for the intended Information Technology (IT) environment:

- Identify the intended information technology (IT) environment,
- State the derived security objectives addressing all of the environmental security concerns and assign the responsibilities of the TOE and its environment in meeting the security needs,
- Identify the IT security requirements for the objectives that the TOE has been determined to meet,
- Identify any security requirements for the IT environment, and
- Specify the security functions and assurance measures offered by the TOE for satisfying those requirements.

The following IT environment factors are described in this ST:

- Assumptions regarding the security environment and the intended usage of the TOE;
- Threats identified for the TOE and the data the TOE protects; and
- Applicable organizational security policies that identify relevant policy statements and rules with which the TOE must comply.

1.1 Identification

Title:	Security Target for Trusted RUBIX 5.0 Multilevel Security Relational Database Management System (RDBMS), Version 1.4.8
TOE Identification:	Trusted RUBIX version 5.0, Infosystems Technology, Inc.
Common Criteria (CC) Identification:	The Common Criteria for Information Technology Security Evaluation version 2.1, August 1999

Registration: (To be completed by registrar)

Keywords: Database, Relational Database Management System, RDBMS, DBMS, COTS, C2, B1, B2, TCSEC, Medium Robustness, Multilevel, Access Control, Mandatory Access Control, MAC, Discretionary Access Control, DAC, Labels, Information Flow Control.

1.2 Security Target Overview

The TOE, Trusted RUBIX 5.0, is an SQL-based client/server multilevel secure relational DBMS product. Its internal database system design focuses on the modularity and layering principles, which are critical in high assurance systems. Trusted RUBIX 5.0 is built to meet the Evaluation Assurance Level 4 (EAL4). Trusted RUBIX functions enable DBMS users to manipulate the data contained in tables. Using RUBIX/SQL users can manipulate the data in a single table, or perform elaborate operations involving several tables.

This Security Target documents the security characteristics of Trusted RUBIX, Version 5.0. The Security Target version 1.4.8 of Trusted RUBIX, Version 5.0 consists of the following sections:

Section 1, Introduction, contains identification of the ST, an overview, conformance claims, conventions, and terms. The ST identification subsection provides the title and keywords to identify the ST and the TOE to which it refers. An ST overview summarizes the ST in narrative form. The Common Criteria (CC) conformance claims state with which portions of the CC the TOE conforms. The conventions subsection states the notations and convention used and the terms subsection provides a list of terminologies and their definitions used in the ST.

Section 2, Target of Evaluation (TOE) Description, describes the Target of Evaluation (TOE), including the product type of the TOE and the scope and boundaries of the TOE in general terms, both in a physical and a logical way.

Section 3, TOE Security Environment, identifies and explains the assumptions about the intended usage of the TOE and the environment of use of the TOE, any known or presumed threats to the assets against which protection will be required, either by the TOE or by its environment, and the organizational security policies with which the TOE must comply.

Section 4, Security Objectives, defines the security objectives for the TOE and for the TOE environment.

Section 5, IT Security Requirements, identifies the TOE security functional requirements and assurance requirements drawn from the Common Criteria Functional and Assurance requirements that the TOE has been determined to meet. It also identifies any security requirements for the IT environment.

Section 6, TOE Summary Specification, describes the IT security functions and security mechanisms with which the TOE meets the functional requirements and shows the mapping of

IT security functions to requirements. This section also describes the assurance measures of the TOE with which the TOE meets the assurance requirements.

Section 7, Rationale, provides the TOE summary specification rationale for concluding that the TOE conforms to the requirements; the security requirements rationale demonstrates that the IT security requirements are suitable to meet the security objectives; and the security objectives rationale demonstrates that the stated security objectives are suitable to counter the identified threats to security, cover all of the identified organizational security policies and assumptions of this ST.

1.3 Conformance Claims

The TOE conforms to the Common Criteria for Information Technology Security Evaluation version 2.1, August 1999, Parts 2 and 3.

The TOE does not claim conformance to any PP.

1.4 Conventions

The main body of this text is typeset in a proportional width upright font (Times new Roman). Trusted RUBIX commands, program statements, and function calls are typed in a fixed width font (`Courier`) in italics (*`Courier`*) and ***`bolded`*** at their first appearance. The underlying operating system directories, files, function calls, and commands are typed in a fixed width font (`Courier`).

The notation, formatting and conventions used for CC requirements in this Security Target are largely based upon those used in the Common Criteria, Version 2.1. Within the Common Criteria functional and assurance requirements in ST Section 5, required text taken verbatim from the CC requirements is typeset in an upright font, while variable text specific to this ST use typeset defined in Table 2.

1.5 Terms

This section describes terms that are used throughout the Security Target. When possible, terms are defined as they exist in the Common Criteria for Information Technology Security Evaluation.

- Access control – The process of limiting access to the resource of a system only to authorized programs, processes, or other systems (in a network). Synonymous with *controlled access* and *limited access*.
- Authorized – Possessing the rights and/or privileges necessary (in accordance with the TSP) to perform an operation. (Derived from CC [2] part 1 § 15 and CC part 2 § 26.)

- Bell-LaPadula Model [1] – A formal state transition model of computer security policy that describes a set of access control rules. In this formal model, the entities in a computer system are divided into abstract sets of subjects and objects. The notion of a secure state is defined and it is proven that each state transition preserves security by moving from secure state to secure state; thus, inductively proving that the system is secure. A system state is defined to be "secure" if the only permitted access modes of subjects to objects are in accordance with a specific security policy. In order to determine whether or not a specific access mode is allowed, the clearance of a subject is compared to the classification of the object and a determination is made as to whether the subject is authorized for the specific access mode. The clearance/classification scheme is expressed in terms of a lattice.
- Classification – A designation attached to information that reflects the damage that could be caused by unauthorized disclosure of that information. A classification includes a sensitivity level (UNCLASSIFIED, CONFIDENTIAL, SECRET, or TOP SECRET) and a set of zero or more compartments (CRYTO, NUCLEAR, etc.). The set of classifications, together with their hierarchical relation defining the allowed information flows between levels, form a lattice. Most dissemination controls, such as NATO, NOFORN, and NOCONTRACTOR, can be handled as additional compartment names.
- Clearance – The degree of trust associated with a person. This is established on the basis of background investigations and the tasks performed by the person. It is expressed in the same way as classifications (i.e., a sensitivity level and a (possibly null) compartment set).
- Confidentiality — (1) The concept of holding sensitive data in confidence, limited to an authorized set of individuals or organizations. (2) The property that information is not made available or disclosed to unauthorized individuals, entities, or processes.
- Container – A multilevel information structure. A container has a classification and may contain objects (each with its own classification) and/or other containers.
- Database — A database is a logical container that may contain many different kinds of data arranged in tables, where the data contained in each table is in some way related to the data in the other tables in that database.
- Data confidentiality — the state that exists when data is held in confidence and is protected from unauthorized disclosure.
- Data integrity - The state that exists when computerized data is the same as that in the source documents and has not been exposed to accidental or malicious alteration or destruction.
- DBMS data (TSF data) — Data that the DBMS (i.e., TOE) creates, maintains, and uses to operate the DBMS (e.g., configuration parameters, user security attributes, transaction log, audit instructions and records).

- DBMS objects — DBMS objects are named objects (i.e., objects that can be named by users outside the DBMS) that are protected by the DBMS TOE security functions (TSFs) for enforcing the TOE Security Policy (TSP). DBMS objects may be aggregations of data contained in other DBMS objects.
- DBMS internal object – All DBMS objects contained within a single database (i.e., all DBMS objects except the database itself).
- DBMS resource — A resource controlled by the DBMS (i.e., within the TSC) including, for example, the software programs used to operate the DBMS.
- Discretionary Access Control (DAC) — A means of restricting access to objects based on the identity of the subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with certain access privilege is capable of passing that privilege (perhaps indirectly) on to any other subject (unless restricted by mandatory access control).
- Dominate — Security level (or sensitivity label) A is said to dominate security level (or sensitivity label) B if the hierarchical classification of A is greater than or equal to that of B and the non-hierarchical categories of A include all those of B as a subset.
- Equal — Security levels (or sensitivity labels) are equal if the hierarchical level of both labels are equal and the non-hierarchically category sets are equivalent.
- Evaluation Assurance Level (EAL) — A package consisting of assurance components from Common Criteria (CC) Part 3 that represents a point on the CC predefined assurance scale. EALs provide an increasing scale that balances the level of assurance obtained with the cost and feasibility of acquiring that degree of assurance.
- Incomparable — Security levels (or sensitivity labels) are incomparable if they are not equal and neither label is greater than the other.
- Least Privilege — Principle that requires that each subject be granted the most restrictive set of privileges needed for the performance of authorized tasks.
- Mandatory Access Control (MAC) — A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity.
- Multilevel Security (MLS) — Concept of processing information with different classifications and categories that simultaneously permits access by users with different security clearances, but prevents users from obtaining access to information for which they lack authorization.
- Multilevel Secure RDBMS — A multilevel secure (MLS) RDBMS can store and process information at multiple security levels and serve multiple users, some of whom may not be cleared for all the information in the machine.

- Platform — The combination of software, hardware, and/or firmware layers underlying the DBMS.
- Relation — A relation is a two-dimensional (row and column) table of related data. Each table can contain different kinds of data.
- Relational Database Management System (RDBMS) — A relational database management system stores and manipulates data in the form of relations.
- Resource — Anything used or consumed while performing a function. The categories of resources are: time, information, objects (information containers), or processors (the ability to use information). Specific examples are: CPU time; terminal connect time; amount of directly-addressable memory; disk space; number of I/O requests per minute, etc.
- Security Level — The combination of a hierarchical sensitivity level and a set of non-hierarchical categories that represents the sensitivity of information. (TCSEC [3])
- Security Relevant Event — Any event that attempts to change the security state of the system, (e.g., change discretionary access controls, change the security level of the subject, change user password, etc.). Also, any event that attempts to violate the security policy of the system, (e.g., too many attempts to login, attempts to violate the mandatory access control limits of a device, attempts to downgrade a file, etc.).
- Sensitivity Label — A piece of information that represents the security level of an object that describes the sensitivity (e.g., classification) of the data in the object. (TCSEC [3]) Sensitivity labels are used by the TOE security functions as the basis for mandatory access control decisions.
- Sensitivity Level — A set of classifications (UNCLASSIFIED, CONFIDENTIAL, SECRET, TOP SECRET) together with their hierarchical relation defining the allowed information flows between levels.
- Security Function Policy (SFP) — The security policy enforced by a security function (SF).
- Strength of Function (SOF) — A qualification of a TOE security function expressing the minimum efforts assumed necessary to defeat its expected security behavior by directly attacking its underlying security mechanisms.
- Subject - An active entity, generally in the form of a person, process, or device that causes information to flow among objects or changes the system state. Technically, a process/domain pair.
- Target of Evaluation (TOE) — An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.
- TOE, CEM Annex B, B.6.2 — *TOE* is the entity that is evaluated as defined by the ST. While there are cases where a TOE makes up the entire product, this need not be the case. The TOE may be a product, a part of a product, a set of products, a unique technology

never to be made into a product, or combinations of all of these, in a specific configuration or set of configurations. This specific configuration or set of configurations is called the *evaluated configuration*. The ST clearly describes the relation between the TOE and any associated products.

- TOE Security Functions (TSF) — A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the TSP.
- TOE Security Functions Interface (TSFI) — A set of interfaces, whether interactive (man-machine interface) or programmatic (application programming interface), through which TOE resources are accessed, mediated by the TSF, or information is obtained from the TSF.
- TOE Security Policy (TSP) — A set of rules that regulate how assets are managed, protected and distributed within a TOE.
- TSF data — Data created by and for the TOE that might affect the operation of the TOE.
- TSF Scope of Control (TSC) — The set of interactions that can occur with or within a TOE and are subject to the rules of the TSP.
- User — Any entity (human or machine) outside the TOE that interacts with the TOE.
  - Authorized DBMS user — A DBMS user who, in accordance with the DBMS security policy (i.e., the TSP), may perform a DBMS operation.
  - Database Administrator (DBA) — A default DBMS administrative role that is explicitly granted all discretionary access authorizations to manage DBMS objects in the DBMS covered by the DBMS discretionary access control policy.
  - DBMS Audit Administrator (AUD) — A default DBMS administrative role that is limited to those authorizations necessary to administer and review the DBMS audit trail.
  - Non-administrative DBMS user — An authorized DBMS user who does not possess any trusted DBMS administrative roles (e.g. database operator, DBMS audit administrator, database administrator, DBMS security administrator).
  - Database Operator (OP) — A default database administrative role whose privileges are limited to those authorizations necessary to operate the DBMS.
  - DBMS Security Administrator (SA) — A default DBMS administrative role that is explicitly granted the authorizations to cause an information flow for information covered by the DBMS mandatory access control policy in order to perform administrative tasks operating on labeled information at multiple levels of security within the DBMS.
  - DBMS user — Any entity (human or machine) outside the DBMS TOE that interacts with the DBMS TOE.
- User ID – A character string used to denote a user of the system.

1.6 Acronyms

AUD	The default DBMS audit administrator (Auditor) defined by the `rubix.audit.*` authorizations
CC	Common Criteria
CLI	Call Level Interface
COTS	Commercial off the Shelf
DAC	Discretionary Access Control
DBA	The default database administrator role defined by the `rubix.dac.*` and `rubix.admin.*` authorizations.
DBMS	Database Management System
EAL	Evaluation Assurance Level
ISQL	Interactive Structured Query Language
IT	Information Technology
I&A	Identification and Authentication
MAC	Mandatory Access Control
MLS	Multi Level Security
OAN	Operational Area Network
OP	The default database operator role defined by the `rubix.restore.backup.*`, `rubix.`restore.logs.ls..*, `rubix.restore.logs.rm.*,` and `rubix.restore.logs.set.*` authorizations.
OS	Operating System
PP	Protection Profile
RDBMS	Relational Database Management Systems
SA	The default DBMS security administrator role defined by the `rubix.mac.*`, `rubix.restore.create.*`, and `rubix.*.grant` authorizations.
SAR	Security Assurance Requirement
SF	Security Function
SFP	Security Function Policy
SFR	Security Functional Requirement
SOF	Strength of Function
SQL	Structured Query Language
ST	Security Target
TCSEC	Trusted Computer System Evaluation Criteria
TOE	Target of Evaluation

TS/SCI	Top Secret/Sensitive Compartmental Information
TSC	TSF Scope of Control
TSF	Target of Evaluation Security Function
TSFI	TOE Security Functions Interface
TSP	TOE Security Policy

2 Target of Evaluation (TOE) Description

The need for information security and trust in computer systems is described in terms of three fundamental goals: confidentiality, data integrity, and controlled access. Confidentiality is a concept of holding sensitive data in confidence, limited to an appropriate set of individuals or organizations. Information is not made available or disclosed to unauthorized individuals, entities, or processes. Integrity assures that information and programs are changed only in a specified and authorized manner, that computer resources operate correctly, and that the data in them is not subject to unauthorized changes. Data integrity is a state that exists when computerized data is in sound, unimpaired or perfect condition and has not been exposed to accidental or malicious alteration or destruction. Controlled access protection is the process of limiting access to the resource of a system only to authorized programs, processes, or other systems (in a network). A security policy (SP) is the framework within which an organization establishes the security rules, procedures, practices, or guidelines for its needed levels of information security to achieve, among other things, the fundamental goals.

To accomplish the goal of controlled access protection to users' sensitive information in a database, Trusted RUBIX provides multilevel security (MLS) in addition to Discretionary Access Control. Trusted RUBIX multilevel security consists of two primary features: labeling and mandatory access controls. Labeling means that all Trusted RUBIX objects (files, processes, tables and rows) have a label that indicates the sensitivity of that object. Mandatory Access Control restricts access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity. Mandatory Access Control prevents anyone other than a security administrator from managing data access authorization in Trusted RUBIX. This is in strong contrast to discretionary access controls where users have the ability to control access to their Trusted RUBIX objects at their discretion.

Trusted RUBIX 5.0 is a multilevel secure (MLS) Relational Database Management System (RDBMS) for the UNIX® environment. Trusted RUBIX 5.0 provides a high level of security assurance and allows different levels of sensitivity data, (e.g., secret data, top secret/special intelligence data), to be represented by different sensitivity labels within a single database.

Trusted RUBIX 5.0 is an SQL-based relational DBMS product operating in a standalone or a client/server architecture as shown in Figure 1**.** Client processes are untrusted application programs that have been linked with Trusted RUBIX client software so that they can communicate with Trusted RUBIX server process. There is one instantiation of the server for each active client. A given client and server pair can run on the same machine or on different machines connected via a network. The server process must reside on the same machine as the data to be accessed.

Trusted RUBIX 5.0 conforms to Entry Level compliance of the SQL-92 industry standards and has many features that conform to Intermediate or Full compliance of the SQL-92 industry standards.

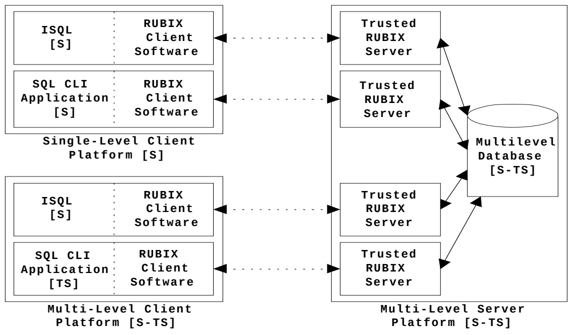


Figure 1 Trusted RUBIX Client/Server Architecture

All communication between client and server takes place on a single-level connection. Two types of clients are supported:

- Instantiations of an Interactive SQL (ISQL) interface that provides ad hoc access to databases; and
- User-developed applications utilizing the SQL Call-Level Interface (CLI), which is an alternative invocation technique to dynamic SQL that provides essentially equivalent operations. CLI is a set of functions that application programs call directly using normal call facilities.

2.1 TOE Physical Boundaries

Trusted RUBIX 5.0 is a MLS RDBMS software product. Its software consists of client software and server software. Client programs are applications linked with Trusted RUBIX client software, which provide the Application Programming Interfaces to the TOE. Trusted RUBIX server software includes Trusted RUBIX server process and a set of trusted programs. Trusted RUBIX server is a Relational DBMS engine, which interacts with physical data through Trusted Solaris 8 operating system files. The trusted programs are database security management utilities for the trusted DBMS administrative personnel.

The listed TOE configuration and documentation from Infosystems Technology, Inc. are the evaluated configuration that constitutes the TOE. The intent was to identify a configuration that consists of all relevant software packages and documentation, so that any unlisted software package and documentation would not be included to compromise a secure configuration.

2.1.1 TOE Configuration

Table 1 shows the evaluated configuration of Trusted RUBIX 5.0 TOE.

Table 1 Trusted RUBIX Evaluated Configuration

Trusted RUBIX Configuration	Evaluated Product Releases
Trusted RUBIX Server Version 5.0	***rxserver*** Version 5.0
Trusted RUBIX Client: ISQL Version 5.0 CLI Version 5.0	***isql*** Version 5.0 ***liblcli.a*** Version 5.0
Trusted RUBIX Administrative Commands Version 5.0	*rxauditrpt* Version 5.0 *rxauditset* Version 5.0 *rxdump* Version 5.0 *rxrestore* Version 5.0 *rximport* Version 5.0 *rxexport* Version 5.0 *rxformat* Version 5.0 *rxrecl* Version 5.0 *rxlogs* Version 5.0 *rxdb* Version 5.0
Platform Configuration	
Trusted Solaris 8™ on Intel Pentium Processors	

2.1.1.1 Trusted RUBIX Client Components

The Trusted RUBIX Client interacts with database users allowing them to execute SQL operations on a database. There are two types of clients: Interactive Structured Query Language (ISQL) and Call Level Interface (CLI). The first client type, ISQL client, provides a prompt driven, interactive interface where SQL operations may be typed in or read from a script file. The second client type, Call Level Interface, is a set of C language function calls that may be used to write application programs to operate on the database. Each client is a program that executes with the credentials and privileges of the initiating user. It may reside on the same machine as the server or on a different machine. The Trusted RUBIX Client component does not perform any security-relevant function. It simply interacts with the Trusted RUIX Server Process.

2.1.1.2 Trusted RUBIX Server Components

For the evaluation configuration of Trusted RUBIX 5.0, the underlying operating system platform selected is Trusted Solaris^tm 8 on an Intel Pentium processor. The Trusted RUBIX Server is a program that directly operates on the database. It is a trusted program that is responsible for enforcing the security policies. The server must reside on the same machine as the database to be operated on; however, the clients may reside on other machines. There is a one-to-one mapping between active clients and active servers.

(A) Trusted RUBIX Server Process
The Trusted RUBIX server process consists of the following major subsystems.

The Server Interface Subsystem is an interface used by the Client subsystems to connect and disconnect to specific databases, start and terminate transactions, manipulate savepoints, and execute SQL operations. The Server Interface Subsystem parses the SQL command text into the query tree, optimizes them for performance, and submits them to the SQL Engine Subsystem.

The SQL Engine Subsystem checks for needed DAC privileges before submitting the operations on record files to the kernel subsystem.

The Kernel Subsystem is responsible for enforcing all MAC restrictions on data objects. This subsystem also provides low-level transaction and database operations. The Kernel Subsystem interacts with the Common Server Subsystem.

The Common Server Subsystem in general interacts with the operating system and provides functions that require shared memory to operate efficiently, such as the main memory page buffering mechanism.

2.1.1.3 Trusted RUBIX Administrative Commands

The Trusted RUBIX administrative commands are generally executed only by privileged users and used to perform administrative duties (e.g., database backup and restore), which the typical user would not be required to perform. The administrative commands are a single process on the server executed from the command line and use a user's authorizations to determine their ability to execute. Trusted RUBIX Administrative Commands include:

- Audit Event Selection – *`rxauditset`*
- Audit Record Review – *`rxauditrpt`*
- Database Import – *`rximport`*
- Database Export – *`rxexport`*
- Database Dump – *`rxdump`*
- Database Drop, List, or Move – *`rxdb`*

- Database Restore – *rxrestore*
- Reclassify Row – *rxrecl*
- Manage Audit/Restore Log Files – *rxlogs*
- Alter Database Session Label – SQL command *ALTER SESSION SET LABEL*

2.1.2 TOE Documentation

The following Trusted RUBIX 5.0 documents are considered to be part of the TOE:

Trusted RUBIX 5.0 Configuration Management Manual

This document describes the Configuration Management Plan and Configuration Management procedures for the TOE evaluated configuration product and evaluation documents. The Release Creation section in the Configuration Management manual describes the procedures necessary for installation and generating the binary representation of the TOE.

Trusted RUBIX 5.0 Delivery and Operation Manual

This document provides the procedures used to deliver both partial and complete instantiations of the TOE to the user. These procedures provide the mechanisms used to detect discrepancies between the developer's master copy of the TOE and the version received at the delivery location, and counter masquerade attempts.

Trusted RUBIX 5.0 Functional Specification

This document describes the functional interface to the TOE giving special detail to those functions that are security relevant. TOE Security Functions are completely described while non-TOE Security Functions are described to the extent that it is clear they do not access any TSF.

Trusted RUBIX 5.0 High Level Design

This document describes major subsystems utilized for the execution of appropriate security policies, including the mandatory access control (MAC) and discretionary access control (DAC) security policies.

Trusted RUBIX 5.0 Low Level Design

This document describes the hierarchical modular structure utilizing abstractions implemented by procedures and state data for the Trusted RUBIX 5.0 system.

Trusted RUBIX 5.0 Security Functions to Functional Specification Correspondence

This document provides a complete mapping of all security functions identified in the TOE Summary Specification to the less abstract security function defined in the Functional Specification. The rationale demonstrates that the FSP is an accurate and complete representation of the security functions in the ST.

Trusted RUBIX 5.0 Functional Specification to High Level Design Correspondence

This document provides a complete mapping of all security functions identified in the Functional Specification to the appropriate subsystem in the High-Level Design. The rationale demonstrates that the HLD is an accurate and complete representation of the TOE security functional interfaces in the FSP.

Trusted RUBIX 5.0 High Level Design to Low Level Design Correspondence

This document provides a complete mapping of all security functions allocated to subsystems in the High Level Design to the less abstract security function defined in the Low Level Design. The rationale demonstrates that the LLD is an accurate and complete representation of the HLD.

Trusted RUBIX 5.0 Low Level Design to Implementation Correspondence

The implementation representation will only be provided for a subset of the TSF. This document provides a complete mapping between the low-level design and the security functionality that is presented in the implementation representation.

Trusted RUBIX 5.0 Security Policy

This document models the MAC and DAC policies and describes the interactions between all subjects and objects within the scope of MAC and DAC. This document also models the supporting policies, e.g., identification, security audit, object reuse, and roles of security management.

Trusted RUBIX 5.0 Security Features User's Guide

This document describes the security protections provided to users. It details the uses and interactions of the various security protections with examples.

Trusted RUBIX 5.0 Trusted Facility Manual

This document describes functions and privileges to be controlled when operating a trusted facility, the security policy, the audit mechanism, and other security procedures. It is primarily intended for the security and administrative personnel responsible for managing the trusted aspects of the system.

Trusted RUBIX 5.0 Life Cycle Support Document

This document describes the software development life-cycle model, security measures, and development tools used to construct the TOE. The software life-cycle processes and procedures described in the document provide the assurance that software is developed and maintained without adverse impact to TOE security functions.

Trusted RUBIX 5.0 Software Test

The sections 3, 4, and 5 of the Trusted RUBIX 5.0 Software Test document describes the plan for testing performed by the SQL regression test suites of Trusted RUBIX 5.0 relational database management system and associated functional interface specifications.

The sections 6, 7, 8, and 9 of the Trusted RUBIX 5.0 Software Test document contain the test procedure descriptions of all test suites of the Trusted RUBIX 5.0 relational database management system and associated with the external and internal functional interfaces.

Trusted RUBIX 5.0 Misuse Analysis

The Trusted RUBIX 5.0 Misuse Analysis analyzes whether the issue of misuse has been addressed in the user guidance, the administrator guidance, and the secure installation, generation, and start-up procedures.

Trusted RUBIX 5.0 Vulnerability Analysis

The developer Trusted RUBIX 5.0 Vulnerability Analysis documents an analysis of the TOE deliverables searching for obvious vulnerabilities and the disposition of these vulnerabilities, so that the vulnerability cannot be exploited in the intended environment for the TOE.

2.2 TOE Logical Boundaries

Trusted RUBIX 5.0 has many high assurance security features, which includes the following:

- Discretionary Access Control--Trusted RUBIX provides Discretionary Access Control (DAC), which restricts access to the objects based on the identity of the subjects and/or groups to which they belong. The Trusted RUBIX DAC policy is implemented with an Access Control List (ACL) associated with each protected DBMS resource.
- Mandatory Access Control--Trusted RUBIX offers a high level of access restriction through Mandatory Access Control (MAC), which restricts access to data objects based on the sensitivity of the information contained in the objects and the "clearance" of users to access such information. Trusted RUBIX is enhanced by the label-based MAC policy based on the Bell-LaPadula [1] model. Trusted RUBIX's MAC policy for sensitivity is implemented with labels. Each protected DBMS resource has associated with it a sensitivity label that consists of a hierarchical level and a set of non-hierarchical categories. These labels are used to determine access to a resource in accordance with the MAC policy. This policy defines a dominance relationship between the labels.
- Security Audit--Trusted RUBIX provides a security audit (Audit) function that recognizes and records security relevant activities, both legitimate (but accidental) errors by users and unauthorized requests. It also provides audit utilities for authorized administrative personnel to perform auditable event selection and audit trail query and examination. A privileged user can use these utilities to determine which security relevant activities took place and who (which user) was responsible for them.

- Object Reuse--Trusted RUBIX performs object reuse and ensures that when rows, tables, views, and schemas are dropped, there is no deleted information that is accessible to a database user.
- Import and Export Data--Trusted RUBIX provides secured import and export operations enabling the user to load data into the database, and extract data from the database into a text file. If desired, a privileged user can import and load multilevel data at specified levels into the database.
- Trusted Recovery--Trusted RUBIX provides trusted recovery to a consistent and secure state from transaction failure and/or system failure. Trusted RUBIX also provides backup and restore facilities to ensure the capability to restore the database as a whole after a primary disk error.
- Security Management--Trusted RUBIX also provides a set of trusted programs (e.g., audit event selection and audit trails review, backup and restore facilities) for administrative personnel to manage audit criteria and analyze audit trails and to restore the database as a whole.

Trusted RUBIX 5.0 has the following advanced relation DBMS features for database performance and integrity:

- A savepoint mechanism that allows transactions to be partially rolled back on user request. Thus, a user can undo all updates performed since the specified savepoint, while at the same time preserve updates performed prior to that point.
- Trusted RUBIX's unique multi-version time-stamping concurrency control technique enables the system to securely serialize all changes taking place within the database, even with multiple applications running. Thus, multiple concurrent users are presented with a consistent view of the data while performance is maintained.
- A relational DBMS architecture, which supports large, complex databases with an unlimited number of simultaneously open tables, views, and indexes.

2.3 External IT Entity Supports

This section addresses the required secure computing environment, which is supported by the operating system (OS) and other trusted products as the basis for the secure operation of Trusted RUBIX. The dependencies of Trusted RUBIX 5.0 on the Sun Microsystems’ Trusted Solaris 8 operating system and other external IT entities are listed below.

Management of Trusted RUBIX Users and Groups

Trusted RUBIX relies on the Trusted Solaris 8 security administrators to

- Grant DBMS users appropriate authorizations for different DBMS security management roles,

- Create the Trusted RUBIX users' accounts and assign users to the appropriate DBMS user groups.
- Add authorizations to the Trusted Solaris 8 security database, create directories to hold executable and data files, set the permissions and sensitivity label on the directories, copy and assign privileges to executable programs.

Identification and Authentication

Trusted RUBIX relies on the OS to perform user identification and authentication (I&A) prior to allowing users to execute DBMS commands and/or applications. Trusted RUBIX also relies on the OS to perform user-subject binding, when a user initiates a trusted or untrusted command or DBMS application. In other words, the process, initiated by the user, will have the user's User ID, Group ID, and current session sensitivity label as its subject security attributes. Depending on the configuration of the Trusted RUBIX Server the user may also be authenticated using the password maintained by the underlying operating system during the database server or administrative command startup.

Access Control

Trusted RUBIX relies on the OS to perform mandatory access control and discretionary access control and enforce its MAC and DAC security policies to protect the operating system's named objects, which include the directories, files, memory, devices, and sockets created and/or used by Trusted RUBIX.

Audit

Trusted RUBIX relies on the OS to

- Provide reliable time for time stamping the occurrences of the audit events.

Object Reuse

Trusted RUBIX relies on the OS to ensure that any residual information content of any resource is unavailable to all operating system objects upon the resource's allocation or deallocation.

Secure Operational Area Network

Trusted RUBIX relies on the TOE environment for its client/server communications security, which includes crypto security, transmission security, emission security, and physical security of communications security (COMSEC) material. Communications security material includes, but is not limited to, key, equipment, devices, documents, firmware or software that embodies or describes cryptographic logic and other items that perform communications security functions.

3 TOE Security Environment

This section describes the security aspects of the environment in which the TOE is intended to be used and the manner in which it is expected to be employed. The TOE security environment identifies the assumptions for secure usage of the TOE, the threats to DBMS assets, and the security policies supported by the TOE.

This information provides the basis for the Security Objectives specified in Section 4 and the Security Requirements for the TOE and the IT Environment specified in Sections 5.

3.1 Secure Usage Assumptions

The TOE is dependent upon both technical assets and operational aspects of its environment.

3.1.1 Personnel Assumptions

A.ADMIN

It is assumed that database administrators, database operators, DBMS security administrators, and DBMS audit administrators are competent, and merit trust place in them.

A.USERS

It is assumed that authorized DBMS users are familiar with applicable DBMS security policies and procedures, and merit the trust placed in them.

3.1.2 Physical Assumptions

A.DISASTER

It is assumed that the DBMS is protected against disasters such as loss of power, fire, flood, and destruction of facilities.

A.PHYSICAL

It is assumed that the DBMS, host OS, and IT environment are protected from physical attack.

3.1.3 Connectivity Assumptions

A.SECURE_COMMS

It is assumed that the environment protects information while it is in transit between the DBMS and components of the IT environment.

3.2 Threats to Security

The assumed threats to TOE security are specified below. Each threat statement identifies a means by which the TOE, the IT environment, or the DBMS assets might be compromised.

DBMS assets consist of the information or resources that the DBMS protects. DBMS assets include the following:

- The DBMS server software (including the security functions that it performs)
- The DBMS client software that provides various interfaces with which authorized users access the DBMS and the data under its management
- The user data and DBMS data (as described in section 2.1) that the DBMS maintains and protects

These threats will be countered by:

- Technical security measures provided by the TOE, the host OS, and the IT environment, in conjunction with
- Non-technical operational security measures (personnel, procedural and physical measures) in the environment.

Many of the threats below were adapted from threats in the *Multilevel Operating System* protection profile [MOS_MED PP]. The statements of the threats in this security target and the MOS_MED PP are not identical because of differences between a DBMS and an operating system. An asterisk (*) in a threat identifier marks a threat adapted from the MOS-MED PP.

T.ABUSE

An authorized DBMS user performs authorized actions that compromise (intentionally or otherwise) DBMS assets.

T.ADMIN_ERROR*

An attacker may exploit vulnerabilities in the DBMS caused by improper administration of the DBMS in order to compromise DBMS assets.

T.ADMIN_ROGUE*

A database administrator, DBMS security administrator, or DBMS audit administrator performs actions that intentionally compromise DBMS assets.

T.AUDIT_CORRUPT*

An attacker may cause audit records to be lost or modified, or may prevent future records from being recorded by taking actions to exhaust audit storage capacity, thus masking an attacker's actions.

T.DOS*

An attacker may exhaustively consume IT environment resources in order to deny DBMS assets to authorized DBMS users.

T.EAVESDROP*

An attacker may gain unauthorized access to DBMS assets (e.g. authentication information or DBMS objects) when the data is transmitted to or from the DBMS.

T.EXPORT

An authorized DBMS user may send information (in soft or hard copy form) from DBMS objects to a recipient who is not authorized to see the information or who subsequently handles the information in a manner that is inconsistent with its sensitivity designation.

T.IMPROPER_INSTALLATION*

An attacker may exploit vulnerabilities in the DBMS caused by improper delivery, installation, or configuration of the DBMS in order to compromise DBMS assets.

T.INSECURE_START*

An attacker may exploit vulnerabilities in the DBMS created during start up or restart of the DBMS or host OS in order to compromise DBMS assets.

T.MASQUERADE*

An attacker or external IT entity may masquerade as an authorized DBMS user or a external IT entity in order to gain unauthorized access to DBMS objects or DBMS resources.

T.POOR_DESIGN*

An attacker may exploit vulnerabilities in the DBMS caused by unintentional or intentional errors in requirements specification, design or development in order to compromise DBMS assets.

T.POOR_IMPLEMENTATION*

An attacker may exploit vulnerabilities in the DBMS caused by unintentional or intentional errors in implementing the design of the DBMS in order to compromise DBMS assets.

T.REPLAY*

An attacker may gain unauthorized access to DBMS assets by replaying authentication information corresponding to an authorized DBMS user.

T.SPOOFING*

An attacker may masquerade as the DBMS or an external IT entity in the IT environment and communicate with authorized DBMS users who incorrectly believe they are communicating with the DBMS or external IT entity.

T.SYSACC*

An attacker may gain unauthorized access to the account of a database administrator, a database operator, a DBMS security administrator, a DBMS audit administrator, or other trusted personnel including IT environment administrators.

T.UNATTENDED_SESSION*

An attacker may gain unauthorized access to DBMS assets using an unattended session of an authorized DBMS user.

T.UNAUTH_ACCESS*

An attacker may gain unauthorized access to DBMS assets either via the DBMS itself or via the IT environment.

T.UNAUTH_MODIFICATION*

An attacker may make unauthorized modifications to the DBMS security policy data or unauthorized use of security functions.

T.UNDETECTED_ACTIONS*

In order to compromise DBMS assets, an attacker may successfully introduce vulnerabilities into the DBMS, or repeatedly exploit vulnerabilities in the DBMS, without being detected by the DBMS.

T.UNIDENTIFIED_ACTIONS*

In order to compromise DBMS assets, an attacker may successfully introduce vulnerabilities into the DBMS, or repeatedly exploit vulnerabilities in the DBMS, without being identified by the DBMS audit administrator.

T.UNKNOWN_STATE*

An attacker may exploit vulnerabilities in the DBMS created by a failure of the DBMS or host OS in order to compromise DBMS assets.

T.USER_CORRUPT*

An attacker may make unauthorized deletions or modifications to DBMS data.

Application note: In general, user authorizations are limited to specific DBMS objects (e.g., tables) and operations (e.g., read). Hence, the attackers in this threat may include authorized users attempting to modify data for which they do not have specific authorization (e.g., modifying another user's data).

In summary, the threats listed above represent two types of threats: threats that directly compromise DBMS assets and threats that cause vulnerabilities in TOE security mechanisms, which can then be used to compromise DBMS assets.

The direct threats to DBMS assets are:

- T.ABUSE
- T.ADMIN_ROGUE*
- T.DOS*
- T.EAVESDROP*
- T.EXPORT
- T.MASQUERADE*
- T.REPLAY*
- T.SPOOFING*
- T.SYSACC*
- T.UNATTENDED_SESSION*
- T.UNAUTH_ACCESS*
- T.USER_CORRUPT*

The threats that target TOE security mechanisms are:

- T.ADMIN_ERROR*
- T.AUDIT_CORRUPT*
- T.IMPROPER_INSTALLATION*
- T.INSECURE_START*
- T.POOR_ DESIGN*
- T.POOR_IMPLEMENTATION*
- T.UNAUTH_MODIFICATION*
- T.UNDETECTED_ACTIONS*
- T.UNIDENTIFIED_ACTIONS*
- T.UNKNOWN_STATE*

Four of the threats to TOE security mechanisms are countered entirely by assurance and non-IT measures, namely, T.ADMIN_ERROR*, T.IMPROPER_INSTALLATION*, T.POOR_DESIGN*, and T.POOR_IMPLEMENTATION*.

3.3 Organizational Security Policies

An organizational security policy is a set of rules or procedures imposed by an organization upon its operations to protect its sensitive data. Although the security policies described below are drawn from DOD sources they apply to many non-DOD environments.

The organizational security policies below were adapted from organizational security policies in the *Multilevel Operating System* protection profile [MOS_MED PP]. The statements of the policies in this security target and the MOS_MED PP are not identical because of differences between a DBMS and an operating system. For consistency with assumptions and threats, an asterisk (*) in a policy identifier marks an organizational security policy adapted from the MOS-MED PP.

P.ACCOUNT*

The users of the DBMS shall be held individually accountable for their actions within the DBMS.

> *Application note: P.ACCOUNT* is related to T.ABUSE* in that the security objectives used to address the policy also counter the threat.*

P.AUTHORIZATION*

The DBMS must limit the extent of each user’s abilities in accordance with the DBMS security policy.

P.AUTHORIZED_USERS*

Only those users who have been authorized to access the information within the DBMS may access the DBMS and users will be granted authorization in accordance with the principle of least privilege.

> *Application note: P.AUTHORIZED_USERS* is related to T.MASQUERADE* in that the security objectives used to address the policy also counter the threat.*

P.CLEARANCE*

The DBMS must limit access to the protected DBMS assets to authorized users whose security level is appropriate for the labeled data.

> *Application note: The DBMS limits access (P.CLEARANCE*) to information based on the clearances of users (P.USER_CLEARANCE*) and the sensitivity of information (P.RESOURCE_LABELS*).*

P.INDEPENDENT_TESTING*

The DBMS must undergo independent security functional and vulnerability testing as part of an independent vulnerability analysis.

P.NEED_TO_KNOW*

The DBMS must limit the access to, modification of, and destruction of the information in DBMS assets to those authorized users who have an explicated stated, “need to know” for that information.

P.RESOURCE_LABELS*

All DBMS assets must have associated labels identifying the security level of the data contained therein.

P.ROLES*

The database administrator, database operator, DBMS security administrator, and DBMS audit administrator shall have separate and distinct roles associated with them.

> *Application note: These roles need not be built into the DBMS product, but rather may be configured as part of the TOE installation process.*

P.TRUSTED_RECOVERY*

After a DBMS failure or other operational discontinuity, the DBMS shall recover without a protection compromise. In all cases DBMS operation must begin in a secure state.

P.USER_CLEARANCE*

Each user must have a clearance identifying the maximum security level of data that the user may access.

4 Security Objectives

4.1 Introduction

Security objectives describe the means by which the TOE will counter identified threats, meet identified organizational security policies, and address assumptions. Security objectives may be objectives for the TOE or objectives for the environment—either the IT environment or the non-IT environment. The identifier of a security objective indicates which type of objective it is: O.TOE for objectives for the TOE, O.IT_ENV for objectives for the IT environment, and O.ENV for objectives for the non-IT environment.

An asterisk (*) in a security objective identifier marks an objective based on a security objective in the *Multilevel Operating System* protection profile [MOS_MED PP].

4.2 Security Objectives for the TOE

This section identifies the security objectives that are to be met by TOE security functions.

O.TOE_ADMIN_GUIDANCE

To provide database administrators, database operator, DBMS security administrators, and DBMS audit administrators with the guidance documentation necessary for secure management of the DBMS.

O.TOE_ADMIN_ROLE*

To isolate administrative capabilities by providing separate and distinct security management roles associated with authorized DBMS users.

O.TOE_SELF_PROTECTION*

To ensure that the security functions of the DBMS cannot be bypassed, and to ensure that the security functions have a separate execution domain, which protects from interference and tampering by untrusted DBMS subjects.

O.TOE_USER_IDENTIFICATION*

To uniquely identify DBMS users.

O.TOE_DISCRETIONARY_ACCESS*

To control accesses to DBMS assets based upon the identity of DBMS users and groups of DBMS users.

O.TOE_MANDATORY_ACCESS*

To control accesses to DBMS assets based upon the security level of users and the sensitivity of the DBMS assets.

O.TOE_USER_GUIDANCE

To provide authorized DBMS users with the guidance documentation necessary for secure use of the DBMS.

O.TOE_RESIDUAL_INFORMATION*

To ensure that any information contained in a protected DBMS object or resource is not released when the DBMS object or resource is reallocated.

O.TOE_AUDIT_GENERATION*

To provide the capabilities to detect security relevant events, create records of such events, and associate each record with the individual user who causes the event.

O.TOE_AUDIT_REVIEW*

To provide the capability to selectively view audit information.

O.TOE_AUDIT_PROTECTION*

To provide the capability to protect audit information.

O.TOE_DEVEL_CM*

To track and control all changes to the DBMS and its development evidence during development.

O.TOE_RECOVERY*

To provide procedures and/or mechanisms to assure that, after a DBMS failure, host OS failure, or other discontinuity, the DBMS recovers without a protection compromise.

O.TOE_SOUND_DEVELOPMENT*

To apply, and accurately document, sound design and implementation principles and techniques to the development of the DBMS.

O.TOE_TESTING*

To independently test the DBMS security functions both for conformance to the DBMS design and for vulnerabilities.

4.3 Security Objectives for the Environment

This section identifies the security objectives that are to be met by the environment in which the TOE is intended to operate.

4.3.1 Security Objectives for the IT Environment

The following objectives are expected to be met by IT products and mechanisms other than the TOE.

O.IT_ENV_SELF_PROTECTION*

To protect the host operating system and its assets from external interference, tampering, or unauthorized disclosure.

O.IT_ENV_TIME

To provide reliable time stamps.

O. IT_ENV_AUDIT_PROTECTION*

To provide the capability to protect audit information.

O. IT_ENV_USER_AUTHENTICATION*

To verify the claimed identity of a DBMS user.

O. IT_ENV_USER_IDENTIFICATION*

To uniquely identify DBMS users.

4.3.2 Security Objectives for the Non-IT Environment

The following five objectives actually re-state the secure usage assumptions described in section 3.1. It is expected that they will be met by policies, procedures, physical protections, and other non-IT mechanisms.

O.ENV_ADMIN

To adequately train host OS administrators (e.g., security administrators, audit administrators, and operators) and DBMS administrators (e.g., database administrators, DBMS security administrators, and DBMS audit administrators), and to obtain appropriate evidence that these administrators are trustworthy.

O.ENV_CONFIG

To install, configure, manage, and maintain the TOE in accordance with its guidance documentation and the applicable security policies and procedures.

Application note: An important aspect of managing the TOE is keeping DBMS user account information current. This includes removing accounts in cases of job termination and changing authorizations in response to changes in user responsibilities.

O.ENV_DISASTER

Those persons responsible for the DBMS and its IT environment shall provide appropriate protection against disasters such as loss of power, fire, flood, and destruction of facilities.

O.ENV_PHYSICAL

Those responsible for the DBMS and its IT environment shall provide appropriate physical security (including preventing unauthorized physical access) for the DBMS, the host OS, and IT environment in accordance with the value of the information stored, processed, and transmitted by the DBMS.

O.ENV_SECURE_COMMS

Those responsible for the TOE and the IT environment will provide measures to protect information while it is in transit between the DBMS and external IT entities in the IT environment.

O.ENV_TRUST_IT

To ensure that each external IT entity on which the TOE relies for security functions is installed, configured, managed, and maintained in a manner appropriate to the IT entity, the DBMS, and the relationship between them.

O.ENV_USERS

To adequately train authorized DBMS users, to obtain appropriate evidence that these users are trustworthy, and to have them acknowledge their obligation to follow applicable DBMS security policies and procedures.

5 IT Security Requirements

This section describes the functional and assurance security requirements for the TOE and for the IT environment.

5.1 TOE Security Functional Requirements

The CC permits four functional component operations: assignment, iteration, refinement, and selection to be performed on functional requirements. These operations are defined in Common Criteria, Part 2, paragraph 2.1.4 as:

- assignment: allows the specification of an identified parameter;
- refinement: allows the addition of details or the narrowing of requirements;
- selection: allows the specification of one or more elements from a list; and
- iteration: allows a component to be used more than once with varying operations.

The formatting of Common Criteria Security Functional Requirements (SFRs) is based on the style listed in Table 2 below. The ST author uses the following notations to identify these operations performed on the CC SFRs:

- *Assignment* operations completed by the ST author are denoted in **Bold** font.
- *Selection operation*s completed by the ST author are denoted in Underline font.
- *Refinements* are identified with "**Refinement**" in superscript font written at the end of the refined SFR. The refined texts are typed in **Bold** & Underline font. They permit the addition of extra detail when the component is used. The underlying notion of a refinement is that of narrowing. There are two types of narrowing possible: narrowing of implementation and narrowing of scope.
- *Iterations* are identified with a number (e.g., inside parentheses ("(#)"). These follow the short family name and allow components to be used more than once with varying operations.

Table 2 Typeset Convention for Security Functional Requirements

Convention	Purpose	Operation
Bold & Underline	The purpose of bolded and underlined text is to alert the reader that the ST author has added new text to the CC and/or PP text in order to **refine** component.	ST Refinement

Convention	Purpose	Operation
Bold	The purpose of sans serif text is to alert the reader that text has been added to the CC text in order to complete an **assignment** operation.	ST author Completed Assignment
Underline	The purpose of underlined text is to inform the reader that a choice was made from a list provided by the CC **selection** operation statement.	ST author Completed Selection
Parentheses (Iteration #)	The purpose of using parentheses and an iteration number is to inform the reader that the PP author has selected a new field of assignments or selections with the same requirement and that the requirement will be used multiple times.	ST Iteration

5.1.1 Security audit (FAU)

5.1.1.1 FAU_GEN.1 Audit data generation

FAU_GEN.1.1 The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;

b) All auditable events for the basic level of audit; and

c) **All audit events listed in Table 3 below**.

FAU_GEN.1.2-NIAP-0347 The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, **the additional information specified in the Additional Data column of Table 3**.

Table 3 FAU_GEN.1 Auditable Events

Component	Event	Additional Data

Component	Event	Additional Data
FAU_GEN.1 Audit data generation	Start-up and shutdown of DBMS audit functions	
FAU_GEN.2 User Identity Association	None	
FAU_SAR.1 Audit review	Reading of information from the audit records.	
FAU_SAR.2 Restricted audit review	Unsuccessful attempts to read information from the audit records.	
FAU_SAR.3 Selectable audit review	None	
FAU_SEL.1 Selective audit	All modifications to the audit configuration that occur while the audit collection functions are operating.	
FAU_STG.1	None	
FAU_STG.4	Actions taken due to the audit storage failure.	
FDP_ACC.2 Complete Access Control	None	
FDP_ACF.1 Attribute based access control	All requests to perform an operation on an object covered by the SFP.	The identity of the object.
FDP_ETC.1 Export of user data without security attributes	All attempts to export information.	
FDP_ETC.2 Export of user data with security attributes	All attempts to export information.	
FDP_IFC.2 Complete information flow control	None	

Component	Event	Additional Data
FDP_IFF.2 Hierarchical security attributes	All decisions on requests for information flow.	
FDP_ITC.1 Import of user data without security attributes	All attempts to import user data.	
FDP_ITC.2 Import of user data with security attributes	All attempts to import user data, including any security attributes.	
FDP_RIP_DB.2 TOE full residual information protection	None	
FDP_ROL.2 Advanced rollback	All attempts to perform rollback operations.	
FIA_ATD.1 User attribute definition	None	
FIA_USB.1 User-subject binding	Success and failure of binding user security attributes to a subject (e.g. success and failure to create a subject).	
FMT_MOF.1 Management of security functions behavior	All modifications in the behaviour of the functions in the TSF.	
FMT_MSA.1 Management of security attributes	All modifications of the values of security attributes.	
FMT_MSA.2 Secure security attributes	All offered and rejected values for a security attribute.	
FMT_MSA.3 Static attribute initialization	Modifications of the default setting of permissive or restrictive rules. All modifications of the initial value of security attributes.	
FMT_MTD.1	All modifications to the values of TSF data.	The new value of the TSF data.

Component	Event	Additional Data
FMT_REV.1 Revocation	All attempts to revoke security attributes.	
FMT_SMF.1 Specification of Management Functions	Use of the management functions	
FMT_SMR.1 Security roles	Every use of the rights of a role.	The role and the origin of the request.
FPT_RCV.4 Function recovery	If possible, the detection of a failure of a security function.	
FPT_RVM_DB.1 TOE non-bypassability of the TSP	None	
FPT_SEP_DB.1 Partial TSF domain separation	None	

5.1.1.2 FAU_GEN.2 User Identity Association

FAU_GEN.2.1-NIAP-0410 For audit events resulting from actions of identified users, the TSF shall be able to associate each auditable event with the identity of the user that caused the event.

5.1.1.3 FAU_SAR.1 Audit review

FAU_SAR.1.1 The TSF shall provide **the DBMS audit administrators** with the capability to read **all DBMS audit information** from the **DBMS** audit records. Refinement

FAU_SAR.1.2 The TSF shall provide the **DBMS** audit records in a manner suitable for the user to interpret the information. Refinement

5.1.1.4 FAU_SAR.2 Restricted audit review

FAU_SAR.2.1 The TSF shall prohibit all users read access to the audit records, except those users that have been granted explicit read-access.

5.1.1.5 FAU_SAR.3 Selectable audit review

FAU_SAR.3.1 The TSF shall provide the ability to perform searches of **DBMS** audit data based on Refinement

a) **Event status (success/failure)**

b) **Database name**

5.1.1.6 FAU_SEL.1 Selective audit

FAU_SEL.1.1 The TSF shall be able to include or exclude auditable **DBMS** events from the set of audited events based on the following attributes: Refinement

a) Database identity

b) User identity

c) Event type

d) **Object sensitivity label**

e) **Subject sensitivity label**.

5.1.1.7 FAU_STG.1 Protected audit trail storage

FAU_STG.1.1-NIAP-0422 The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

FAU_STG.1.2-NIAP-0423 The TSF shall be able to prevent unauthorized modifications to the audit records in the audit trail.

5.1.1.8 FAU_STG.4 Prevention of audit data loss

FAU_STG.4.1-NIAP-0429 The TSF shall ignore auditable events and **automatically terminate the Trusted RUBIX server process immediately** if the audit trail is full.

5.1.2 User Data Protection (FDP)

5.1.2.1 FDP_ACC.2 Complete Access Control

5.1.2.1.1 Complete Access Control on Database, Catalog, Schema, Table, View and Index

FDP_ACC.2.1(1) The TSF shall enforce the **Discretionary Access Control Policy** on **subjects: process acting on the behalf of users, and objects: databases, catalogs, schemas, tables, views, and indices** and all operations**: create, destroy, and list,** among subjects and objects covered by the SFP. Refinement

FDP_ACC.2.2(1) The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

5.1.2.1.2 Complete Access Control on Column in Table

FDP_ACC.2.1(3) The TSF shall enforce the **Discretionary Access Control Policy** on **subjects: process acting on the behalf of users, and object: a subset of columns in table** and all operations**: select, insert, delete, reference by foreign key, update, create view, and reference by view** among subjects and objects covered by the SFP. Refinement

FDP_ACC.2.2(3) The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

5.1.2.1.3 Complete Access Control on Column in View

FDP_ACC.2.1(4) The TSF shall enforce the **Discretionary Access Control Policy** on **subjects: process acting on the behalf of users, and object: a subset of columns in view** and all operations**: select, insert, delete, create view, and update** among subjects and objects covered by the SFP. Refinement

FDP_ACC.2.2(4) The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

Application Note: Delete a subset of columns in a row of a table or view results in deleting the entire row (i.e., all columns of the row).

5.1.2.1.4 Complete Access Control on Alter Table

FDP_ACC.2.1(5) The TSF shall enforce the **Discretionary Access Control Policy** on **subjects: process acting on the behalf of users, and objects: tables** and all operations**: alter table** among subjects and objects covered by the SFP. Refinement

FDP_ACC.2.2(5) The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

5.1.2.2 FDP_ACF.1 Access Control Functions

5.1.2.2.1 Create or Destroy Database

FDP_ACF.1.1-NIAP-0416(1) The TSF shall enforce the **Discretionary Access Control Policy** to **DBMS** objects based on the following: Refinement

a) **The subject (process) attributes:**
   - **Named individuals as specified by the process real UID**
     - **Real User ID (UID) used for privilege checks**
   - **Named group of individuals as specified by the process Real GID**
     - **Real Group ID (GID) used for privilege checks**

b) **The DBMS object attributes:**

- **Access control list (ACL) on DBMS objects.**

Application Note: An ACL entry specifies the operation privileges and the privileges for granting or revoking user operation privilege on a DBMS object for the user. Each ACL entry identifies its user by specifying the User ID, the Group ID, or Public.

FDP_ACF.1.2(1) The TSF shall enforce the following rules to determine if an operation**: create or destroy** among controlled subjects**, acting on the behalf of the users,** and controlled objects**: database** is allowed**: The operation is granted if the following rules are true**: Refinement

a) **Any subject is allowed to create a database. The subject must have the ADMINISTRATIVE privilege to destroy the database.**

Application Note: Database is contained in OS files, not in DBMS object. Database is the container object of catalogs.

FDP_ACF.1.3(1) The TSF shall explicitly authorize access of subjects to **DBMS** objects based on the following additional rules: **None**. Refinement

FDP_ACF.1.4(1) The TSF shall explicitly deny access of subjects to **DBMS** objects based on the following additional rules: **If an ACL entry explicitly grants the subject the NULL privilege, it results in the subject having no access to the object with which this ACL is associated**. Refinement

5.1.2.2.2 Create or Destroy Database Objects

FDP_ACF.1.1-NIAP-0416(2) The TSF shall enforce the **Discretionary Access Control Policy** to **DBMS** objects based on the following: Refinement

a) **The subject (process) attributes:**
   - **Named individuals as specified by the process real UID**
     - **Real User ID (UID) used for privilege checks**
   - **Named group of individuals as specified by the process real GID**
     - **Real Group ID (GID) used for privilege checks**

b) **The DBMS object attributes:**
   - **Access control list (ACL) on DBMS objects.**

FDP_ACF.1.2(2) The TSF shall enforce the following rules to determine if an operation**: Create or Destroy** among controlled subjects**: acting on the behalf of users,** and controlled objects: **catalog, schema, table, index, or view** is allowed**: The operation is granted if the following rules are true**: Refinement

a) **The subject must have EXECUTE privilege to all containing objects in database that hold the DBMS object, and**

b) **The subject must have WRITE privilege to the container object in**

database, from which the DBMS object is to be created or destroyed, and

c) **For creating a View object, the subject must also have CREATE VIEW privilege on columns of the tables and/or views, from which the View object is to be created (i.e., columns directly referenced in the view definition).**

d) **For creating a View object, the subject must also have REFERENCE VIEW privilege on columns of the underlying tables, from which the View object is to be created (i.e., the underlying table columns from which the view will access rows).**

e) **For creating a Table, the subject must also have REFERENCES privilege on all columns referenced by a foreign key constraint.**

Application Note: Destroy a DBMS object implies "Drop" operation for database, catalog, schema, index, table, and view objects and "Delete" operation on columns of row.

Application Note: Database is the container object of catalogs. Catalog is the container object of schemas. Schema is the container object of indices, tables, and views. Table and view are the container objects of columns. Table is the container object of rows.

FDP_ACF.1.3(2) The TSF shall explicitly authorize access of subjects to **DBMS** objects based on the following additional rules: **If the subject is a member of Database Administrator role, create or destroy DBMS object is allowed**. Refinement

FDP_ACF.1.4(2) The TSF shall explicitly deny access of subjects to **DBMS** objects based on the following additional rules: **If an ACL entry explicitly grants the subject the NULL privilege, it results in the subject having no access to the object with which this ACL is associated**. Refinement

5.1.2.2.3 Access Column in Table

FDP_ACF.1.1-NIAP-0416(3) The TSF shall enforce the **Discretionary Access Control Policy** to **DBMS** objects based on the following: Refinement

a) **The subject (process) attributes:**
   - **Named individuals as specified by the process real UID**
     - **Real User ID (UID) used for privilege checks**
   - **Named group of individuals as specified by the process real GID**
     - **Real Group ID (GID) used for privilege checks**

b) **The DBMS object attributes:**

- **Access control list (ACL) on DBMS objects.**

FDP_ACF.1.2(3) The TSF shall enforce the following rules to determine if an operation**: SELECT, UPDATE, INSERT, and/or DELETE** among controlled subjects**, acting on the behalf of the DBMS users,** and controlled objects**: a subset of columns in table** is allowed**: The operation is granted if the following rules are true**: Refinement

a) **The subject must have EXECUTE privilege to all containing objects that hold the DBMS object(s), and**

b) **The subject must have the SELECT, UPDATE, INSERT, and/or DELETE privileges for the requested access to the object.**

c) **The subject must have the SELECT on any columns referenced by a sub-query.**

d) **If a sub-query references a View object the subject must also have REFERENCE VIEW privilege on the underlying table columns from which the View object is created.**

FDP_ACF.1.3(3) The TSF shall explicitly authorize access of subjects to **DBMS** objects based on the following additional rules: **If the subject is a member of Database Administrator role, all accesses on table columns are allowed**. Refinement

FDP_ACF.1.4(3) The TSF shall explicitly deny access of subjects to **DBMS** objects based on the following additional rules: **If an ACL entry explicitly grants the subject the NULL privilege, it results in the subject having no access to the object with which this ACL is associated**. Refinement

5.1.2.2.4 Access Column in View

FDP_ACF.1.1-NIAP-0416(4) The TSF shall enforce the **Discretionary Access Control Policy** to **DBMS** objects based on the following: Refinement

a) **The subject (process) attributes:**
   - **Named individuals as specified by the process real UID**
     - **Real User ID (UID) used for privilege checks**
   - **Named group of individuals as specified by the process real GID**
     - **Real Group ID (GID) used for privilege checks**

b) **The DBMS object attributes:**
   - **Access control list (ACL) on DBMS objects.**

FDP_ACF.1.2(4) The TSF shall enforce the following rules to determine if an operation**: SELECT, UPDATE, INSERT, and/or DELETE** among controlled subjects**, acting on the behalf of the DBMS users,** and controlled objects**: a subset of**

columns in view is allowed**: The operation is granted if the following rules are true**: Refinement

a) **The subject must have EXECUTE privilege to all containing objects that hold the DBMS object,**

b) **The subject must have the SELECT, UPDATE, INSERT, and/or DELETE privileges for the requested access to the object, and**

c) **The subject must have REFERENCE VIEW privilege on all underlying table columns from which the View object is created.**

d) **For SELECT on a View object that is in the information schema, the subject must also have EXECUTE privilege to all containing objects of the object being listed and READ privilege on the object being listed.**

e) **The subject must have the SELECT on any columns referenced by a sub-query.**

f) **If a sub-query references a View object the subject must also have REFERENCE VIEW privilege on the underlying table columns from which the View object is created.**

FDP_ACF.1.3(4) The TSF shall explicitly authorize access of subjects to **DBMS** objects based on the following additional rules: **If the subject is a member of Database Administrator role, all accesses on view columns are allowed.** Refinement

FDP_ACF.1.4(4) The TSF shall explicitly deny access of subjects to **DBMS** objects based on the following additional rules: **If an ACL entry explicitly grants the subject the NULL privilege, it results in the subject having no access to the object with which this ACL is associated**. Refinement

5.1.2.2.5 List DBMS Internal Objects

FDP_ACF.1.1-NIAP-0416(5) The TSF shall enforce the **Discretionary Access Control Policy** to **DBMS** internal objects based on the following: Refinement

a) **The subject (process) attributes:**

- **Named individuals as specified by the process real UID**
  - **Real User ID (UID) used for privilege checks**
- **Named grpup of individuals as specified by the process real GID**
  - **Real Group ID (GID) used for privilege checks**

b) **The DBMS object attributes:**

- **Access control list (ACL) on DBMS objects.**

FDP_ACF.1.2(5) The TSF shall enforce the following rules to determine if an operation**: List a DBMS internal object** among controlled subjects and controlled objects is allowed: Refinement

a) **The subject must have EXECUTE privilege to all containing objects that hold the DBMS object,**

b) **The subject must have READ privilege to the container object, from which the DBMS object is to be read and listed, and**

c) **The subject must have EXECUTE privilege to the information schema , SELECT privilege to the referenced columns of the information schema view, and REFERENCE VIEW privilege on the underlying definition schema table columns from which the information schema view is created.**

Application Note: The Information Schema views gives users read access to the tables in the Definition Schema. The Definition Schema is where tables containing the attributes of the database objects (database, catalogs, schemas, views, tables, indexes, and constraints) reside. To list information of objects implies a search of the relevant information schema views. Each of the information schema views is built upon the SELECT statement on columns of tables in the definition schema.

FDP_ACF.1.3(5) The TSF shall explicitly authorize access of subjects to **DBMS** internal objects based on the following additional rules: **If the subject is a member of Database Administrator role, list a DBMS object is allowed**. Refinement

FDP_ACF.1.4(5) The TSF shall explicitly deny access of subjects to **DBMS** internal objects based on the following additional rules: **If an ACL entry explicitly grants the subject the NULL privilege, it results in the subject having no access to the object with which this ACL is associated**. Refinement

5.1.2.2.6 Alter Table

FDP_ACF.1.1-NIAP-0416(6) The TSF shall enforce the **Discretionary Access Control Policy** to **DBMS** objects based on the following: Refinement

a) **The subject (process) attributes:**

- **Named individuals as specified by the process real UID**
  - **Real User ID (UID) used for privilege checks**
- **Named group of individuals as specified by the process real GID**
  - **Real Group ID (GID) used for privilege checks**

b) **The DBMS object attributes:**

- **Access control list (ACL) on DBMS objects.**

FDP_ACF.1.2(6) The TSF shall enforce the following rules to determine if an operation**: Alter Table** among controlled subjects and controlled objects is allowed: Refinement

a) **The subject must have EXECUTE privilege to all containing objects that hold the DBMS object,**

b) **The subject must have EXECUTE and WRITE privileges to the schema of the table.**

c) **The subject must have REFERENCE privileges on any column referenced by a foreign key.**

FDP_ACF.1.3(6) The TSF shall explicitly authorize access of subjects to **DBMS** objects based on the following additional rules: **If the subject is a member of Database Administrator role, alter table is allowed**. Refinement

FDP_ACF.1.4(6) The TSF shall explicitly deny access of subjects to **DBMS** objects based on the following additional rules: **If an ACL entry explicitly grants the subject the NULL privilege, it results in the subject having no access to the object with which this ACL is associated**. Refinement

5.1.2.3 FDP_ETC.1 Export of user data without security attributes

FDP_ETC.1.1 The TSF shall enforce the **Discretionary Access Control policy and the Mandatory Access Control policy** when exporting user data, controlled under the SFP(s), outside of the TSC.

FDP_ETC.1.2 The TSF shall export the user data without the user data's associated security attributes.

5.1.2.4 FDP_ETC.2 Export of user data with security attributes

FDP_ETC.2.1(1) The TSF shall enforce the **Mandatory Access Control policy** when exporting user data, controlled under the SFP, outside of the TSC.

FDP_ETC.2.2(1) The TSF shall export the user data with the user data's associated security attributes**: the sensitivity labels**. Refinement

FDP_ETC.2.3(1) The TSF shall ensure that the security attributes**: the sensitivity labels**, when exported outside the TSC, are unambiguously associated with the exported user data. Refinement

FDP_ETC.2.4(1) The TSF shall enforce the following rules when user data is exported from the TSC:

a) **The subject session sensitivity label must dominate the table being exported**,

b) **Only table rows dominated by the subject session sensitivity label are exported**.

FDP_ETC.2.1(2) The TSF shall enforce the **Discretionary Access Control policy** when exporting user data, controlled under the SFP(s), outside of the TSC.

FDP_ETC.2.2(2) The TSF shall export the user data with the user data's associated security attributes**: the sensitivity labels**. Refinement

FDP_ETC.2.3(2) The TSF shall ensure that the security attributes**: the sensitivity labels**, when exported outside the TSC, are unambiguously associated with the exported user data. Refinement

FDP_ETC.2.4(2) The TSF shall enforce the following rules when user data is exported from the TSC:

a) **The subject must be a member of RUBIX Database Administrators, or**

b) **If the object is a Table, the subject must have EXECUTE privilege to all containing objects that hold the table, and SELECT privilege on all columns of the table.**

c) **If the object is a View that is not in the information schema, the subject must have EXECUTE privilege to all containing objects that hold the view, SELECT privilege on all columns of the view, and REFERENCE VIEW privilege on the underlying table columns.**

d) **If the object is a View that is in the information schema, the subject must have EXECUTE privilege to all containing objects that hold the view, SELECT privilege on all columns of the view, and REFERENCE VIEW privilege on the underlying table columns. In addition the subject must have EXECUTE privilege to all containing objects of the object being listed and READ privilege on the object being listed.**

Application Note: The Discretionary Access Control policy and the Mandatory Access Control policy are enforced when database objects are accessed for retrieving data to be exported.

5.1.2.5 FDP_IFC.2 Complete information flow control

FDP_IFC.2.1 The TSF shall enforce the **Mandatory Access Control policy** on **subjects: processes, acting on the behalf of the users, and information contained in the databases, catalogs, schemas, tables, views, indexes, and rows objects** and all operations that cause that information to flow to and from subjects covered by the SFP.

FDP_IFC.2.2 The TSF shall ensure that all operations that cause any information in the TSC to flow to and from any subject in the TSC are covered by an information flow control SFP.

5.1.2.6 FDP_IFF.2 Hierarchical security attributes

FDP_IFF.2.1-NIAP-0417 The TSF shall enforce the **Mandatory Access Control policy** based on the following types of subject and information security attributes:

a) **Sensitivity label of the subject, consisting of a hierarchical classification and a set of non-hierarchical categories; and**

b) **Sensitivity label of information, consisting of a hierarchical classification and a set of non-hierarchical categories.**

FDP_IFF.2.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules, based on the ordering relationships between security attributes hold:

a) **A subject may read a piece of information if the subject's sensitivity label dominates the sensitivity label of the information. (a read operation)**

b) **The subject can modify a piece of information if the subject session sensitivity label equals the sensitivity label of the information (a read and write operation).**

FDP_IFF.2.3 The TSF shall enforce the **<u>additional information flow control SFP rules:</u>**
Refinement

a) **When a piece of information is created it is labeled with the subject's sensitivity label.**

b) **When two pieces of information are combined, the sensitivity label of the resultant information must be chosen to dominate the sensitivity label of the original pieces of information.**

c) **When a piece of information is extracted from another piece of information, it inherits the sensitivity label of the original piece of information.**

d) **The sensitivity labels of all Mandatory Access Control protected objects in a container (object) must be equal to or greater than the sensitivity label of the container (object).**

FDP_IFF.2.4 The TSF shall provide the following **<u>additional information flow control SFP capabilities</u>: none.** Refinement

FDP_IFF.2.5 The TSF shall explicitly authorize an information flow based on the following rules: **none**.

FDP_IFF.2.6 The TSF shall explicitly deny an information flow based on the following rules: **none**.

FDP_IFF.2.7 The TSF shall enforce the following relationships for any two valid information flow control security attributes:

a) There exists an ordering function that, given two valid security attributes, determines if the security attributes are equal, if one security attribute is greater than the other, or if the security attributes are incomparable; and

b) There exists a “least upper bound” in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is greater than or equal to the two valid security attributes; and

c) There exists a “greatest lower bound” in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is not greater than the two valid security attributes.

Application Note: SFR FDP_IFF.2.7 is used internally when a MAC decision involving the relationships for any two valid pieces of information (e.g., import or insert row into table) for enforcing the Mandatory Access Control policy.

5.1.2.7 FDP_ITC.1 Import of user data without security attributes

FDP_ITC.1.1(1) The TSF shall enforce the **Discretionary Access Control policy** when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.1.2(1) The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.

FDP_ITC.1.3(1) The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC:

a) The subject must be a member of RUBIX Database Administrator, or

b) The subject must have EXECUTE privilege to all containing objects that hold the DBMS object,

c) The subject must have the INSERT privilege for all columns of the DBMS object, and

d) If the object is a View, the subject must have REFERENCE VIEW privilege on all underlying table columns from which the View object is created.

FDP_ITC.1.1(2) The TSF shall enforce the **Mandatory Access Control policy** when importing **unlabeled** user data, controlled under the SFP, from outside of the TSC. Refinement

FDP_ITC.1.2(2) The TSF shall ignore any security attributes associated with the **unlabeled** user data when imported from outside the TSC. Refinement

FDP_ITC.1.3(2) The TSF shall enforce the following rules when importing **unlabeled** user data controlled under the SFP from outside the TSC: Refinement

a) To import user data into a database table, the subject’s sensitivity label must dominate the sensitivity label of the table where the user

data are to be inserted

b) **The data is to be labeled with the session sensitivity label of the subject**.

5.1.2.8 FDP_ITC.2 Import of user data with security attributes

FDP_ITC.2.1(1) The TSF shall enforce the **Mandatory Access Control policy** when importing **labeled** user data, controlled under the SFP, from outside of the TSC. Refinement

FDP_ITC.2.2(1) The TSF shall use the security attributes**: the sensitivity labels** associated with the imported **labeled** user data. Refinement

FDP_ITC.2.3(1) The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes**: the sensitivity labels** and the **labeled** user data received. Refinement

FDP_ITC.2.4(1) The TSF shall ensure that interpretation of the security attributes**: the sensitivity labels** of the imported **labeled** user data is as intended by the source of the user data. Refinement

FDP_ITC.2.5(1) The TSF shall enforce the following **Mandatory Access Control** rules when importing **labeled** user data controlled under the SFP from outside the TSC: Refinement

a) **The ability to import user data with multilevel sensitivity labels into a database table or view is restricted to the members of RUBIX Security Administrator.**

b) **To import user data with multilevel sensitivity labels into a database table or view, the subject's sensitivity label must dominate the sensitivity label of the table.**

FDP_ITC.2.1(2) The TSF shall enforce the **Discretionary Access Control policy** when importing **labeled** user data, controlled under the SFP, from outside of the TSC. Refinement

FDP_ITC.2.2(2) The TSF shall use the security attributes**: the sensitivity labels** associated with the imported **labeled** user data. Refinement

FDP_ITC.2.3(2) The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes**: the sensitivity labels** and the **labeled** user data received. Refinement

FDP_ITC.2.4(2) The TSF shall ensure that interpretation of the security attributes**: the sensitivity labels** of the imported **labeled** user data is as intended by the source of the user data. Refinement

FDP_ITC.2.5(2) The TSF shall enforce the following **Discretionary Access Control** rules when importing **labeled** user data controlled under the SFP from outside the TSC: Refinement

a) **The subject must be a member of RUBIX Database Administrator, or**

b) **The subject must have EXECUTE privilege to all containing objects that hold the DBMS object,**

c) **The subject must have the INSERT privilege for all columns of the DBMS object, and**

d) **If the object is a View, the subject must have REFERENCE VIEW privilege on all underlying table columns from which the View object is created.**

5.1.2.9 FDP_RIP_DB.2 TOE full residual information protection

FDP_RIP_DB.2.1 The TSF, when invoked by the underlying host OS, shall ensure that any previous information content of a resource in the TSC is made unavailable upon the allocation of the resource to all objects in the TSC.

5.1.2.10 FDP_ROL.2 Advanced rollback

FDP_ROL.2.1(1) The TSF shall enforce **Discretionary Access Control policy and Mandatory Access Control policy** to permit the rollback of all the operations on **all** the **DBMS objects**. Refinement

FDP_ROL.2.2(1) The TSF shall permit operations to be rolled back within the **current transaction at the time of a database transaction failure**.

FDP_ROL.2.1(2) The TSF shall enforce **Discretionary Access Control policy and Mandatory Access Control policy** to permit the rollback of all the operations on **all** the **DBMS objects**. Refinement

FDP_ROL.2.2(2) The TSF shall permit operations to be rolled back within the **active transactions at the time of a system or media failure**.

5.1.3 Identification and authentication (FIA)

5.1.3.1 FIA_ATD.1 User attribute definition

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users:

a) **DBMS user identifier**

b) **DBMS group identifier**

c) **Security-relevant DBMS authorizations**

d) **User session sensitivity label**.

5.1.3.2 FIA_USB.1 User-subject binding

FIA_USB.1.1-NIAP-0415 The TSF shall associate the following user security attributes with subjects acting on behalf of that user:

a) **User ID**

b) **Group ID**

c) **Security-relevant DBMS authorizations**

d) **Session sensitivity label**

5.1.4 Security Management (FMT)

5.1.4.1 FMT_MOF.1 Management of security functions behavior

FMT_MOF.1.1(1) The TSF shall restrict the ability to <u>disable and enable</u> the functions: **auditing of individual auditable events by the audit generation function** to **DBMS Audit Administrators**.

FMT_MOF.1.1(2) The TSF shall restrict the ability to <u>enable</u> the functions: **importing multilevel user data into the database by the import function** to **DBMS Security Administrators**.

FMT_MOF.1.1(3) The TSF shall restrict the ability to <u>enable</u> the functions: **drop, list. or move databases** to **DBMS Database Administrators**.

5.1.4.2 FMT_MSA.1 Management of security attributes

FMT_MSA.1.1(1) The TSF shall enforce the **Discretionary Access Control Policy and the Mandatory Access Control policy** to restrict the ability to <u>modify</u> the security attributes **in the Access Control List (ACL) associated all DAC DBMS objects** to **the following subjects:**

a) **The Database Administrators whose current session sensitivity label is equal to the sensitivity label of the object, and**

b) **The authorized DBMS users whose current session sensitivity label is equal to the sensitivity label of the object, and who have EXECUTE privileges on all containing DBMS objects, and have the privilege with the GRANT OPTION for the privilege being modified.**

Application Note: Table row is a MAC protected object, but is not a DAC protected object.

FMT_MSA.1.1(2) The TSF shall enforce the **Discretionary Access Control Policy and the Mandatory Access Control policy** to restrict the ability to <u>modify</u> the **<u>following</u>** security attributes to **the DBMS Security Administrators:** ***Refinement***

a) **The sensitivity label of table row in DBMS, and**

b) **The session sensitivity label.**

FMT_MSA.1.1(3) The TSF shall enforce the **Discretionary Access Control Policy and the Mandatory Access Control policy** to restrict the ability to query the security attributes **in the Access Control List (ACL) associated with all DAC DBMS objects** to **the following subjects:**

a) **The Database Administrators operating at a session sensitivity label that dominates the sensitivity label of the object, and**

b) **The DBMS users, whose current session sensitivity label dominates the sensitivity label of the object, have EXECUTE privilege on all containing DBMS objects, have READ privilege to the object, have EXECUTE privilege to the information schema, have SELECT privilege to the referenced columns in the information schema view, and have REFERENCE VIEW privilege on all underlying definition schema table columns from which the information schema view object is created.**

5.1.4.3 FMT_MSA.2 Secure security attributes

FMT_MSA.2.1 The TSF shall ensure that only secure values are accepted for security attributes.

5.1.4.4 FMT_MSA.3 Static attribute initialization

FMT_MSA.3.1 The TSF shall enforce the **Discretionary Access Control policy and Mandatory Access Control policy** to provide restrictive default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow **none of the identified security roles** to specify alternative initial values to override the default values when an object or information is created.

5.1.4.5 FMT_MTD.1 Management of the TSF data

FMT_MTD.1.1(1) The TSF shall restrict the ability to query, modify, delete**, and add** the **auditable event selections for audit generation function** to **the DBMS Audit Administrators**.

FMT_MTD.1.1(2) The TSF shall restrict the ability to query**, extract, and examine** the **audit trails** to **the DBMS Audit Administrators**.

5.1.4.6 FMT_REV.1 Revocation

FMT_REV.1.1 The TSF shall restrict the ability to revoke security attributes **in the ACL entries** associated with the **DBMS** objects within the TSC to **the Database**

Administrators and users who have *EXECUTE* privileges on all containing DBMS objects and the privilege with the GRANT OPTION on the object for the privilege being revoked. Refinement

FMT_REV.1.2 The TSF shall enforce the rules **for all database transactions that start after the ACL updating transaction successfully commits**.

Application Note: ACL entry is the only place holding the revocable security attributes associated with TOE objects.

5.1.4.7 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following security management functions:

a) Security audit function management

b) Security audit data management

c) DBMS object access privileges management

d) Database backup and restore

e) Data reclassification

f) Single-level database import and export

g) Multilevel database Import

h) Drop, list, or move database

5.1.4.8 FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the **authorized DBMS user** roles:

a) DBMS audit administrator

b) Database administrator

c) Database operator

d) DBMS security administrator

e) Non-administrative DBMS user

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

5.1.5 Protection of the TOE Security Functions (FPT)

5.1.5.1 FPT_RCV.4 Function recovery

FPT_RCV.4.1 The TSF shall ensure that **the following SFs** have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state: Refinement

a) **Database transaction**
b) **Audit record logging**
c) **Modification of DAC Privileges in ACL**

5.1.5.2 FPT_RVM_DB.1 TOE non-bypassability of the TSP

FPT_RVM_DB.1.1 The TSF, when invoked by the underlying host OS, shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

5.1.5.3 FPT_SEP_DB.1 Partial TSF domain separation

FPT_SEP_DB.1.1 The TSF, when invoked by the underlying host OS, shall maintain a security domain that protects it from interference and tampering by untrusted subjects in the TSC.

FPT_SEP_DB.1.2 The TSF, when invoked by the underlying host OS, shall enforce separation between the security domains of subjects in the TSC.

5.1.6 Strength of Function

Although this TOE does not have any security functions that are realized by probabilistic of permutational mechanisms, the overall strength of function claim is SOF-medium.

5.2 Security Requirements for the IT Environment

5.2.1 User Data Protection (FDP)

5.2.1.1 FDP_ACC.1 Access Control Policy

FDP_ACC.1.1 The TSF shall enforce the **Discretionary Access Control (DAC) Policy** on system subjects: processes acting on the behalf of users and File System objects: **files and directories for operations: create, unlink, execute, open, and rename**.

5.2.1.2 FDP_ACF.1 Access Control Functions

FDP_ACF.1.1-**NIAP-0416** The **security functions of the IT environment** shall enforce the **Discretionary Access Control Policy** to **File System** objects based on the following:

a) **The user identity and group membership(s) associated with a subject; and**

b) The access control attributes associated with an object: ACL, permission bits

FDP_ACF.1.2 The **<u>security functions of the IT environment</u>** shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

IF the object has an explicit ACL, THEN:

- access granted to the object's owner is based on the user::rwx permissions

- access granted to individuals specified in the ACL is based on the bitwise AND operation of the user:[specified]:rwx and mask:rwx permissions

- access granted to subjects who belong to the object's group is based on the bitwise AND operation of the group::rwx and the mask:rwx entries

- access granted to subjects who belong to groups specified in the ACL is based on the bitwise AND operation of the group:[specified]:rwx and mask:rwx permissions

- access granted to all other subjects is based on the object's other permissions

ELSE

- access granted to the object's owner is based on the object user rwx permissions

- access granted to subjects who belong to the object's group is based on the object group rwx permissions

- access granted to all other subjects is based on the object other rwx permissions

FDP_ACF.1.3 The **<u>security functions of the IT environment</u>** shall explicitly authorise access of subjects to objects based on the following additional rules:

a) If a subject has an appropriate override privilege the TSF shall authorize access of the subject to any filesystem object

FDP_ACF.1.4 The **<u>security functions of the IT environment</u>** shall explicitly deny access of subjects to objects based on **no additional rules**.

5.2.1.3 FDP_IFC.1 Subset information flow control

FDP_IFC.1.1 The TSF shall enforce **the Mandatory Control Policy** on **subjects, objects and all operations amongst subjects and objects covered by the MAC policy** .

5.2.1.4 FDP_IFF.1 Simple security attributes

FDP_IFF.1.1 The TSF shall enforce the **Mandatory Control Policy** based on the following types of subject and information security attributes:

• the sensitivity label of the subject; and

• the sensitivity label of the object containing the information.

Sensitivity label of subjects and objects shall consist of the following:

• a hierachical level; and

• a set of non-hierachical categories.

FDP_IFF.1.2 The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules based on the ordering relationships between security attributes hold:

• if the sensitivity label of the subject is greater than or equal to the sensitivity label of the object, then the flow of information from the object to the subject is permitted (a read operation);

• if the sensitivity label of the object is greater than or equal to the sensitivity label of the subject, and the sensitivity label of the object is less than or equal to the clearance of the subject, then the flow of information from the subject to the object is permitted (a write operation);

• if the sensitivity label of subject A is greater than or equal to the sensitivity label of subject B, then the flow of information from subject B to subject A is permitted.

FDP_IFF.1.3 The TSF shall enforce **no additional information flow control SFP rules**.

FDP_IFF.1.4 The TSF shall provide **no additional information flow control SFP capabilities**.

FDP_IFF.1.5 The TSF shall explicitly authorise an information flow based on the following rules:

a) If a subject has an appropriate override privilege the TSF shall authorize the information flow

FDP_IFF.1.6 The TSF shall explicitly deny an information flow based on **no additional rules**.

5.2.1.5 FDP_ITT.1 Internal TOE Transfer

FDP_ITT.1.1 The **<u>security functions of the IT environment</u>** shall enforce the **Discretionary Access Control policy and the Mandatory Access Control policy** to prevent the <u>disclosure or modification</u> of **<u>authorized DBMS</u>** user data when it is transmitted between physically-separated parts of the **<u>IT environment</u>**. Refinement

5.2.1.6 FDP_RIP_OS.2 Environment full residual information protection

FDP_RIP_OS.2.1 The **<u>security functions of the host OS</u>** shall ensure that any previous information content of a resource in the scope of control of the host OS is made unavailable upon the <u>allocation of the resource to</u> all objects in the scope of control of the host OS.

5.2.2 Identification and authentication (FIA)

5.2.2.1 FIA_ATD.1 User attribute definition

FIA_ATD.1.1 The **<u>security functions of the host OS</u>** shall maintain the following list of security attributes belonging to individual users: Refinement

a) **User identifier,**
b) **Group memberships,**
c) **Authentication data,**
d) **User clearances, and**
e) **Security-relevant authorizations.**

5.2.2.2 FIA_UAU.2 User authentication before any action

FIA_UAU.2.1 The **<u>security functions of the host OS</u>** shall require each user to be successfully authenticated before allowing any TSF-mediated actions on behalf of that user. Refinement

5.2.2.3 FIA_UID.2 User identification before any action

FIA_UID.2.1 The **<u>security functions of the host OS</u>** shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user. Refinement

5.2.2.4 FIA_USB.1 User-subject binding

FIA_USB.1.1-NIAP-0415 The **<u>security functions of the host OS</u>** shall associate the following user security attributes with subjects acting on behalf of that user: Refinement

a) **User ID**
b) **Group ID**
c) **Security-relevant authorizations**
d) **Session sensitivity label**

5.2.3 Security management (FMT)

5.2.3.1 FMT_MSA.1 Management of security attributes

FMT_MSA.1.1 The **security functions of the host OS** shall enforce **the Discretionary Access Control Policy and the Mandatory Access Control Policy** to restrict the ability to modify the security attributes **of the DBMS user** to **the OS security administrators.** Refinement

5.2.3.2 FMT_MTD.1 Management of TSF data

FMT_MTD.1.1(1) The **security functions of the host OS** shall restrict the ability to **initialize** the **user authentication data: passwords** to **the OS security administrators**. Refinement

FMT_MTD.1.1(2) The **security functions of the host OS** shall restrict the ability to **change** the **user authentication data: his own password** to **the authorized user or the OS security administrators**. Refinement

FMT_MTD.1.1(3) The **security functions of the host OS** shall restrict the ability to modify the **user name, user ID, user groups, and user clearance** to **the OS security administrators.** Refinement

5.2.4 Protection of the TOE Security Functions (FPT)

5.2.4.1 FPT_AMT.1 Abstract machine testing

FPT_AMT.1.1 The **security functions of the host OS** shall run a suite of tests during initial start-up and at the request of an authorized user to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF. Refinement

5.2.4.2 FPT_ITT.1 Basic internal TSF data transfer protection

FPT_ITT.1.1 The **security functions of the host OS** shall protect TSF data from disclosure or modification when it is transmitted between separate parts of the **IT environment**. Refinement

5.2.4.3 FPT_RCV. 2-NIAP-0406 Recovery from Failure

FPT_RCV.2.1-NIAP-0406 For the **following list of failures,** the TSF shall ensure the return of the TOE to a secure state using automated procedures

a) A specified privilege is assigned to any role but the database containing the privilege data is not on-line or the particular data table is inaccessible;

b) A specified role has been assigned to a particular user but the database containing the role membership information is not on-line or the particular data table is inaccessible.

FPT_RCV.2.2-NIAP-0406 When automated recovery from a failure or service discontinuity is not possible, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.

5.2.4.4 FPT_RVM_OS.1 Environment non-bypassability of the TSP

FPT_RVM_OS.1.1 The **security functions of the host OS** shall ensure that **host OS security policy** enforcement functions are invoked and succeed before each function within the **scope of control of the host OS** is allowed to proceed.

5.2.4.5 FPT_SEP_OS.1 Environment TSF domain separation

FPT_SEP_OS.1.1 The **security functions of the host OS** shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects in the scope of control of the host OS.

FPT_SEP_OS.1.2 The **security functions of the host OS** shall enforce separation between the security domains of subjects in the **scope of control of the host OS**.

5.2.4.6 FPT_STM.1 Reliable time stamps

FPT_STM.1 The **security functions of the host OS** shall be able to provide reliable time stamps for its own use and **for the TOE**. $^{\text{Refinement}}$

5.3 TOE Security Assurance Requirements

This section defines the assurance requirements for Trusted RUBIX TOE. The evaluation assurance level (EAL) of this ST is EAL4 from the CC, Part 3. The Security Assurance Requirements for EAL4 are shown in Table 4.

Table 4 EAL4 Security Assurance Requirements

Class	Components	
ACM: Configuration management	ACM_AUT.1	Partial CM automation
	ACM_CAP.4	Generation support and acceptance procedures
	ACM_SCP.2	Problem tracking CM coverage
ADO: Delivery and operation	ADO_DEL.2	Detection of modification
	ADO_IGS.1	Installation, generation, and start-up procedures
ADV: Development	ADV_FSP.2	Fully defined external interfaces
	ADV_HLD.2	Security enforcing high-level design
	ADV_IMP.1	Subset of the implementation of the TSF
	ADV_LLD.1	Descriptive low-level design
	ADV_RCR.1	Informal correspondence demonstration
	ADV_SPM.1	Informal TOE security policy model
AGD: Guidance documents	AGD_ADM.1	Administrator guidance
	AGD_USR.1	User guidance
ALC: Life cycle support	ALC_DVS.1	Identification of security measures
	ALC_LCD.1	Developer defined life-cycle model
	ALC_TAT.1	Well-defined development tools
ATE: Tests	ATE_COV.2	Analysis of coverage
	ATE_DPT.1	Testing: high-level design
	ATE_FUN.1	Functional testing
	ATE_IND.2	Independent testing - sample
AVA: Vulnerability assessment	AVA_MSU.2	Validation of analysis
	AVA_SOF.1	Strength of TOE security function evaluation
	AVA_VLA.2	Independent vulnerability analysis

5.3.1 Configuration Management (CM)

5.3.1.1 ACM_AUT.1 Authorization controls

Developer action elements:

ACM_AUT.1.1D	The developer shall use a CM system.
ACM_AUT.1.2D	The developer shall provide a CM plan.

Content and presentation of evidence elements:

ACM_AUT.1.1C	The CM system shall provide an automated means by which only authorized changes are made to the TOE implementation representation.
ACM_AUT.1.2C	The CM system shall provide an automated means to support the generation of the TOE.
ACM_AUT.1.3C	The CM plan shall describe the automated tools used in the CM system.
ACM_AUT.1.4C	The CM plan shall describe how the automated tools are used in the CM system.

5.3.1.2 ACM_CAP CM capabilities

Developer action elements:

ACM_CAP.4.1D	The developer shall provide a reference for the TOE.
ACM_CAP.4.2D	The developer shall use a CM system.
ACM_CAP.4.3D	The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.4.1C	The reference for the TOE shall be unique to each version of the TOE.
ACM_CAP.4.2C	The TOE shall be labeled with its reference.
ACM_CAP.4.3C	The CM documentation shall include a configuration list, a CM plan, **and an acceptance plan.** The configuration list shall uniquely identify all configuration items that comprise the TOE.
ACM_CAP.4.4C	The configuration list shall describe the configuration items that comprise the TOE.
ACM_CAP.4.5C	The CM documentation shall describe the method used to uniquely identify the configuration items.
ACM_CAP.4.6C	The CM system shall uniquely identify all configuration items.
ACM_CAP.4.7C	The CM plan shall describe how the CM system is used.
ACM_CAP.4.8C	The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.4.9C	The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.
ACM_CAP.4.10C	The CM system shall provide measures such that only authorised changes are made to the configuration items.
ACM_CAP.4.11C	The CM system shall support the generation of the TOE.
ACM_CAP.4.12C	The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

5.3.1.3 ACM_SCP Problem tracking CM coverage

Developer action elements:

ACM_SCP.2.1D	The developer shall provide a list of configuration items for the TOE.

Content and presentation of evidence elements:

ACM_SCP.2.1C	The list of configuration items shall include the following: implementation representation; security flaws; and the evaluation evidence required by the assurance components in the ST.

5.3.2 Delivery and Operation (DO)

5.3.2.1 ADO_DEL.2 Detection of modification

Developer action elements:

ADO_DEL.2.1D	The developer shall document procedures for delivery of the TOE or parts of it to the user.
ADO_DEL.2.2D	The developer shall use the delivery procedures.

Content and presentation of evidence elements:

ADO_DEL.2.1C	The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.
ADO_DEL.2.2C	The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.
ADO_DEL.2.3C	The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

5.3.2.2 ADO_IGS.1 Installation, generation, and start-up procedures

Developer action elements:

ADO_IGS.1.1D	The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.1.1C	The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

5.3.3 Development (DV)

5.3.3.1 ADV_FSP.2 Fully defined external interfaces

Developer action elements:

ADV_FSP.2.1D	The developer shall provide a functional specification.

Content and presentation of evidence elements:

ADV_FSP.2.1C	The functional specification shall describe the TSF and its external interfaces using an informal style.
ADV_FSP.2.2C	The functional specification shall be internally consistent.
ADV_FSP.2.3C	The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing **complete** details of **all** effects, exceptions and error messages.
ADV_FSP.2.4C	The functional specification shall completely represent the TSF.
ADV_FSP.2.5C	The functional specification shall include rationale that the TSF is completely represented.

5.3.3.2 ADV_HLD.2 Security enforcing high-level design

Developer action elements:

ADV_HLD.2.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.2.1C	The presentation of the high-level design shall be informal.
ADV_HLD.2.2C	The high-level design shall be internally consistent.

ADV_HLD.2.3C	The high-level design shall describe the structure of the TSF in terms of subsystems.
ADV_HLD.2.4C	The high-level design shall describe the security functionality provided by each subsystem of the TSF.
ADV_HLD.2.5C	The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.
ADV_HLD.2.6C	The high-level design shall identify all interfaces to the subsystems of the TSF.
ADV_HLD.2.7C	The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.
ADV_HLD.2.8C	The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects exceptions and error messages, as appropriate.
ADV_HLD.2.9C	The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

5.3.3.3 ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements:

ADV_IMP.1.1D	The developer shall provide the implementation representation for a selected subset of the TSF.

Content and presentation of evidence elements:

ADV_IMP.1.1C	The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.
ADV_IMP.1.2C	The implementation representation shall be internally consistent.

5.3.3.4 ADV_LLD.1 Descriptive low-level design

Developer action elements:

ADV_LLD.1.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements:

ADV_LLD.1.1C	The presentation of the low-level design shall be informal.
ADV_LLD.1.2C	The low-level design shall be internally consistent.
ADV_LLD.1.3C	The low-level design shall describe the TSF in terms of modules.
ADV_LLD.1.4C	The low-level design shall describe the purpose of each module.

ADV_LLD.1.5C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.1.6C The low-level design shall describe how each TSP-enforcing function is provided.

ADV_LLD.1.7C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.1.8C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.1.9C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_LLD.1.10C The low-level design shall describe the separation of the TOE into TSP enforcing and other modules.

5.3.3.5 ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_RCR.1.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

Content and presentation of evidence elements:

ADV_RCR.1.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

5.3.3.6 ADV_SPM.1 Informal TOE security policy model

Developer action elements:

ADV_SPM.1.1D The developer shall provide a TSP model.

ADV_SPM.1.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements:

ADV_SPM.1.1C The TSP model shall be informal.

ADV_SPM.1.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.1.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.1.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

5.3.4 Guidance Documents (GD)

5.3.4.1 AGD_ADM.1Administrator guidance

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

AGD_ADM.1.2C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.3C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.4C The administrator guidance shall describe all assumptions regarding user behaviour that are relevant to secure operation of the TOE.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall be consistent with all other documentation supplied for evaluation.

AGD_ADM.1.8C The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.

5.3.4.2 AGD_USR.1 User guidance

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

AGD_USR.1.2C	The user guidance shall describe the use of user-accessible security functions provided by the TOE.
AGD_USR.1.3C	The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.
AGD_USR.1.4C	The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behaviour found in the statement of TOE security environment.
AGD_USR.1.5C	The user guidance shall be consistent with all other documentation supplied for evaluation.
AGD_USR.1.6C	The user guidance shall describe all security requirements for the IT environment that are relevant to the user.

5.3.5 Life Cycle Support (LC)

5.3.5.1 ALC_DVS.1 Identification of security measures

Developer action elements:

ALC_DVS.1.1D	The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.1.1C	The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.
ALC_DVS.1.2C	The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

5.3.5.2 ALC_LCD.1 Developer defined life-cycle model

Developer action elements:

ALC_LCD.1.1D	The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.
ALC_LCD.1.2D	The developer shall provide life-cycle definition documentation.

Content and presentation of evidence elements:

ALC_LCD.1.1C	The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.
ALC_LCD.1.2C	The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

5.3.5.3 ALC_TAT.1 Well-defined development tools

Developer action elements:

ALC_TAT.1.1D	The developer shall identify the development tools being used for the TOE.
ALC_TAT.1.2D	The developer shall document the selected implementation-dependent options of the development tools.

Content and presentation of evidence elements:

ALC_TAT.1.1C	All development tools used for implementation shall be well-defined.
ALC_TAT.1.2C	The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.
ALC_TAT.1.3C	The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

5.3.6 Security Testing (TE)

5.3.6.1 ATE_COV.2 Analysis of coverage

Developer action elements:

ATE_COV.2.1D	The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements:

ATE_COV.2.1C	The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.
ATE_COV.2.2C	The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

5.3.6.2 ATE_DPT.1 Testing: high-level design

Developer action elements:

ATE_DPT.1.1D	The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.1.1C	The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.

5.3.6.3 ATE_FUN.1 Functional testing

Developer action elements:

ATE_FUN.1.1D	The developer shall test the TSF and document the results.
ATE_FUN.1.2D	The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.1.1C	The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.
ATE_FUN.1.2C	The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.
ATE_FUN.1.3C	The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.
ATE_FUN.1.4C	The expected test results shall show the anticipated outputs from a successful execution of the tests.
ATE_FUN.1.5C	The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

5.3.6.4 ATE_IND.2 Independent testing – sample

Developer action elements:

ATE_IND.2.1D	The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.2.1C	The TOE shall be suitable for testing.
ATE_IND.2.2C	The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

5.3.7 Vulnerability Assessment (VA)

5.3.7.1 AVA_MSU.2 Validation of analysis

Developer action elements:

AVA_MSU.2.1D	The developer shall provide guidance documentation.

AVA_MSU.2.2D The developer shall document an analysis of the guidance documentation.

Content and presentation of evidence elements:

AVA_MSU.2.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.2.2C The guidance documentation shall be complete, clear, consistent and reasonable.

AVA_MSU.2.3C The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.2.4C The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

AVA_MSU.2.5C The analysis documentation shall demonstrate that the guidance documentation is complete.

5.3.7.2 AVA_SOF.1 Strength of TOE security function evaluation

Developer action elements:

AVA_SOF.1.1D The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

Content and presentation of evidence elements:

AVA_SOF.1.1C For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.

AVA_SOF.1.2C For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.

5.3.7.3 AVA_VLA.2 Independent vulnerability analysis

Developer action elements:

AVA_VLA.2.1D The developer shall perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP.

AVA_VLA.2.2D The developer shall document the disposition of **identified** vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.2.1C The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.2.2C The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

6 TOE Summary Specification

This section describes how the TOE implements the security requirements defined in Section 5.1 TOE Security Functional Requirements and 5.2 TOE Security Assurance Requirements. Section 6.1 lists the TOE security functions, grouped by function set and then by function. Section 6.2 lists the assurance measures, grouped by assurance measure set and then by assurance measure.

TOE security functions and TOE assurance measures completely implement their corresponding TOE security requirements as listed in Section 5. The exceptions are those TOE security requirements that are implemented by a combination of TOE security functions (in other words, those requirements listed in section 8.3.1 that map to more than one security function). In those cases the security function description in this section identifies which portion of those requirements is implemented by that function.

6.1 TOE Security Functions

This section describes Trusted RUBIX in terms of security functions and explains how the requirements are satisfied by those functions. The security functions of Trusted RUBIX include:

- Security Audit,
- User Identification and Subject Binding
- Mandatory Access Control
- Discretionary Access Control
- Database Import and Export
- Security Management
- Residual Information Protection
- Trusted Recovery
- Protection of Security Functions

The following naming conventions are used for the purposes of traceability from the security functional requirements to the security functions.

Each security function has been given a full name and a two-letter symbol. For example, the Audit function has its full name “AUDIT and a symbol “AU”. Each security function contains one or more refined functions. Each of the refined functions also has a full name and a four-letter symbol. The typeset conventions used in this section are described in the first paragraph of Section 1.4, Conventions, of this document.

6.1.1 Security Audit (AUDIT)

Trusted RUBIX security audit function consists of four components:

- Audit event detection and recording functions
- A utility for managing which auditable events are to be recorded
- A utility for examining the recorded audit events
- Procedures for managing audit data.

The security audit function provides the means by which the audit administrators are able to manage the security audit in a manner that ensures that all RDBMS users are held accountable for their actions in accordance with the security policy of their organization. Auditing of database operations is performed on the Trusted RUBIX server machine. Trusted RUBIX provides the audit administrator with the ability to specify which events are to be selected for recording by the audit recording function through the definition of pre-selection criteria. When a security relevant event occurs that meets the pre-selection criteria, it is recorded into the Trusted RUBIX audit trail. If any failure occurs in writing audit data to the Trusted RUBIX auditing trail (e.g., audit trail is full, storage device failure), the following series of Trusted RUBIX events happen:

1. All active transactions are rolled back;
2. The server process closes all open objects and exits; and
3. The client is given an audit record storage error code.

Trusted RUBIX relies on the DBMS Audit Administrators (AUD), a default role with a set of `rubix.audit.*` authorizations, to properly manage the audit data once it has been recorded. The DBMS Audit Administrators are provided with audit management tools enabling them to obtain a subset of previously recorded audit data through the use of post-selection criteria, list the names and sizes of the audit files, and delete audit files.

6.1.1.1 Audit Record Generation (AU.RGEN)

When Trusted RUBIX detects that a security relevant action has taken place that warrants auditing, it determines if the event is auditable based on whether it meets the pre-selection criteria established by the DBMS audit administrator. If the event meets the criteria, Trusted RUBIX attempts to record the relevant information about the subject, object(s), and other event specific information into the Trusted RUBIX audit trail using the Trusted RUBIX audit recording functions.

All audit records consists of two parts of data. The first part is audit information common to all audit records. The second part is event specific information and varies according to the event being audited. The common information saved for all audit records is listed in Table 5 Common Audit Information in Audit Records. The event specific audit information saved in audit records is listed in Table 6 Event Specific Audit Information in Audit Records.

It should be noted that the order of information, which appears in the tables is the same order that data appears in the audit record.

Table 5 Common Audit Information in Audit Records

Common Audit Information	
Event name	**Name of the Trusted RUBIX audit event**
User ID	User ID of the subject who submitted the operation
Group ID	Group ID of the subject who submitted the operation
Database name	Name of the currently opened database
Session sensitivity label	The database session sensitivity label
Operation status	Outcome of the audited operation
Timestamp	Time when the operation occured
Transaction ID	Transaction ID of the current transaction.
Process ID	Process ID of the process that is auditing the event.
Session ID	Process session ID of the process group that is auditing the event.

It should be noted that in Table 6 information enclosed in brackets is optional, depending upon the specific operation being audited.

Table 6 Event Specific Audit Information in Audit Records

Database Events Audit Information	
Event	**Audit Information**
Create Database	database label, database name
Drop Database	database label, database name
Open Database	Process ID, session ID, target database label, target database name.

Database Events Audit Information

Event	Audit Information
Create Catalog	catalog label, catalog name
Drop Catalog	catalog label, catalog name
Create Schema	schema label, catalog name, schema name
Drop Schema	schema label, catalog name, schema name
Create View	view label, catalog name, schema name, view name, view structure, view definition
Drop View	view label, catalog name, schema name, view name
Create Relation	relation label, catalog name, schema name, relation name, relation structure
Drop Relation	relation label, catalog name, schema name, relation name
Alter Relation	relation label, catalog name, schema name, relation name, relation structure, operation type, [new relation structure]
Create Index	index label, catalog name, schema name, relation name, index name, index definition
Drop Index	index label, catalog name, schema name, relation name, index name
Create Transaction	None
Commit Transaction	None
Rollback Transaction	None
Savepoint Declare	internal savepoint name, user defined savepoint name
Savepoint Rollback	internal savepoint name, user defined savepoint name
Create Row	row label, catalog name, schema name, relation name, row data
Delete Row	row label, catalog name, schema name, relation name, row data
Update Row	row label, catalog name, schema name, relation name, old row data, new row data
ACL Modify	ACL label, operation type, old ACL bitmap, new ACL bitmap, grantor | revokee, object name
Fetch	composite row label, current catalog, current schema, cursor definition, composite row data

Database Events Audit Information	
Event	**Audit Information**
Cursor Open	cursor label, current catalog, current schema, cursor name, cursor definition
Cursor Close	cursor label, cursor name
Audit Modify	database label, command line arguments
Audit Review	database label, command line arguments
Import (per row)	relation label, catalog name, schema name, relation name, unix import file name, row data
Export (per row)	relation label, catalog name, schema name, relation name, unix export file name, row data
Dump	database label, command line arguments
Restore	database label, command line arguments
Alter Session label	new session label
Reclassify Row	old row label, new row label, catalog name, schema name, relation name, unix input file, row data
Manage Log Files	database label, command line arguments
Drop, List, or Move (Rename) a Database	database label, command line arguments

6.1.1.2 Audit Selective Review and Analysis (AU.ANLY)

Trusted RUBIX provides the default DBMS Audit Administrator role the capability of extracting the recorded data in a manner that provides sufficient data for analysis without inundating the analyst with unnecessary information. Trusted RUBIX has an audit data reduction utility, ***`rxauditrpt`*** command, that gives the audit administrator the ability to specify the criteria, which identify the set of records to be selected for extraction and display from the audit trail.

The criteria, known as post-selection criteria, to specify which events are to be extracted and displayed from the audit trail include the following:

- Event status (success/failure);
- Database name

6.1.1.3 Audit Selection from Auditable Events (AU.SELT)

The audit function provides the default DBMS Audit Administrators with the capability to specify criteria, which allow the Trusted RUBIX audit subsystem to select which audit events are to be recorded in the audit trail. The criteria, known as pre-selection criteria, include the following:

- Events to be recorded system-wide at all times;
- Events to be recorded based on the event type;
- Events to be recorded based on the identity of the user performing the action;
- Events to be recorded based on the sensitivity label of the subject performing the action;
- Events to be recorded based on the sensitivity label of the object(s) involved;
- Events to be recorded based on the database being operated upon; and
- Turn the auditing system on or off without affecting the other audit criteria.

The utility used to specify the pre-selection criteria for auditing is ***`rxauditset`*** command. This utility allows the auditor to specify which events are to be recorded on a system-wide basis, as well as those to be recorded for specific users, for DBMS objects residing at a specific level and for DBMS objects residing within a range of levels.

Trusted RUBIX provides a set of auditable events to record the security relevant actions to the DBMS objects. The following types of events are recorded by Trusted RUBIX:

- Introduction of subjects to Trusted RUBIX;
- Introduction of objects into a user's address space;
- Deletion of objects;
- Actions taken by computer operators and other administrators;
- Operations (updates, retrievals, and inserts) initiated by untrusted users;
- Discretionary and mandatory access control policy decisions made by Trusted RUBIX;
- Creation and modification of ACLs;
- Use of commands in an administrative role; and
- Other security relevant events.

Table 7 documents the list of auditable security-relevant events audited by Trusted RUBIX and provides a brief description of each event.

Table 7 Trusted RUBIX Auditable Events

Event	Symbolic Name	Description
Create Database	sql_db_create	An attempt was made to create a database (e.g., via the ***CREATE DATABASE*** command)
Drop Database	sql_db_drop	An attempt was made to destroy a database (e.g., via the ***DROP DATABASE*** command)
Open Database	sql_db_open	An attempt was made to open a database. The database is opened prior to any SQL command being executed on that database. The database is also opened when "***rximport***" and "***rxexport***" are issued.
Create Catalog	sql_catalog_create	An attempt was made to create a catalog (e.g., via the ***CREATE CATALOG*** command)
Drop Catalog	sql_catalog_drop	An attempt was made to destroy a catalog (e.g., via the ***DROP CATALOG*** command)
Create Schema	sql_schema_create	An attempt was made to create a schema (e.g., via the ***CREATE SCHEMA*** command)
Drop Schema	sql_schema_drop	An attempt was made to destroy a schema (e.g., via the ***DROP SCHEMA*** command)
Create Index	sql_idx_create	Create an auxiliary index for a relation (e.g., via the ***CREATE INDEX*** command)
Drop Index	sql_idx_drop	Destroy an auxiliary index for a relation (e.g., via the ***DROP INDEX*** command)
Create View	sql_view_create	An attempt was made to create a defined view (e.g., via the ***CREATE VIEW*** command)
Drop View	sql_view_drop	An attempt was made to remove a defined view (e.g., via the ***DROP VIEW*** command)
Create Relation	sql_rel_create	An attempt was made to create a relation within a database (e.g., via the ***CREATE TABLE*** command)
Drop Relation	sql_rel_drop	An attempt was made to remove a relation from within a database (e.g., via the ***DROP TABLE*** command)

Event	Symbolic Name	Description
Alter Relation	sql_rel_alter	An attempt was made to change the structure of a relation (e.g., via the ***ALTER TABLE*** command)
Savepoint Declare	sql_savepoint_declare	An attempt was made to declare a savepoint (e.g., via the ***DECLARE SAVEPOINT*** command)
Savepoint Rollback	sql_savepoint_rollback	An attempt was made to rollback to a savepoint (e.g., via the ***ROLLBACK TO SAVEPOINT*** command)
Create Transaction	sql_trans_create	An attempt was made to start a new transaction (e.g., via the ***SET TRANSACTION*** command)
Commit Transaction	sql_trans_commit	An attempt was made to terminate a transaction (e.g., via the ***COMMIT*** command)
Rollback Transaction	sql_trans_rollback	An attempt was made to terminate a transaction (e.g., via the ***ROLLBACK*** command)
Create Row	sql_row_create	An attempt was made to insert a new row into a relation (e.g., via the ***INSERT*** command)
Delete Row	sql_row_delete	An attempt has been made to delete a row from the current representation of a relation (e.g., via the ***DELETE*** command)
Update Row	sql_row_update	An attempt has been made to modify one or more elements of a row within a relation (e.g., via the ***UPDATE*** command)
Audit Modify	sql_audit_modify	An attempt has been made to modify the auditable events, turn auditing on, or turn auditing off using ***rxauditset***.
Audit Review	sql_audit_review	An attempt has been made to review the audit trail using ***rxauditrpt***.
ACL Modify	sql_acl_modify	An attempt has been made to modify a privilege to a subject (e.g., via the ***GRANT*** or ***REVOKE*** commands)
Fetch	sql_fetch	An attempt has been made to fetch a row from a table (e.g., via the ***SELECT*** command)

Event	Symbolic Name	Description
Cursor Open	sql_cursor_open	An attempt has been made to open a cursor (e.g., via the ***SELECT*** command)
Cursor Close	sql_cursor_close	An attempt has been made to close a cursor (e.g., via the ***SELECT*** command)
Reclassify Row	sql_recl_row	An attempt was made to reclassify a row (e.g., via the ***rxrecl*** command)
Import	sql_import	An attempt has been made to import data using ***rximport***.
Export	sql_export	An attempt has been made to export data using ***rxexport***.
Dump	sql_dump	An attempt has been made to backup a database using ***rxdump***.
Restore	sql_restore	An attempt has been made to restore a database using ***rxrestore***.
Alter Session label	sql_lbl_alter	An attempt was made to alter the session sensitivity label (e.g., via the ***ALTER SESSION SET LABEL*** command)
Manage log files	sql_logs	An attempt was made to manage the audit or restore log files using the ***rxlogs*** command.
Drop, List, or Move (Rename) a Database	sql_db	An attempt was mode to drop, list, or move (rename) a database using the ***rxdb*** command.

6.1.1.4 Audit Record Logging (AU.RLOG)

Trusted RUBIX Audit Storage function stores the audit records into audit files as the audit records are generated by Audit Record Generation function. Audit files are created automatically as needed by the Trusted RUBIX server as Trusted Solaris 8 operating system files. The files are owned by NOBODY/RUBIXTP. They are stored at system high with read/write permissions set for the RUBIXTP group only. Each database, including the master database, stores its own audit files in the databases/DBNAME/DBNAME_adtlogs directory. Audit data in the master database's audit files are auditable events that are not directly associated with a database. Audit files are named aud-YYYY-MM-DD-LSN (e.g., aud-2002-01-14-1), where YYYY, MM, and DD are the year, month, and creation date of the audit file respectively. LSN is a logical sequence number that starts at "1" each day. New audit files are created automatically as the present audit files reach a maximum size. The next audit file created on the same day would be named with the incremented SEQ number (e.g., aud-2002-01-14-2). Upon detection of audit storage failure, the Trusted RUBIX server process automatically terminates itself as the prevention of audit data loss.

The database specific audit files are destroyed when that database is destroyed. Audit files may be garbage collected (selectively deleted) by the audit administrator by using the Trusted RUBIX command, ***`rxlogs`***, which allows the audit administrators to:

- Delete audit files for a specific database that were created prior to a given date, and
- List the audit files and their sizes.
- Set audit log files.

Upon detection of audit storage failure, the Trusted RUBIX server process automatically terminates itself as the prevention of audit data loss.

6.1.2 User Identification and Subject Binding (Identification)

Trusted RUBIX identifies authorized users of a particular Trusted RUBIX database by relying on the previously performed Identification and Authentication (I&A) of the underlying trusted operating system. Depending on the configuration of the Trusted RUBIX Server the user may also be authenticated using the password maintained by the underlying operating system during the database server or administrative command startup.

Trusted RUBIX is an SQL-based client/server multilevel secure (MLS) Relational DBMS operating in a client-server architecture. There is one instantiation of the server for each active client. A given client and server pair can run on the same machine or on different machines connected via a network. All communication between client and server takes place on a single-level connection. The client process is an application program, which has been linked with the Trusted RUBIX client software to communicate with its associated Trusted RUBIX server process, which resides on the same machine as the database to be protected. Currently, two types of clients are supported in Trusted RUBIX:

1. Instantiations of an Interactive SQL (ISQL) interface that provides ad hoc access to databases; and

2. User-developed applications utilizing the SQL Call-Level Interface (CLI).

6.1.2.1 User Security Attribute Definitions (IA.UATD)

When a user successfully establishes a login session on the Trusted Solaris operating system, processes, operating on behalf of that user, are assigned user security attributes that reflect the identity of the user (i.e., user ID and group ID) and the subject sensitivity label as the sensitivity label of the user (login) session. When a user wishes to perform operations with Trusted RUBIX, the user must first initiate an Interactive SQL (ISQL) or Call Level Interface (CLI) client process. The client process is associated with the security attributes of the user ID, group ID, and session sensitivity label.

When the client process initiates a connection to a specific database, a trusted server process is created. This trusted process, called Trusted RUBIX server, stores the user's security attributes (i.e., user ID, group ID, session sensitivity label) and then changes the process's user ID and group ID attributes to `RUBIX` and `RUBIXTP`, respectively, and the process security level to system high (`SYSHIGH`). For each of the Trusted RUBIX defined authorizations, Trusted RUBIX process (e.g., Trusted RUBIX Server process, Trusted RUBIX command process) asks the OS whether the user has this authorization and maintains the user authorizations as part of this user security attributes in the process. The Trusted RUBIX server process uses the stored user security attributes to identify the user and to perform Mandatory Access Control and Discretionary Access Control mediation on the DBMS objects, and describe users in the audit trail. The stored user security attributes include:

- User ID,
- Group ID,
- Session sensitivity label, and
- Trusted RUBIX defined authorizations for security-relevant roles.

Security-relevant roles in Trusted RUBIX are established by setting up various Trusted Solaris authorizations and then giving users authorizations to assume their security-relevant roles. The entire Trusted Solaris Role Based Access control mechanism may be utilized to create trusted roles that satisfy the requirements of the end user. Default authorizations are logically grouped to establish a role as:

- DBMS Audit Administrators (AUD or Auditor) defined by the *`rubix.audit.*`* authorizations;
- Database Administrators (DBA) defined by the *`rubix.dac.*,`* *`rubix.restore.create.*`*, and *`rubix.admin.*`* authorizations;
- Database Operators (OP) defined by the *`rubix.restore.backup.*`*, *`rubix.restore.listlogs.*`*, *`rubix.restore.rmlogs.*`*, and *`rubix.restore.setlogs.*`* authorizations..
- DBMS Security Administrators (SA) defined by the *`rubix.mac.*`* and *`rubix.grant`* authorizations;
- Non-Administrative DBMS users optionally with the *`rubix.user.*`* authorizations;

6.1.2.2 User-Subject Binding (IA.USBD)

The client and server processes may reside on the same machine (local mode) or different machines (non-local mode). The user-subject binding for the local mode and non-local mode are described in detail below.

Local Client and Server Subject Attributes

In local mode, the client process is initiated by an operating system authenticated user and has the security attributes of the user. The server process is initiated through an *exec* system call when the client process initiates connection to a specific database.

When the server process starts up, it has the same security attributes as that of the client process. The server process then stores the user security attributes and changes the process security attributes to Trusted RUBIX reserved attributes (i.e., user ID of `RUBIX`, group ID of `RUBIXTP`, and level `SYSHIGH`). The trusted server process serves the client with database operations, using the following stored attributes to arbitrate access to DBMS objects according to the Mandatory Access Control and Discretionary Access Control policies:

- User identifier;
- Group identifier;
- User authorizations; and
- User session sensitivity label.

Non-Local Client and Server Subject Attributes

In non-local mode, the client process is initiated by an operating system authenticated user on the client machine and has the user's security attributes. The Trusted RUBIX server (e.g., *rxserver)* executables are restricted to the `RUBIXTP` group and run with the UNIX set group ID on execute bit set. No login user is allowed in the `RUBIXTP` group. Trusted RUBIX Server (*rxserver)* is set up as an inet daemon service in inetd configuration file. When the client process initiates an ISQL command or a CLI connect (connect system call) to the IP address of the server at the special port number of *rxserver* service, the Trusted Solaris inetd daemon on the server machine starts up an instance of *rxserver* with RUBIX for uid and RUBIXTP for gid and connects the socket from the client to the stdin of the rxserver process. Trusted RUBIX server process retrieves the user's security attributes (session sensitivity label, clearance, UID, GID) through system call of the Trusted Solaris socket. Those user security attributes are stored in server process memory. The server process then changes the process sensitivity label to system high, accepts the connection. The client sends the first message to Trusted RUBIX Server to initialize the database connection. Trusted RUBIX Server obtains the user's authorizations from the operating system and stores the authorizations in server process memory. Trusted RUBIX Server processes database request from the client process using the following stored user attributes to arbitrate access to DBMS objects based on the Mandatory Access Control and Discretionary Access Control policies:

- User identifier;
- Group identifier;
- User authorizations, and
- User session sensitivity label.

6.1.3 Mandatory Access Control (MAC)

Mandatory Access Control is a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity. The mandatory access control policy enforced by Trusted RUBIX is based on Bell and LaPadula's sensitivity policy. The sensitivity policy prevents a subject from reading information that is too sensitive for the subject's clearance. These policies are based on a dominance relationship between the subject and the object. The mandatory security policy is enforced in Trusted RUBIX by associating sensitivity labels with subjects and objects and mediating all accesses based on those sensitivity labels. Sensitivity labels are composed of a hierarchical security classification and some number of categories. A label is said to *dominate* another label if the classification of the first is equal to or greater than the classification of the second and all of the categories of the second are included in the categories of the first. A label is said to *equal* another label if the classification and categories are the same. If a label dominates but is not equal to a second label, the first is said to *strictly dominate* the second.

6.1.3.1 Relationship to the Operating System's MAC policy

Trusted RUBIX runs on the Trusted Solaris 8 kernel, which enforces the MAC policy by mediating all accesses based on the sensitivity labels of the subjects and objects controlled by the operating system. Trusted RUBIX MAC Security Functions are integrated with the underlying Trusted Solaris 8 kernel to form a single MAC Policy.

In Trusted RUBIX, all MAC protected data are stored in single-level operating system file objects. To support fine-grained multilevel objects, sensitivity labels are attached to individual database named objects within each operating system single-level file object. Note that these labels are DBMS labels and not operating system labels. The operating system views these sensitivity labels strictly as data and attaches no security significance to them. Trusted RUBIX is trusted to properly associate and maintain the sensitivity label of each named object and to correctly interpret those labels so that, in cooperation with the operating system kernel, the security policy can be correctly enforced.

6.1.3.2 Subject and Object MAC Label in Trusted RUBIX (DP.MACL)

Trusted RUBIX supports all security labels available in the operating system.

(A) Subject MAC Label

Subjects are assigned a sensitivity label reflecting the maximum sensitivity of the information they are permitted to access. In the Trusted RUBIX operating environment, the system label of a user is the label that is assigned to that user by the underlying operating system at login time. When a user connects to Trusted RUBIX, the label assigned is the session sensitivity label, which is used to classify all objects inserted into any database and is also used to enforce all MAC policies.

Trusted RUBIX provides facilities to dynamically alter the session label. Depending on the authorizations of the user, the user is allowed to set the new session label to a label that dominates the original session label, is dominated by the original session label, or to a label that is incomparable to the original session label. Furthermore, depending on the authorizations of the user, the user may submit read-only or read-write transactions. The user must be in the DBMS Security Administrator role to perform this operation.

(B) Object MAC Label

Objects are immediately and automatically protected by MAC at creation time in Trusted RUBIX. Objects are labeled with the session sensitivity label of the creating user and the label reflects the sensitivity of the data contained within. Containers that hold MAC protected objects (e.g., databases which hold tables) may contain objects equal to or greater than the container's label. Trusted RUBIX databases, catalogs, schemas, tables, views, indexes, and rows are MAC labeled and protected containers.

Database

When a database is created, it is immediately assigned a sensitivity label that matches the user's session sensitivity label. A new database is created along with a Trusted RUBIX defined number of volumes. It may be created with or without transaction logging enabled. The database can hold an object that is at this sensitivity label or higher.

Catalog

Multiple catalogs can be created at the label or higher of the container database. When a catalog is created, it is assigned a sensitivity label that matches the user's session sensitivity label.

Schema

Multiple schemas can be created at the label or higher of the container catalog. A schema also inherits the session sensitivity label when it is created.

Table

Multiple tables can be created at the label or higher of the container schema. The table is labeled the same as the user's session sensitivity label.

View

Multiple views can be created at the label or higher of the container schema. A new view is created with the specified columns. The tables referenced by the view must be dominated by the user's session sensitivity label. The view will be labeled with the user's session sensitivity label.

Index

Multiple indexes can be created at the label or higher of the container schema. The index is labeled the same as the user's session sensitivity label.

Row

In Trusted RUBIX, a label is attached to each row. When a user creates a table, the labels are automatically attached (i.e., stored internally in binary form in the column named “rowlabel”) to each row inserted in the table. The sensitivity label is identical to the user session sensitivity label during the row insertion. The item (row) within a table can be inserted at the sensitivity label of the table or higher.

Combine Information

When two pieces of information are combined, the sensitivity label of the resultant information is chosen to dominate the sensitivity label of the original pieces of information. Trusted RUBIX labels the new information by taking the higher of the two hierarchical classification levels and combining both sets of categories.

Extract Information

When a piece of information is extracted from another piece of information, it is labeled with the same sensitivity label of the original piece of information.

6.1.3.3 Mandatory Access Control Enforcement in Trusted RUBIX (DP.MACM)

Trusted RUBIX provides mandatory access control functions that are built atop the mandatory access control primitives of the trusted operating system. The security lattice defined for the operating system is used for Trusted RUBIX protected objects and subjects. Trusted RUBIX enforces the following mandatory access control rules:

- A subject is allowed to read a piece of information if the subject’s sensitivity label dominates the sensitivity label of the information.
- The subject is allowed to modify a piece of information if the subject session sensitivity label equals the sensitivity label of the information.

Trusted RUBIX allows the use of Standard SQL syntax to perform label comparisons. This feature is invoked by the subject to determine MAC dominance, i.e., which labels dominate, are dominated by, or are equal to other labels. By converting the sensitivity labels from standard character string (ASCII) to internal binary form and utilizing the underlying trusted operating system’s binary label comparisons, Trusted RUBIX provides the following label comparison functions:

Clause	Function
=	Is equal to
<	Is less than (strictly dominated by)
<=	Is less than or equal to (dominated by)

Clause	Function
>	Is greater than (strictly dominates)
<>	Is not equal to

The principle of MAC dominance and label comparison impacts the outcome of various SQL statements. Trusted RUBIX enforces the mandatory access control rules on each SQL statement and evaluates the various combinations of user session sensitivity labels and object sensitivity labels for mediating the resulting MLS outcome.

The following Table 8 describes the conditions associated with various SQL statements and the resulting outcomes of the SQL statements.

Table 8 Required Label Conditions associated with SQL Statements

SQL Statement	Sensitivity label Condition	MLS Outcome
`ALTER TABLE`	Session >= database Session >= parent catalog Session >= parent schema Session = table . Session = table referenced by a foreign key	Table sensitivity label is unchanged.
`CONNECT`	Session >= database	Connected at the session sensitivity label.
`CREATE CATALOG`	Session >= database	Created at the session sensitivity label and accessible to users who dominate.
`CREATE DATABASE`	Any Sensitivity label	Created at user session sensitivity label and accessible to all users who dominate.
`CREATE INDEX`	Session >= database Session >= parent catalog Session >= parent schema Session = table	Created at the sensitivity label of the table and is usable by all the users who dominate.
`CREATE SCHEMA`	Session >= database Session >= parent catalog	Created at the session sensitivity label, and accessible to users who dominate.

SQL Statement	Sensitivity label Condition	MLS Outcome
`CREATE TABLE`	Session >= database Session >= parent catalog Session >= schema Session = table referenced by a foreign key	Created with session sensitivity label. Accessed by users who dominate.
`CREATE VIEW`	Session >= database Session >= parent catalog Session >= parent schema Session >= table Session >= view	Created at user session sensitivity label. Accessible to users who dominate.
`DELETE`	Session >= database Session >= parent catalog Session >= parent schema Session >= table, [view] Session = row	Delete rows at session sensitivity label.
`DROP CATALOG`	Session >= database Session = catalog Session = any schema, table, or view inside of the catalog Session = any view that references a table and/or view inside of the catalog	The catalog and all containing objects (if cascade option specified) are dropped.
`DROP DATABASE`	Session = database	All objects in the database are destroyed.
`DROP INDEX`	Session >= database Session >= parent catalog Session >= parent schema Session = index	Drop the specified index.

SQL Statement	Sensitivity label Condition	MLS Outcome
`DROP SCHEMA`	Session >= database Session >= parent catalog Session = schema Session = any table or view inside of the schema Session = any view that references a table and/or view inside of the schema	The schema and all containing objects (if cascade option specified) are dropped.
`DROP TABLE`	Session >= database Session >= parent catalog Session >= parent schema Session = table Session = any view referencing the table	The table, all indices, and all rows are dropped. All referencing views are dropped (if cascade option specified).
`DROP VIEW`	Session >= database Session >= parent catalog Session >= parent schema Session = view Session = any view referencing the view	The view is dropped. All referencing views are dropped (if cascade option specified).
`GRANT`	Session >= containers Session = object	Privileges valid for all sensitivity labels that may acces object.
`INSERT`	Session >= database Session >= parent catalog Session >= parent schema Session >= table, [view]	The row inserted inherits the session sensitivity label.
`REVOKE`	Session >= containers Session = object	Privileges revoked are applicable for all sensitivity labels that may access object.
`SELECT`	Session >= database Session >= parent catalog Session >= parent schema Session >= table, [view]; Session >= row	Rows dominated by session sensitivity label are returned.
`SET CATALOG`	Session >= database Session >= catalog	Set at the session sensitivity label.

SQL Statement	Sensitivity label Condition	MLS Outcome
`SET SCHEMA`	Session >= database Session >= parent catalog Session >= schema	Set at the session sensitivity label.
`UPDATE`	Session >= database Session >= parent catalog Session >= parent schema Session >= table, [view]; Session = row	Rows at the session sensitivity label are updated.
`sub-query`*	Session >= database Session >= parent catalog Session >= parent schema Session >= table, [view]; Session >= row	Rows dominated by session sensitivity label are returned for the sub-query. * A sub-query is an implied operation on any DELETE, UPDATE, or SELECT operation that includes a WHERE clause and any INSERT operation that has a SELECT clause.

6.1.4 Discretionary Access Control (DAC)

Discretionary Access Control (DAC) is a means of restricting access to objects based upon the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access privilege is capable of passing that privilege to any other subject. Trusted RUBIX DAC is characterized in terms of subjects, named objects, and the operations that subjects can perform upon named objects. Subjects hold certain access privileges with respect to the named objects maintained by Trusted RUBIX. The named objects protected by Trusted RUBIX DAC are:

- Databases,
- Catalogs,
- Schemas,
- Indices,
- Tables,
- Views, and
- Columns.

6.1.4.1 Discretionary Access Control List (DP.DACL)

Each Trusted RUBIX named object is associated with an *access control list* (ACL). Each record in the ACL has the following three fields: User identifier, sum of all privileges, and a column-bitmap for tables. The User identifier can be user ID, group ID, or public. The sum of all privileges indicates all privileges the particular user has to that object. The column-bitmap field is used for tables. The various column grantable privileges are stored in this field.

Each Trusted RUBIX database has an ACL that specifies which users have read, write, execute/search, and/or administrative (i.e., *READ, WRITE, EXEC*, and *ADMIN*) privileges and their associated *GRANT* privilege on the object. In addition, the NULL privilege negates all other privileges, and the associated GRANT (i.e., GRANTNULL) privileges to grant the NULL privilege to someone. User has the *GRANT* privilege is allowed to grant or revoke the access (e.g., READ, WRITE) privilege associated with the *GRANT* privilege.

Each Trusted RUBIX catalog and schema has an ACL that specifies which users have read, write, execute/search (i.e., *READ, WRITE, and EXEC*) privileges and their associated *GRANT* privilege on the object. The NULL privilege negates all other privileges, and its GRANTNULL privileges to grant the NULL privilege to someone.

Each Trusted RUBIX table has an ACL that specifies the distribution of *DELETE, SELECT(I), INSERT(I), UPDATE(I), REFERENCES(I), CREATE VIEW(I), REFERENCE VIEW(I), NULL(I)* privileges and their associated *GRANT* privileges.

Each Trusted RUBIX view has an ACL that specifies the distribution of *DELETE, SELECT(I), INSERT(I), UPDATE(I), and CREATE VIEW(I), NULL(I)* privileges and their associated *GRANT* privileges.

The PRIVILEGE(I) form of these privileges permits the subject PRIVILEGE access to column "I" of a table or view. The *NULL* privilege can be used to explicitly deny a user access to an object. When calculating a user's effective privileges to an object, the *NULL* privilege can negate all other privileges.

(A) Default DAC Privileges

When named objects are created, they have default DAC privileges associated with them. Trusted RUBIX only grants privileges to the creator of the named object.

Named Objects	**Default DAC Privileges**	**Comments**
Databases	READ, WRITE, EXEC, ADMIN, GRANTNULL	Creator initially granted all valid privileges for a database.
Catalogs	READ, WRITE, EXEC, GRANTNULL	Creator initially granted all valid privileges for a catalog.
Schemas	READ, WRITE, EXEC, GRANTNULL	Creator initially granted all valid privileges for a schema.
Column of Table	UPDATE(I), SELECT(I), INSERT(I), DELETE, REFERENCES(I). GRANTNULL(I), CRVIEW(I), REFVIEW(I)	Creator initially granted all valid privileges for table columns.
Column of View	Computed per view. Creator granted creator's privileges on columns referenced in the view definition. The GRANT option is always given. All privileges valid for tables are valid for views, except REFERENCES(I) and REFVIEW(I).	Created initially with privileges of base object. The GRANT option is always given

The default PUBLIC DAC access privileges given to automatically created objects are:

Object	**Default Public DAC Privilege**
default_ catalog CATALOG	READ, EXEC

system_catalog CATALOG	READ, EXEC (fixed)
default_schema SCHEMA	READ, EXEC
info_schem SCHEMA	READ, EXEC
definition_schema SCHEMA	READ, EXEC (fixed)
optimizer_schema SCHEMA	READ, EXEC (fixed)
statistics_schema SCHEMA	READ, EXEC (fixed)
All TABLE columns inside *system_catalog*	CRVIEW(I), REFVIEW(I) (fixed)
All VIEW columns inside the *info_schem*	SELECT(I)

(B) Management of DAC Privileges in ACL

Discretionary access control security is managed by allowing authorized users to specify which users and groups are authorized to perform specific operations on particular objects. Privileges to access the various named objects in Trusted RUBIX must be explicitly granted to the various DBMS users and groups. The *GRANT* statement grants privileges on Trusted RUBIX objects, to one or more users, provided the current user has the appropriate grant privileges on that object. The ability to further grant any privilege is governed by the *WITH GRANT OPTION*. Only a user who had previously been granted a privilege that included the *WITH GRANT OPTION* may further propagate it. Also, the user may have many other privileges, but may only propagate those privileges for which they have the associated *WITH GRANT OPTION*.

Privileges to access the various named objects in Trusted RUBIX can be explicitly revoked from the various DBMS users and groups. The *REVOKE* statement revokes privileges on a database, catalog, schema, columns of table or view to one or more users, provided the current user has the appropriate revoke privilege on that object. The ability to revoke any privilege is governed by the *WITH GRANT OPTION*.

The management functions to grant or revoke users' privileges are under the control of the DAC policy. Trusted RUBIX supports trusted users operating with the DBMS Database Administrator role to manipulate ACLs in the database. Trusted RUBIX also allows users who are not operating with the role to grant and revoke other users privileges on database object, if he has the privilege with the required *WITH GRANT OPTION* on the targeted object and has *EXECUTE* privileges on all the containing objects. The following table shows the required privileges for users, who are not operating within the Database Administrator role, to grant and/or revoke privileges on particular objects.

SQL Statements	**Subject DAC Privileges required on Container**	**Subject DAC Privileges required on Target Object**
`GRANT` or `REVOKE` privileges on databases		`WITH GRANT OPTION` privilege on database
`GRANT` or `REVOKE` privileges on catalogs	`EXECUTE` on database	`WITH GRANT OPTION` privilege on catalog
`GRANT` or `REVOKE` privileges on schemas	`EXEUTEC` on database `EXECUTE` on catalog	`WITH GRANT OPTION` privilege on schema
`GRANT` or `REVOKE` privileges on column of relations	`EXECUTE` on database `EXECUTE` on catalog `EXECUTE` on schema	`WITH GRANT OPTION` privilege on column of relations
`GRANT` or `REVOKE` privileges on column of views	`EXECUTE` on database `EXECUTE` on catalog `EXECUTE` on schema	`WITH GRANT OPTION` privilege on column of views

To modify privileges on an object, the user must be operating at the same MAC label as the object. The updated ACLs are in effect after the ACL updating transaction has successfully committed.

6.1.4.2 Discretionary Access Control Enforcement (DP.DACM)

Users operating within the DBMS Database Administrator role can access all objects in a Trusted RUBIX database without ACL authorization. All accesses to Trusted RUBIX named objects are validated against entries in its ACL. Discretionary Access Control enforcements are applied to any operation between subject and object. The required subject privileges for operations on objects are listed in the Table 9 Required Trusted RUBIX Privileges for Operations.

A subject's access is calculated from the contents of the object's ACL according to the following algorithm:

1. If an entry appears in the ACL matching the subject's real user ID, the subject is given the access specified in that ACL entry—the grant of the `NULL` privilege to the subject results in the subject having no access; otherwise,
2. If an entry appears in the ACL matching the subject's real group ID, the subject is given the access specified in that ACL entry— the grant of the `NULL` privilege to the group results in the subject having no access; otherwise,
3. If a "public" entry appears in the ACL, the subject is given the access specified in that entry; otherwise,

4. The subject is given no access to the object.

It should be noted that table rows are not DAC protected objects in Trusted RUBIX. DBMS users access table rows by referencing the specified table and table columns in a query. The discretionary access control mediations are made based upon the subject's privileges on the table/view and the columns in the table/view defined in the ACL associated with that table/view (See subsection "DAC Enforcement on Tables/Views" for detailed description).

(A) DAC Enforcement on Databases

The operations which may be exercised upon a Trusted RUBIX database are create, destroy, read, write, and execute. Databases may be created via the *CREATE DATABASE* command by any user. A database may be destroyed via the *DROP DATABASE* command if the user holds ADMIN privileges on the database. To scan (list) a database for catalog names requires that the user performing the operation hold *READ* privilege on that database.

When a database is created, Trusted RUBIX gives the creator all privileges. No other users are granted access. The ability to further grant any privilege is governed by the *WITH GRANT OPTION*. Only a user who had previously been granted a privilege that included the *WITH GRANT OPTION* may further propagate it. Also, the user may have many other privileges, but may only propagate those privileges for which he has the grant option.

To connect to a database, the subject must have *EXECUTE* privilege to the database. *READ* and *WRITE* privileges on a Trusted RUBIX database correspond to read and write permissions on a UNIX directory, respectively. To destroy a database the user must have ADMIN privileges.

At database creation the *default_catalog* and the *system_catalog* are created. Two associated schemas are created in the *default_catalog*: the *default_schema* and *info_schem*. Four associated schemas are created in the *system_catalog*: the *default_schema*, the *definition_schema*, the *statistics_schema*, and the *optimizer_schema*. The public is granted *READ* and *EXECUTE* privileges on all of these catalogs and schemas. All views inside the *info_schema* have the *SELECT* privilege given to public. All tables inside the *system_catalog's* schemas have the *CREATE VIEW* and *REFERENCE VIEW* given to public. The *system_catalog* and all of its contents have ACL's that may not be altered by any user. To be able to use any of the schema, the user must hold *EXECUTE* privilege on the database. The *NULL* privilege denies a user any access to a database. Granting the *NULL* privilege requires the *GRANTNULL* privilege.

(B) DAC Enforcement on Catalogs

The operations which may be exercised upon a Trusted RUBIX catalog are create, destroy, read, write, and execute. A catalog may be created via the *CREATE CATALOG* command if the user holds *WRITE* and *EXECUTE* privileges on the database. A catalog may be destroyed via the *DROP CATALOG* command if the user holds *WRITE* and *EXECUTE* privileges on the database.

The *NULL* privilege denies a user any access to a catalog. Granting the *NULL* privilege requires the *GRANTNULL* privilege.

The distribution of catalog access privileges to various users and groups of users is specified in the ACL attached to the catalog. The ACL specifies the distribution of *NULL, READ, WRITE*, and *EXECUTE* privileges. To create or destroy a catalog the user must have *WRITE* and *EXECUTE* privileges on the database containing the catalog. To list schemas in a catalog the user must have *READ* and *EXECUTE* privileges on the catalog. To operate on a catalog the user must have *EXECUTE* privilege on the containing database.

At catalog creation, the *default_schema* is created.

(C) DAC Enforcement on Schemas

The operations which may be exercised upon a Trusted RUBIX schema are create, destroy, read, write, and execute. A schema may be created via the *CREATE SCHEMA* command if the user holds *WRITE* and *EXECUTE* privileges on the catalog where the schema is to be created. A schema may be destroyed via the *DROP SCHEMA* command if the user holds *WRITE* and *EXECUTE* privileges on the catalog. The *NULL* privilege denies a user any access to a schema. Granting the *NULL* privilege requires the *GRANTNULL* privilege.

The distribution of schema access privileges to various users and groups of users is specified in the ACL attached to the schema. The ACL specifies the distribution of *NULL, READ, WRITE*, and *EXECUTE* privileges. *WRITE* privilege on a Trusted RUBIX schema corresponds to WRITE permission on a UNIX directory.

Schema is the container object for tables, views, and indexes. To create or destroy (via add or delete object entry) a table, defined view, or index in a schema, the user needs *WRITE* and *EXECUTE* privileges on the schema. To access any object inside a schema, the user must hold *EXECUTE* privilege. To list tables, views, or indices in a schema the user must have *EXECUTE* and *READ* privileges on the schema.

(D) DAC Enforcement on Tables

The operations that may be exercised upon a Trusted RUBIX table are create, destroy, alter, create a referencing view, and referenced through a view. A table is created via the *CREATE TABLE* command, which requires that the user hold *WRITE* and *EXECUTE* privileges on the schema where the table is to be placed and *REFERENCES(I)* privileges on any column referenced by a foreign key. A table may be destroyed via the *DROP TABLE* command if the user holds *WRITE* and *EXECUTE* privileges on the schema.

The individual rows that reside within the table may be subjected to the select, update, delete, and insert operations. For these operations the ACL privileges are defined on column objects of the table, (e.g., whether the user has the privileges on all the columns he attempts to select). The columns of the table may be altered (i.e., alter table) and referenced (e.g., reference a column by foreign key, reference a column from a view). In addition, different views of a single table or many tables together can be created. All these privileges are grantable to other users. The *SELECT* SQL command reports upon the contents of a table. In order to extract rows from a table, the user must hold *SELECT* privilege. The three operations, which modify the contents of a Trusted RUBIX table, are delete, insert, and update.

The privileges applicable to table objects are *DELETE, SELECT(I), INSERT(I), UPDATE(I), REFERENCES(I), CREATE VIEW(I)*, and *REFERENCE VIEW(I)*.

There is no *DELETE* privilege on an individual column. The *DELETE* privilege is held on all columns or no columns. To delete a row from a table a user must hold the *DELETE* privilege on all the columns. For the *SELECT* operation to be permitted, the user should hold column level *SELECT* privilege on all the columns being retrieved. The *INSERT* operation is permitted if the user has column level *INSERT* privilege on all the columns being inserted. The *UPDATE* operation is permitted if the user has column level *UPDATE* privilege on all the columns being updated and the user also has the *SELECT* privilege for the *column-identifier* being used.

To alter a table (i.e., Add or drop column or add or drop constraint) the user must have *EXECUTE* and *WRITE* privileges on the containing schema and *REFERENCES(I)* privileges on any column referenced by a foreign key.

The *CREATE VIEW* (create view) privilege permits the user to create a view that references the tables' columns. Finally, in order for a user to access the tables underlying a view, the subject must have *REFERENCE VIEW* (reference view) privilege on all of the tables' columns referenced in the view.

The distribution of access privileges to various users and groups of users is specified in the ACL attached to the table columns. The ACL specifies the distribution of *DELETE, SELECT(I), INSERT(I), UPDATE(I), REFERENCES(I), CREATE VIEW(I)* and *REFERENCE VIEW(I)* privileges. Basic privileges can be granted to other users and groups using the *GRANT* statement if the user has the *WITH GRANT OPTION* privilege. The *NULL* privilege denies a user any access to a table. Granting the *NULL(I)* privilege requires the *GRANTNULL(I)* privilege.

(E) DAC Enforcement on Defined Views

The operations that may be exercised upon a Trusted RUBIX defined view are create, list, and destroy. A defined view may be created via the *CREATE VIEW* command if the user holds *WRITE* and *EXECUTE* privileges on the schema where the defined view is to be installed. A defined view may be destroyed, e.g., via the *DROP VIEW* command, if the user holds *EXECUTE* and *WRITE* privileges on the schema.

When a defined view is to be created, the user supplies a view expression. This expression is parsed to determine its semantic validity; this operation requires that the user be empowered to locate all constituent tables and determine their row structures (i.e., *EXECUTE* on their schema).

In addition, the user must have *CREATE VIEW(I)* privilege on each of the underlying column in tables or views. The parsed view expression is stored in the database and constitutes the defined view. Note that, aside from this parsed string, there is no actual data storage associated with a defined view. To use a view (e.g., select from, insert into, delete from, or update a view), the user must hold *REFERENCE VIEW(I)* privilege on all tables referenced columns in the view definition.

The view ACL is initialized differently from base table ACLs because the view is permitted to override the privileges on the underlying tables. When a view is created, the creator's ACL entry is set to the intersection of the creator's ACL entries on the corresponding attributes of the underlying tables. The ability to further propagate privileges on a view is governed by the *WITH GRANT OPTION* of the SQL *GRANT* statement.

To delete a row from a view a user must hold the *DELETE* privilege on all the columns.

The view privileges *INSERT*, *UPDATE* and *SELECT* can be specified with respect to a subset of its columns. For the *SELECT* operation to succeed, the user should hold column level *SELECT* privilege on all the columns being retrieved. The *INSERT* operation succeeds if the user has column level *INSERT* privilege on all the columns being inserted. The *UPDATE* operation succeeds if the user has column level *UPDATE* privilege on all the columns being updated.

To SELECT from a view that resides in the information schema the user must, in addition to those privileges mentioned above, also hold EXECUTE and READ privilege to the object being listed.

The *NULL(I)* privilege denies a user any direct access to column "I" in a view. Granting the *NULL(I)* privilege requires the *GRANTNULL(I)* privilege.

(F) DAC Enforcement on Indexes

The operations that may be exercised upon a Trusted RUBIX index are create and destroy. An index may be created via the *CREATE INDEX* command if the user holds *WRITE* and *EXECUTE* privileges on the schema where the index is to be created. An index may be destroyed, e.g., via the *DROP INDEX* command, if the user holds *EXECUTE* and *WRITE* privileges on the schema.

6.1.4.3 DAC Enforcement on Listing Objects

Trusted RUBIX allows a user to list catalogs in database, schemas in catalog, tables, views, and indices in schema. Any reference to listing of objects implies a search (i.e., select operation) of the relevant information schema views. Users are allowed to list an object if they are operating within the DBMS Database Administrator role or if they have *EXECUTE* and *READ* privileges to the object to be listed, *EXECUTE* and *SELECT* privileges to the referenced columns in the information schema view, and REFERENCE VIEW privileges on the view's underlying table columns.

6.1.4.4 DAC Enforcement on Retrieving Privileges On DAC Objects

Trusted RUBIX allows a user to retrieve the current ACLs on all objects to which DAC applies. The information can be retrieved from the information schema using various SQL queries. Users are allowed to retrieve an object's ACL information (i.e., object name, grantee, grantee group, privilege type, grantable indicator) if they are operating within the DBMS Database Administrator role or if they have *EXECUTE* and *READ* privileges to the object to be retrieved and *EXECUTE* and *SELECT* privileges to the referenced columns in the information schema view, and REFERENCE VIEW privileges on the view's underlying table columns..

6.1.4.5 Summary Of Trusted RUBIX Discretionary Access Controls

A user does not need any special privileges to create a Trusted RUBIX database. The Trusted RUBIX privileges required for non-DBA users to perform various operations upon the Trusted RUBIX named objects are tabularized in Table 9 below.

Table 9 Required Trusted RUBIX Privileges for Operations

	Trusted RUBIX DAC Privilege				
Operation	**Database**	**Catalog**	**Schema**	**Table**	**View**
Alter Session Set Label	-	-	-	-	-
Alter Table**	EXECUTE	EXECUTE	EXECUTE, WRITE	-	-
Commit	-	-	-	-	-
Connect	EXECUTE	-	-	-	-
Create Catalog	EXECUTE, WRITE	-	-	-	-
Create Database	-	-	-	-	-
Create Index	EXECUTE	EXECUTE	EXECUTE, WRITE	-	-

Operation	Trusted RUBIX DAC Privilege				
	Database	Catalog	Schema	Table	View
Create Schema	EXECUTE	EXECUTE, WRITE	-	-	-
Create Table**	EXECUTE	EXECUTE	EXECUTE, WRITE	-	-
Create View	EXECUTE	EXECUTE	EXECUTE, WRITE	CRVIEW(I) (if directly referenced in select clause), REFVIEW(I) (all underlying base table columns)	CRVIEW(I) (if directly referenced in select clause)
Delete from Table***	EXECUTE	EXECUTE	EXECUTE	DELETE	-
Delete from View***	EXECUTE	EXECUTE	EXECUTE	REFVIEW(I)	DELETE
Drop Catalog	EXECUTE, WRITE	-	-	-	-
Drop Database	ADMIN	-	-	-	-
Drop Index	EXECUTE	EXECUTE	EXECUTE, WRITE	-	-
Drop Schema	EXECUTE	EXECUTE, WRITE	-	-	-
Drop Table	EXECUTE	EXECUTE	EXECUTE, WRITE	-	-
Drop View	EXECUTE	EXECUTE	EXECUTE, WRITE	-	-
Grant on Catalog	EXECUTE	Privilege with GRANT OPTION	-	-	-
Grant on Database	Privilege with GRANT OPTION		-	-	-

Operation	Trusted RUBIX DAC Privilege				
	Database	**Catalog**	**Schema**	**Table**	**View**
Grant on Schema	EXECUTE	EXECUTE	Privilege with GRANT OPTION	-	-
Grant on Table	EXECUTE	EXECUTE	EXECUTE	Privilege with GRANT OPTION	-
Grant on View	EXECUTE	EXECUTE	EXECUTE	REFVIEW(I)	Privilege with GRANT OPTION
Insert into Table***	EXECUTE	EXECUTE	EXECUTE	INSERT(I)	-
Insert into View***	EXECUTE	EXECUTE	EXECUTE	REFVIEW(I)	INSERT(I)
List of Catalogs (in addition to Select on view)*	EXECUTE, READ	-	-	-	-
List of Indices (in addition to Select on view)*	EXECUTE	EXECUTE	EXECUTE, READ	-	-
List of Schema (in addition to Select on view)*	EXECUTE	EXECUTE, READ	-	-	-
List of Tables (in addition to Select on view)*	EXECUTE	EXECUTE	EXECUTE, READ	-	-
List of Views (in addition to Select on view)*	EXECUTE	EXECUTE	EXECUTE, READ	-	-

Operation	Trusted RUBIX DAC Privilege				
	Database	**Catalog**	**Schema**	**Table**	**View**
Reference Table by foreign key constraint**	EXECUTE	EXECUTE	EXECUTE	REFERENCES(I)	-
Revoke on Catalog	EXECUTE	Privilege with GRANT OPTION	-	-	-
Revoke on Database	Privilege with GRANT OPTION		-	-	-
Revoke on Schema	EXECUTE	EXECUTE	Privilege with GRANT OPTION	-	-
Revoke on Table	EXECUTE	EXECUTE	EXECUTE	Privilege with GRANT OPTION	-
Revoke on View	EXECUTE	EXECUTE	EXECUTE	REFVIEW(I)	Privilege with GRANT OPTION
Rollback	-	-	-	-	-
Select from Table***	EXECUTE	EXECUTE	EXECUTE	SELECT(I)	
Select from View*, ***	EXECUTE	EXECUTE	EXECUTE	REFVIEW(I)	SELECT(I)
Update a Table***	EXECUTE	EXECUTE	EXECUTE	UPDATE(I)	-
Update a View***	EXECUTE	EXECUTE	EXECUTE	REFVIEW(I)	UPDATE(I)
sub-query on view***	EXECUTE	EXECUTE	EXECUTE	REFVIEW(I)	SELECT(I)
sub-query on table***	EXECUTE	EXECUTE	EXECUTE	SELECT(I)	-

[*] Listing objects is performed by issuing a SELECT operation on the corresponding Information Schema View. Therefore, the special DAC rules apply for the object being listed as well as the general DAC rules for the View being queried.

[**] Referencing a Table by a foreign key constraint is an implied operation when either an ALTER TABLE or CREATE TABLE command is issued which creates a foreign key constraint.

[***] The sub-query operation is an implied operation when a DELETE, SELECT, or UPDATE command is issued with a WHERE clause or when an INSERT command is issued with a SELECT clause.

6.1.5 Database Import and Export

Trusted RUBIX provides several mechanisms to support bulk transfer of data into and out of the database. These import and export operations (*`rximport`* and *`rxexport`* commands) enable the user to load data into the database, optionally at the level specified, and extract data from the database into a text file. This functionality is quite useful, but again, it is highly critical to protect these programs. When running in privileged mode, these programs may cause data to be entered into the Trusted RUBIX environment using containers, which do not match the original or targeted level of the data.

6.1.5.1 Export of User Data (DP.EXPT)

The ***`rxexport`*** command lists all rows in a table or view to a file or standard output. The data listed by *`rxexport`* is immediately usable by *`rximport`*. Authorized DBMS users may use it to export data without the row sensitivity label. Authorized DBMS users may also specify an option to list the data with the row sensitivity label as the first column. Authorized DBMS users also are provided with options to either use the default or the specified database, catalog, schema, delimiter, or output file for exporting a table or view. This command can only be run on the server. All MAC and DAC rules apply to *`rxexport`* as if the user were selecting the table or view using an SQL statement. To successfully use this command, the MAC and DAC rules are listed below:

- The user session sensitivity label must dominate the targeted table.
- If a table is being exported, the user must have *`SELECT`* privilege on all columns of the targeted table and *`EXECUTE`* privileges on the containing database, catalog, and schema; or be operating within the DBMS Database Administrator role.
- If a view that is not in the information schema is being exported, the user must have *`SELECT`* privilege on all columns of the targeted view, *`REFERENCE VIEW`* privilege on all table columns upon which the view is build, and *`EXECUTE`* privileges on the

containing database, catalog, and schema; or be operating within the DBMS Database Administrator role.

- If a view that is in the information schema is being exported, the user must have *SELECT* privilege on all columns of the targeted view, *REFERENCE VIEW* privilege on all table columns upon which the view is build, *EXECUTE* privileges on the containing database, catalog, and schema, and *EXECUTE* and *READ* privilege to the object being listed; or be operating within the DBMS Database Administrator role.

Only Rows dominated by session sensitivity label are returned.

6.1.5.2 Import of User Data (DP.IMPT)

The ***rximport*** command reads data from a text file, inserting rows into a table or view. The *rximport* command provides user options to either use the defaults or to specify a data file and a format file as input files for import and specific database, schema, and table or view name as the targeted table or view. The targeted table or view must be created before *rximport* is invoked. This command can only be executed on the server.

There are two methods of assigning labels to the imported rows:

- Rows to be assigned the current (static) session sensitivity label : When *rximport* is used without –m option, it assumes the import data have no label. Therefore, it will fail if label is in the first column in data record of the import file. When *rximport* is used in this manner, the session sensitivity label must dominate the table sensitivity label and each row loaded into the database is assigned the sensitivity label of the user session running the load operation. All rows are inserted and labeled at the user session sensitivity label. This single level loading function is available to a non-administrative DBMS user.

- Rows to be assigned a specified label (per row) : The Security Administrator can load rows and assign them a specified sensitivity label using the *rximport* command by specifying the *-m* flag. The *rximport* command with the "-m" option can only be used by a Security Administrator. Reclassification of data is described by three operations: upgrade (raise the label of the data as compared to the current session label); downgrade (lower the label of the data as compared to the current session label); and across (change the label of the data to one that is incomparable as compared to the current session label). Each of the three operations is controlled by an individual authorization. Each row is inserted and labeled with the row sensitivity label as specified by the first column in each data record of the import file.

To successfully use this command, the user must have *INSERT* privilege on the targeted table or view and *EXECUTE* privileges on the containing database, catalog, and schema. If the command is used on a view the user must also have the *REFERENCE VIEW* privilege on all table columns upon which the view is built. If the user does not have the proper privileges they must be operating within the DBMS Database Administrator role.

The `rximport` command reads data from a text file and inserts that data into a preexisting table. The input data may be free formed or formatted. If no format file is provided, `rximport` reads the data file in free form mode. In this case each record of the data file produces one row in the table. Within each record, successive fields are used as the input for successive columns. If there are fewer fields in a record than there are columns, the unfilled columns are marked as unknown. If a format file is provided, `rximport` reads the data in formatted mode. The data file contains data representing the rows to be inserted and is interpreted according to the contents of the format file. The format file indicates the position and length of each input column, as well as the number of lines per row. In either case the data may be single level or multilevel.

6.1.6 Security Management (MT.ADMN)

6.1.6.1 Security Management Roles (MT.ROLE)

Trusted RUBIX provides a role-based mechanism to allow trusted administrative users to supercede the explicit MAC and DAC security policies in performing certain administrative functions. These roles are established by granting specific authorizations to users via the underlying operating system's Identification and Authentication (I&A) component in tandem with the Role Based Access Control component. When a user initiates Trusted RUBIX while having an authorization, the user implicitly has all of the privileges associated with the corresponding role.

Each user who attains access to Trusted RUBIX is placed into a specific role. Trusted RUBIX defines the following five default roles:

- DBMS Audit Administrator
- Database Administrator;
- Database Operator;
- DBMS Security Administrator;
- Non-administrative DBMS User.

These default roles are implemented as Trusted Solaris 8 execution profiles that are created during the installation of Trusted RUBIX. These default roles and their associated authorizations are:

- DBMS Audit Administrators (AUD or Auditor) defined by the *rubix.audit.** authorizations;
- Database Administrators (DBA) defined by the *rubix.dac.**, and *rubix.admin.** `authorizations`;
- Database Operators (OP) defined by the *rubix.restore.backup.*, rubix.restore.logs.ls.*, rubix.restore.logs.rm.**, and *rubix.restore.logs.set.** authorizations.

- DBMS Security Administrators (SA) defined by the *rubix.mac.** and *rubix.*.grant*, *rubix.restore.create.**, authorizations;
- Non-Administrative DBMS users optionally with the *rubix.user.** authorizations;

The corresponding administrative responsibility and the Trusted RUBIX provided security management functions for each role are described in the following sections. The following table shows the relationship between the Trusted RUBIX default roles and security management operations they can perform.

Operation	Audit Admin.	Database Admin.	Operator	Security Admin.	User
Produce Audit Report	X	-	-	-	-
Set Audit Criteria	X	-	-	-	-
List Audit Log Files	X	-	-	-	-
Delete Audit Log Files	X	-	-	-	-
Set Audit Log Directory/Files	X	-	-	-	-
DAC Exemption CREATE, DELETE, DROP, GRANT, INSERT, REVOKE, SELECT, UPDATE	-	X	-	-	-
Database Restore (Create)	-	-	-	X	-
Drop Database	-	X	-	-	-
List Current Database	-	X	-	-	-
Move Current Database	-	X	-	-	-
Database Backup	-	-	X	-	
List Restore Log Files	-	-	X	-	-
Delete Restore Log Files	-	-	X	-	-
Set Restore Log Directory	-	-	X	-	-
Alter Session Set Label (Lower)	-	-	-	X	-

Operation	Audit Admin.	Database Admin.	Operator	Security Admin.	User
Alter Session Set Label (Raise)	-	-	-	X	-
Alter Session Set Label (Raise - Read Only)	-	-	-	X	-
Upgrade Row Label	-	-	-	X	-
Downgrade Row Label	-	-	-	X	-
Grant Authorizations	-	-	-	X	-
Multi-level Data Import	-	-	-	X	-
Data Export	-	-	-	-	X
Data Import	-	-	-	-	X

6.1.6.2 Audit Administrator Management Functions (MT.AUDM)

The Audit Administrator (AUD, i.e., auditor) bears sole responsibility for administering the Trusted RUBIX audit subsystem and ensuring that all Trusted RUBIX users (especially those performing administrative functions) are accountable for their actions. For default Trusted RUBIX roles, the DBMS Audit Administrators (AUD or Auditor) is defined by having the *rubix.audit.** authorizations. This role enables the auditor to perform the following activities:

- Establish audit criteria as necessary to assure that users (and administrators) are accountable for their actions (`rxauditset`).
- Examine the recorded audit data (`rxauditrpt`).
- Manage audit log files for a specific database (`rxlogs`)

It is possible, and sometimes recommended, that there be more than one user on a system capable of executing the responsibilities of the auditor. Since an auditor has the capability to disable the recording of actions that he performs, it is suggested that (given sufficient qualified staff), users capable of operating in this role not be allowed to operate in other administrative roles.

(A) Establish audit criteria (`rxauditset`)

The Trusted RUBIX `rxauditset` command can only be run on the server by those in the role of Audit Administrator. The `rxauditset` command allows an auditor to specify or display the criteria that determine whether an audible event is eligible to be recorded by Trusted RUBIX in the audit trails. The audit criteria to be selected via `rxauditset` command are:

- System-wide basis,
- Specific event type,
- Specific object identity,
- Specific user identity,
- Specific subject sensitivity label,
- Objects with a specific sensitivity label, and
- Objects within a range of sensitivity labels.

Any updates made to the audit event list will not affect the existing server processes. All new invocations of the database server will use the updated audit event list.

(B) Examine audit data (*`rxauditrpt`*)

The `rxauditrpt` command can only be run on the server by those in the role of Audit Administrator to view and generate audit data reports. The `rxauditrpt` command is a utility allowing the audit administrator to extract and view a specified subset of the audit trail based upon certain criteria. The criteria (known as post-selection criteria) available to an auditor to specify which events are to be extracted from the audit trail include:

- Event status (success/failure);
- Database name;

(C) Manage Audit Logs (*`rxlogs`*)

The `rxlogs` command gives authorized users the ability to list, delete, move, and set audit log files for a specific database. The log files are created as operations that are performed on a database. Over time these files may need to be removed to free up disk space.

6.1.6.3 Database Administrator Management Functions (MT.DBAM)

The Database Administrator (DBA) role is empowered to perform all operations that maintain the consistency and integrity of the stored data. There may be more than one DBA on a system. The DBA is only DAC exempt, not MAC exempt. To alter the Access Control Lists (ACLs) on a table, the DBA must be operating at the table’s label. If the DBA’s label strictly dominates the table’s label then the table's ACL cannot be altered. If the table’s label strictly dominates the DBA’s label, then the table will not be visible to the DBA.

For default Trusted RUBIX roles, the Database Administrators (DBA) is defined by having the *rubix.dac.** and *rubix.admin.** authorizations. Thus, the DBA is DAC exempt and may perform system-wide database management functions and manage the ACLs on Trusted RUBIX objects via SQL `GRANT` and `REVOKE` statements.

The DBA may perform the following operations that are exclusively limited to this role:

- Supersede the DAC security policy (i.e., *rubix.dac.** authorization) as the MAC policy permits.
- Drop, List, or Move databases (`rxdb`) as the MAC policy permits.

(A) Drop, List, or Move Database (`rxdb`)

The `rxdb` command can be used to drop, list, or move a database. Options can be used to qualify these operations. When used to drop a database, it is an alternative to the SQL DROP DATABASE command and will work even if the database has been corrupted. When used to display database(s) information the information can include version, security level and name. The move operation is equivalent to a rename operation. To drop a database the user must have the `rubix.admin.db.rm.dbname` authorization. To list a database the database sensitivity label must be dominated by the user's Trusted Solaris 8 session sensitivity label and the user must have the `rubix.admin.db.ls.dbname` authorization. To move a database the database sensitivity label must equal the user's Trusted Solaris 8 session sensitivity label and the user must have the `rubix.admin.db.mv.dbname` authorization.

6.1.6.4 Database Operator Management Functions (MT.OPRM)

The Database Operator (OP) role is charged with the responsibility of performing database backups via the use of the `rxdump` command. No other responsibilities are associated with this role. Note that, although operators are able to back-up entire databases, they hold no special capability as far as reading or writing those databases, but are instead bound by the standard MAC and DAC mechanisms. For default Trusted RUBIX roles, the Database Operators (OP) is defined by having the *rubix.restore.backup.*, rubix.restore.logs.ls.*, rubix.restore.logs.rm.*,* and *rubix.restore.logs.set.** authorizations.

(A) Database Backup (`rxdump`)

The Trusted RUBIX `rxdump` command can only be run by those in the role of Trusted RUBIX Database Operator. The `rxdump` command dumps all committed operations of exactly one database to a specified output device. All records that were created by previously committed transactions in the database will be backed up as of the *latest consistent moment* (*lcm*) from the start of the program; only records whose timestamp is less than or equal to the *lcm* will be written. For this reason a perfectly consistent backed-up dataset can be attained while the Trusted RUBIX system is up and running. This command should be run at `SYSHIGH` MAC level to ensure that all records are read. The `rxdump` tape spans as many tapes as is required to backup the database. Newly created `rxdump` tapes contain a 32-bit CRC, which is used to verify the contents of the tape.

(B) Manage Restore Logs (`rxlogs`)

The `rxlogs` command gives authorized users the ability to list, delete, move, and set restore log files for a specific database. The log files are created as operations that are performed on a database. Over time these files may need to be removed to free up disk space.

6.1.6.5 Security Administrator Management Functions (MT.SADM)

The Security Administrator (SA) is responsible for all operations, which may arbitrarily determine the label of a DBMS object. For default Trusted RUBIX roles, the DBMS Security Administrators (SA) defined by the *rubix.mac.**, *rubix.restore.create.**, and *rubix.*.grant* authorizations. The Security Administrator may:

- Reclassify the security level of rows of data using the Trusted RUBIX command `rxrecl`,
- Change the database session level using the SQL command `ALTER SESSION SET LABEL`,
- Import multilevel data into the database using the Trusted RUBIX command `rximport` with the *"`-m`"* option, and
- Perform a database restore (*rxrestore*).

(A) Reclassification of rows (`rxrecl`)
The `rxrecl` command gives user operating in the Security Administrator role the ability to change the label of stored rows in a table. Reclassification of data is described by three operations: upgrade (raise the label of the data); downgrade (lower the label of the data); and across (change the label of the data to one that is incomparable). Each of the three operations is controlled by an individual authorization.

Trusted RUBIX associates each DBMS object on the system with a sensitivity label. This sensitivity label is stored in the '`ROWLABEL`' hidden column associated with each row. Reclassification of data is done on a primary key basis. Rows are specified (i.e., named) in a data file by their primary key along with the existing label of the row and the new label to upgrade or downgrade to, with the system performing all necessary checks and audits to ensure that the operation is a valid one.

Row reclassification in Trusted RUBIX is governed by the polyinstantiation method used when the table was created. In the POLYNONE case, any row can be reclassified, because the row with the specified primary key is unique. In the POLYHIGH case, the reclassification is rejected if and only if the table already contains a row with the same primary key at the new level. In the POLYLOW case, the reclassification is rejected if and only if the table already contains a row with the same primary key at a level equal to or dominated by the new level.

(B) Change database session security level (`ALTER SESSION SET LABEL`)
The user must be in the DBMS Security Administrator role to use the `ALTER SESSION SET LABEL` SQL command, which gives authorized administrators the ability to change database session sensitivity label. The session sensitivity label may be set to `DBHIGH`, `DBLOW`, the current operating system session sensitivity label, or a specified sensitivity label. There must not be a current transaction. All database operations that follow will be executed at the new session security level. Trusted RUBIX allows the DBMS security administrator to alter session label within

the database for the range of labels for which the DBMS security administrator has been cleared by the operating system. The session label may be changed to one that strictly dominates the original session label, one that strictly dominates the original session label, or one that is incomparable to the original session label. These three alter session modes are controlled by individual authorizations. Furthermore, the DBMS Security Administrator may subsequently submit either read-only or read-write transactions depending on the authorizations held.

(C) Import multilevel data (*`rximport`*)

This *`rximport`* command can only be executed on the server. Only Security Administrators are allowed to use the multilevel option for import. The Trusted RUBIX *`rximport`* command with the "*`-m`*" option gives the security administrators the ability to import rows into a table or view and specify the object sensitivity label of each row. For multilevel data import operation, table row sensitivity label is specified as the first data field of the import data lines for that row. Reclassification of data is described by three operations: upgrade (raise the label of the data as compared to the session label); downgrade (lower the label of the data as compared to the session label); and across (change the label of the data to one that is incomparable as compared to the session label). Each of the three modes of operation is controlled by an individual authorization.

(D) Perform a database restore (*`rxrestore`*)

The *`rxrestore`* command can only be run on the server machine by those in the role of Security Administrator. The *`rxrestore`* command creates a new database from existing backup and transaction log files. This is a two-fold process in which the backup is restored and then, overlaid with transaction logging entries from previously committed transactions. The effects of previously rolled back transactions are not applied to the database upon restore. The database must remain off-line until the entire restoration process is completed. The *rxrestore* command restores exactly one database from the specified file into the current directory. The database name must not already exist as defined in the current directory. Only Security Administrator with the authorization of *rubix.restore.create.dbname* are allowed to use this command.

6.1.6.6 Non-administrative DBMS User Management Functions (MT.USRM)

The DBMS User is a user of the underlying operating system who has the Discretionary Access Control (DAC) permissions to use the Database. The DBMS User role is the default role associated with all non-administrative users whose officially sanctioned duties do not require them to perform trusted actions. Subjects executing in this role are not enrolled in any of the system administrative groups introduced above. For default Trusted RUBIX roles, the Non-Administrative DBMS users are optionally with the *rubix.user.** authorizations. The User may:

- Submit SQL operations as the DAC and MAC policies permit,

- Import single level rows into the database (`rximport`) as the DAC and MAC policies permit,
- Export rows from the database (`rxexport`) as the DAC and MAC policies permit, and

The import (`rximport)` and export functions (`rxexport`) enable the user to load data into the database and extract data from the database into a text file.

(A) SQL Operations and Transaction Control

The users SQL statements and the required MAC condition are described in Table 8. The users SQL statements and the required DAC privileges are described in Table 9.

The SQL statements also include transaction control statements (e.g., `COMMIT, ROLLBACK`). Users' applications should complete each transaction explicitly with either the `COMMIT` or `ROLLBACK` statement. Ceasing execution without ending the transaction causes either a `COMMIT` or `ROLLBACK` according to Trusted RUBIX-defined criteria. The `ROLLBACK` statement allows the user to complete the current transaction; close any cursors that the transaction opened; and provide that any changes made during the transaction do not take effect.

(B) Export Command (`rxexport`)

The export command, `rxexport`, extracts data from the user specified table or view into a text file or standard output. This command can only be run on the server. All Mandatory Access Control and Discretionary Access Control rules apply as if the user were selecting the table using an SQL statement. The data listed by `rxexport` is immediately usable by `rximport`.

Users specify the table or view name to be exported with the export options to indicate whether the row sensitivity label is to be placed as the first column in the output line and whether all versions of polyinstantiated rows are to be listed.

Users can also use options to specify the database name, schema name, and file name to be used for the `rxexport` operation.

(C) Import Command (`rximport`)

The `rximport` command can only be executed on the server. The imported rows are labeled with the user's session sensitivity label. All Mandatory Access Control and Discretionary Access Control rules apply as if the user were inserting into the table or view using an SQL statement.

6.1.6.7 Security Attribute Protection (MT.MSA)

The Security Attribute Protection function validates the user's role, authorization, privileges, and the session sensitivity label to enforce the TSP of DAC and MAC prior to process user's request for security attribute modification. The Security Attribute Protection function validates and ensures that only secure values are accepted for security attributes: (1) ACL privileges grant or revoke, (2) row sensitivity label reclassification, and (3) session security level alteration.

6.1.7 Residual Information Protection (DP.URIP)

This Residual Information Protection security function performs residual information protection and ensures that any residual information content of any resources is unavailable to all objects upon the resource's allocation. When Trusted RUBIX drops a database all of the operating system files are removed. The OS is relied upon to perform object reuse of the files correctly. Trusted RUBIX has only two basic objects that may be reused, the page and the record within the page. All other object allocation relies on one or both of these basic object allocations. For example, a table is made up of its associated data pages. The table's structure and ACL information are stored in other information schema tables. Catalogs and schemas are simply tables that hold table and view names. Thus, every storage object, other than a row, is a table constructed of pages. Rows within a table correspond to a record in a page. When a page is allocated its entire space is overwritten with zeros. Thus, each newly allocated segment or page starts out clean. New records within the page are allocated when a record is inserted into that page. Only the exact number of bytes necessary for the record is allocated and the new record data is immediately written into all of the bytes of that specific allocated record. It should be noted that due to the Multiversion Timestamp Ordering (MVTO) scheduling protocol Trusted RUBIX objects are not physically deallocated until a garbage collection mechanism initiates the process. This initiation may be delayed for a potentially unbounded amount of time. Because the object reuse policy is only concerned with objects as they are reused, this delay in deallocation has no impact on the security aspects of the reuse policy.

6.1.8 Trusted Recovery (PT.TRCV)

Trusted RUBIX handles three common failure conditions that a DBMS may encounter and recovers it back to a consistent and secure state after a failure.

- A transaction failure;
- An inconsistent state of main memory (system failure); and
- A primary disk error (media failure).

The following sections describe the logging, backup, restore, and recovery sub-functions for supporting the TOE Trusted Recovery (PT.TRCV) from the failures listed above.

6.1.8.1 Logging (PT.TRCV.LOGT)

In Trusted RUBIX, logs are used to record operations that have been submitted to the database. These logs are used to restore the database to a consistent state after one of the previously mentioned failure conditions occurs.

Trusted RUBIX has two types of logs: the rollback/recovery log and the restore log. The rollback/recovery log is used to recover from transaction and system failures. One such log is maintained for each database and the files corresponding to this log are stored in the database directory. The restore log is maintained in an administrator-defined location and is used to recover from a media failure. The state of each log is placed on persistent storage at proper times in order to guarantee the database can be placed back in a correct state following a failure.

6.1.8.2 Transaction Failure and Recovery (PT.TRCV.RCVT)

A transaction failure is due to a partially executed transaction performing an operation that violates the notion of correctness (i.e., serializability) defined for transactions. Examples of this in the Trusted RUBIX system are read/write conflicts. When this condition is detected, the effects of the transaction are removed from the database and the transaction is marked as aborted. This is known as a "transaction rollback." The need for transaction rollback is detected by the Trusted RUBIX and the rollback is performed automatically. Transaction rollback may also be initiated by the user.

Transaction recovery is performed on-line automatically by Trusted RUBIX system as transaction failures occur. Rollback is executed by reading back through the rollback/recovery log and removing the effects of all changes the transaction made to the database. Sufficient disk space must be available for maintenance of the rollback/recovery log.

6.1.8.3 System Failure and Recovery (PT.TRCV.RCVS)

The second failure condition is an inconsistent state of main memory due to a system failure. This can result from an abnormal shutdown of the system or a programming error that cannot be recovered from using the state of main memory. In this case, the system is recovered using the state of the primary disk(s). The need for system recovery is detected by Trusted RUBIX and is performed automatically. Typically, system recovery should not seriously delay the execution of transactions.

System recovery is performed automatically on a per-database basis as the database is initially opened. During the recovery process, no transactions may execute. The recovery process takes the current state of the database on persistent storage along with the rollback/recovery log and produces a correct database state reflecting the updates of all transactions that committed prior to the system failure. Sufficient disk space must be available for maintenance of the rollback/recovery log.

6.1.8.4 Media Failure and Restore (PT.TRCV.RCVM)

The third failure condition is a primary disk error (media failure). Examples of this are a physically damaged hard drive or an abnormal shutdown that results in a partial disk write. In this case, the system is generally recovered using the state of a back up storage device such as a non-primary disk or tape drive. This is known as "media restore."

Media restore is the responsibility of the Trusted RUBIX Operator (OP) and can delay transaction processing. Prior to a media failure, an image of the database is dumped to a non-primary disk or tape by the Operator. If a media failure occurs, the media is first checked for fatal defects and, if found, replaced. The previously dumped database image is then restored and the contents of the restore log are added by the Trusted RUBIX operator.

6.1.8.5 Backup (PT.TRCV.BKUP)

It is important to consider accidental data loss and/or loss of data integrity, while installing any computer system. Trusted RUBIX comes with a set of utilities to recover from catastrophic loss. Trusted RUBIX provides Operators flexibility in implementing their backup paradigm. An Operator can fully backup databases, to tape or disk, at selected intervals. An Operator can also log all database transactions between full system backups to tape or disk. In Trusted RUBIX, operators are assigned the task of performing a full backup. The command must be issued at the highest level at which data will be retrieved. Thus, the invoker must be `SYSHIGH` to guarantee the backup of all data.

It is also advisable to make full archives as opposed to full backups. A full archive is a backup that will not be overwritten. Full backups are only valid until the next backup is produced. The newer backup supersedes the older backup, whereas, archiving can be used for recording database status at appropriate times in the life of a database. The archive produced can be kept as a permanent record, and can be used at a later date should the need arise.

The full backup schedule that is selected is determined by database activity. If the database in question is very dynamic, then backups should be more frequent than for an equivalent relatively static database.

6.1.8.6 Restore (PT.TRCV.RSTR)

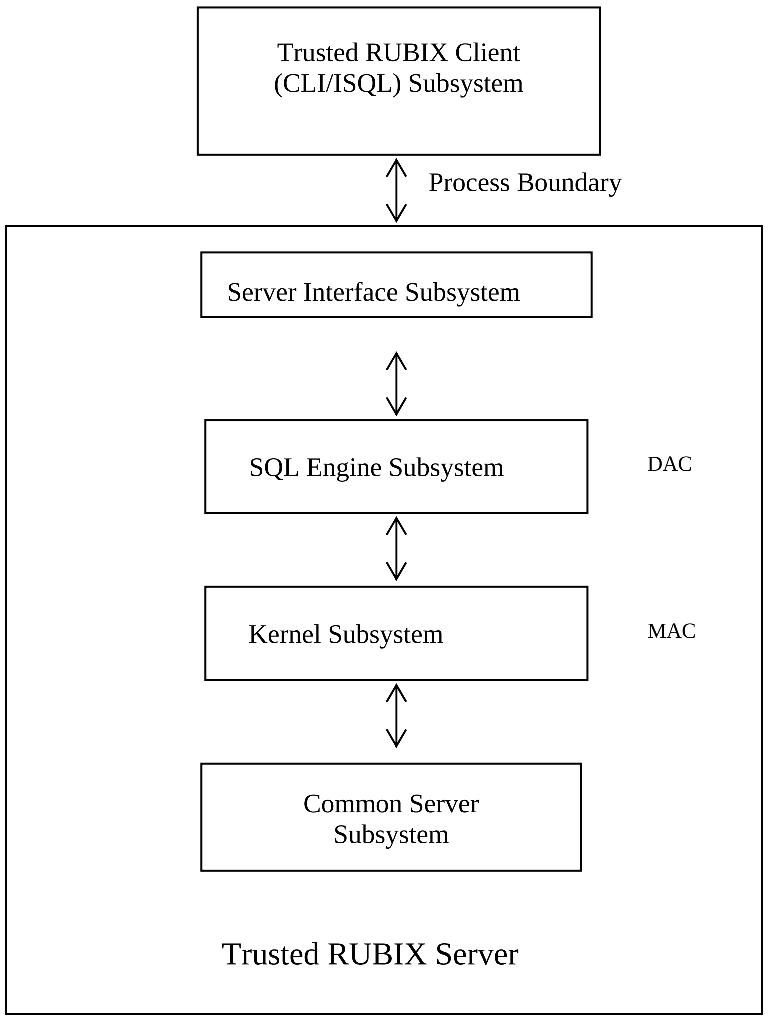
The restoration of data is a two-fold process in which the backup is restored and then, overlaid with the transaction logging entries. Only the Trusted RUBIX Security Administrator (SA) is allowed to perform this operation.

The database must remain off-line until the entire restoration process is completed. The entire roll forward operation must be completed before any new transactions can be applied. The roll forward is part of the restore process. After the database has been restored from the dump file, the log directory is searched for log files. Each log file is applied to the database. Only effects from previously committed transactions are applied to the database. When all the log files have been applied, the *`rxrestore`* command ends and the database is restored and ready to use.

6.1.9 Protection of Security Functions (PT)

Trusted RUBIX implements the TOE Security Functions (TSFs) on the server machine, where the databases are being kept. This section will describe the Trusted RUBIX client/server architecture and major subsystems of the TSFs. Figure 2 shows the major subsystems of the Trusted RUBIX server and their relationship.

Figure 2 Trusted RUBIX Architecture



6.1.9.1 Reference Monitor (PT.RFMN)

The reference monitor security function ensures that the TSF is always invoked (i.e., non-bypassability) before any functions are allowed to proceed. The Trusted RUBIX client/server architecture is based on the Trusted RUBIX server receiving requests from the client for operation on the database. The Trusted RUBIX Server resides on the machine where the database is stored. Within the Trusted RUBIX Server, the Trusted RUBIX Server Interface subsystem is the client's only entry point into the TOE security functions of Trusted RUBIX. The Server Interface subsystem handles client's database operation requests and invokes the TSP enforcement security functions of the Trusted RUBIX Server. Therefore, no database operation can proceed prior to the TSP enforcement functions having been invoked. The Server Interface subsystem invokes the SQL engine subsystem, which is responsible for enforcing DAC policies for operation on the database. The SQL Engine subsystem checks for needed DAC privileges before submitting the operations on record files to the Kernel subsystem. The Kernel subsystem is responsible for enforcing all MAC restrictions on data objects. The Kernel subsystem checks the needed MAC sensitivity label of the user before performing the low level transaction and database operations.

6.1.9.2 Domain Separation and Protection Mechanism (PT.SEPT)

In most cases, the Trusted RUBIX client and the Trusted RUBIX server reside on different machines. The Trusted RUBIX server resides on the machine where the databases are kept and protected. Trusted RUBIX stores its objects in UNIX files using what is known as the container storage model. The most primitive storage structure for Trusted RUBIX objects are UNIX files. Each database is stored in multiple UNIX files. All UNIX files for a unique database are stored in a single UNIX directory. The directories for all databases are stored in a single *databases* directory. Because Trusted RUBIX uses this container storage model, the isolation and protection of Trusted RUBIX file and directory objects are accomplished by using the operating system's DAC, MAC, and domain separation and protection mechanisms. A special Trusted RUBIX user ID (`RUBIX`) and group ID (`RUBIXTP`) is created and reserved for this purpose. Each database file and directory is owned by the user/group pair of `RUBIX/RUBIXTP`. The Trusted RUBIX server, executing with user/group as `RUBIX/RUBIXTP`, views the database space as its own address space. For each of the directory and file objects, its permissions are set so only members of `RUBIXTP` group may access these objects. The Trusted RUBIX executables are restricted to the `RUBIXTP` group. No login user is allowed in the `RUBIXTP` group. These UNIX directories and files of the databases are labeled at system high sensitivity label to offer additional protections. Thus, only Trusted RUBIX executables labeled at system high may access the UNIX directories and files that contain DBMS objects.

Under a single directory that is named "databases", there is one subdirectory (named the same as the database name) per database representing each existing database in Trusted RUBIX. DBMS objects within a single database are stored in multiple UNIX files in this subdirectory. Trusted RUBIX treats each UNIX file as a container. The DBMS objects such as databases, catalogs, schemas, tables, views, indexes, and records are created from the container. Each of these abstract objects is assigned a security label and an access control list (ACL). Each UNIX file may hold information labeled at any sensitivity label. Trusted RUBIX ensures the isolation and protection of DBMS objects in the UNIX files and arbitrates subject access to each DBMS data object, which is associated with an ACL and a sensitivity label, based upon the MAC and DAC policies.

Login users are not assigned as members of `RUBIXTP` group, therefore, they cannot access any database files or directories that are owned by the user/group pair of `RUBIX/RUBIXTP`. The Trusted RUBIX server implements the Server Interface subsystem using a remote procedure call (RPC) as the only interface to be used by the client subsystems to submit operations on databases.

6.2 TOE Security Assurance Measures

Each of the following subsections provides a list of names of the TOE evaluation evidences as security assurance measure for each EAL4 class. It also lists the security assurance requirements to be met, a description of the assurance evidence, and the rationale demonstrating these evidences meet the assurance requirements.

6.2.1 Configuration Management (CM)

Infosystems Technology, Inc. (ITI) uses the automated UNIX Revision Control System (RCS) for tracking version changes of Trusted RUBIX 5.0 product and TOE documentation. ITI also provides Trusted RUBIX 5.0 Configuration Management Manual for CM Plan and CM procedures.

The Security Assurance Requirements satisfied by these CM system and CM documentation are: ACM_AUT.1, ACM_CAP.4, and ACM_SCP.2.

6.2.1.1 ACM Assurance Measures

The CM system used for Trusted RUBIX 5.0 is the automated UNIX configuration management system known as Revision Control System (RCS).

The Configuration Management Document provided by ITI for the Trusted RUBIX 5.0 is the Trusted RUBIX 5.0 Configuration Management manual. The manual includes the configuration management plan and procedures for using the CM system.

6.2.1.2 Rationale for ACM_AUT.1 Authorization controls

All configuration management plan and procedures are documented in the Trusted RUBIX 5.0 Configuration Management manual, including procedures for checking source file revisions in and out, compiling the TOE, and examining the history of file revisions.

The source representation of Trusted RUBIX 5.0 is managed by the automated UNIX configuration management system known as Revision Control System (RCS) and is described in the Trusted RUBIX 5.0 Configuration Management manual. This CM system provides the capability to restrict file access to authorized individuals, which mitigates the potential for any unauthorized changes to be introduced in the system.

The Trusted RUBIX 5.0 Configuration Management manual describes the procedures necessary to automatically generate a binary representation of the TOE using the UNIX tools “imake”, “make”, and “makedepend”. Together, these tools provide the capability to automatically detect changes in source representation files when generating the binary representation of the TOE.

6.2.1.3 Rationale for ACM_CAP.4 CM capabilities

The Trusted RUBIX 5.0 Configuration Management manual describes the CM system used by the developer, provides a configuration item (CI) list of uniquely identifiable items that comprise release 5.0 of Trusted RUBIX, and documents CM and acceptance plans.

The CI list uniquely identifies each of the following types of files: TOE source files, API include files, “make” files, and on-line documentation files. The method for establishing the unique identification for each CI is documented in the Trusted RUBIX 5.0 Configuration Management manual.

The Unix RCS CM system is available for review to demonstrate operational compliance with the CM plan and effective management of the CIs. The evidence of using the CM system can be obtained through reviewing the historical log information for each configuration item. RCS provides security features for ensuring only authorized changes are made to configuration items and supports the generation of the TOE.

The Configuration Management manual contains an acceptance plan that describes the procedures used to accept modified or newly created configuration items as part of the TOE.

6.2.1.4 Rationale for ACM_SCP.2 Problem tracking CM coverage

The Trusted RUBIX 5.0 Configuration Management manual describes the methods and procedures used to track the TOE implementation source files and TOE documentation including: design documents, test documents, guidance documentation, CM documents, and security flaws. Security flaws are maintained in a web-based tool, which records needed changes in sufficient detail to coordinate source code modification kept in the RCS tool.

6.2.2 Delivery and Operation (DO)

The Delivery and Operation Documents provided by ITI for the Trusted RUBIX 5.0 include:

- Trusted RUBIX 5.0 Delivery and Operation manual,
- Trusted RUBIX 5.0 Configuration Management Manual

The Security Assurance Requirements satisfied by the document are: ADO_DEL.2 and ADO_IGS.1.

6.2.2.1 ADO Assurance Measures

The Trusted RUBIX 5.0 Delivery and Operation manual provides the procedures used to deliver both partial and complete instantiations of the TOE to the user. The Release Creation section in the Configuration Management manual describes the procedures necessary for installation and generating the binary representation of the TOE.

6.2.2.2 Rationale for ADO_DEL.2 Detection of modification

The Trusted RUBIX 5.0 Delivery and Operation manual provides a description of the procedures used to deliver both partial and complete instantiations of the TOE to the user. These procedures provide a narrative of the mechanisms used to detect discrepancies between the developer's master copy of the TOE and the version received at the delivery location. Additionally, the procedures counter masquerade attempts, through the use of checksum values, even in cases in which the developer has sent nothing to the user's site. These delivery procedures and scripts are available for review as evidence of use.

6.2.2.3 Rationale for ADO_IGS.1 Installation, generation, and start-up procedures

The Trusted RUBIX 5.0 Delivery and Operation manual and Configuration Management manual together provide a complete description of the procedures required for the secure installation, generation, and start-up of the TOE. The Configuration Management manual describes the procedures necessary for generating the binary representation of the TOE prior to customer delivery. Additionally, the Delivery and Operation manual documents the steps necessary for correctly configuring, generating, and starting up Trusted RUBIX 5.0 in a secure state.

6.2.3 Development (DV)

ITI provides all the required TOE development documentation necessary for evaluating the product at EAL4.

The Security Assurance Requirements satisfied by these guidance documents are: ADV_FSP.2, ADV_HLD.2, ADV_IMP.1, ADV_LLD.1, ADV_RCR.1, and ADV_SPM.1.

6.2.3.1 ADV Assurance Measures

The Development Documents provided by ITI for Trusted RUBIX 5.0 include:

- Trusted RUBIX 5.0 Functional Specification (FSP),
- Trusted RUBIX 5.0 High Level Design (HLD) document,
- Trusted RUBIX 5.0 Low Level Design document,
- Trusted RUBIX 5.0 Security Functions to Functional Specification Correspondence document
- Trusted RUBIX 5.0 Functional Specification to High Level Design Correspondence document,
- Trusted RUBIX 5.0 High Level Design to Low Level Design Correspondence document,
- Trusted RUBIX 5.0 Security Policy Model document, and

- Trusted RUBIX 5.0 Implementation Subset.

6.2.3.2 Rationale for ADV_FSP.2 Fully defined external interfaces

Trusted RUBIX 5.0 Functional Specification document describes all the external interfaces, which include the functional application programming interfaces (API) and the administrative trusted programs and commands.

The Trusted RUBIX 5.0 Functional Specification provides special details for security-relevant functions. TOE Security Functions are identified and completely described while non-TSF interfaces are described and clearly indicated that they are not security-relevant functions. This document specifies all external interfaces and provides complete details of all effects, exceptions and error messages. Since the document contains a description of all external interfaces, including security functions, it consequently contains a complete representation of the TSF.

6.2.3.3 Rationale for ADV_HLD.2 Security enforcing high-level design

The Trusted RUBIX 5.0 High Level Design document describes the structure of the TSF in terms of subsystems and describes the security functionality provided by each subsystem of the TSF, such as the mandatory access control (MAC) and discretionary access control (DAC) security functions. This design document also identifies the underlying trusted operating system required by the TSF and the relationship between their MAC and DAC functions. The Trusted RUBIX 5.0 subsystems include:

- Trusted RUBIX Server Interface Subsystem
- SQL Engine Subsystem
- Kernel Subsystem
- Common Server Subsystem
- Common Library Subsystem
- RUBIX Administrative Commands

All interfaces to each subsystem are described individually and identified as TOE internal or external and either TSP-enforcing or non-TSP enforcing interfaces.

6.2.3.4 Rationale for ADV_IMP.1 Subset of the implementation of the TSF

A subset of Trusted RUBIX 5.0 implementation source files, which serves as a representation of the TSF-enforcing functions implementation, is available to the evaluation team.

6.2.3.5 Rationale for ADV_LLD.1 Descriptive low-level design

The Trusted RUBIX 5.0 Low Level Design document describes the hierarchical modular structure utilizing abstractions implemented by procedures and state data for the Trusted RUBIX 5.0 system. The purpose of each module and dependencies on other modules are defined.

The functions associated with each module are described as externally or internally visible interfaces and as TSP-enforcing or non TSP-enforcing interfaces.

The Low Level Design document provides each function in a module with the function purpose and method of use, details of effects, exceptions and error messages.

For functions that are TSP-enforcing, a description of the methods used in that TSP enforcing function is provided.

6.2.3.6 Rationale for ADV_RCR.1 Informal correspondence demonstration

The Trusted RUBIX 5.0 Security Functions to Functional Specification Correspondence document maps and illustrates that all security functions identified in the TOE Summary Specification are completely and accurately represented in the Trusted RUBIX 5.0 Functional Specification.

The Trusted RUBIX 5.0 Functional Specification to High Level Design Correspondence document depicts the correct and complete design of all security functions from the Functional Specification in the High Level Design. For each security function identified in the functional specification, the correspondence document indicates which TSF subsystems are involved in the support of the function.

The Trusted RUBIX 5.0 High Level Design to Low Level Design Correspondence document illustrates the complete and correct refinement of all security functions from the High-Level Design in the Low-Level Design. For each TSF subsystem described in the high-level design, the correspondence document indicates which TSF modules identified in the low-level design are involved in the support of the subsystem.

The Trusted RUBIX 5.0 Low Level Design to Implementation Correspondence mapping provides evidence that the selected TSF-enforcing function(s) from the low level design is completely and correctly represented in the corresponding source code implementation.

6.2.3.7 Rationale for ADV_SPM.1 Informal TOE security policy model

Trusted RUBIX 5.0 Security Policy document describes the interactions between all subjects and objects within the scope of MAC and DAC. Additionally, the document models the security policy associated with Identification and Authentication (I&A), Audit Management, Object Reuse, and operations associated with Trusted RUBIX roles.

The Security Policy Model covers:

- Simple Security Property
- *-Property
- Discretionary Security Property

Rationale is provided to demonstrate consistent and complete correspondence with the Functional Specification.

6.2.4 Guidance Documents (GD)

The Guidance Documents provided by ITI for Trusted RUBIX 5.0 include:

- Trusted RUBIX 5.0 Trusted Facility Manual (TFM) as the administrator guidance document and
- Trusted RUBIX 5.0 Security Features User's Guide (SFUG) as the user guidance document.

The Security Assurance Requirements satisfied by these guidance documents are: AGD_ADM.1 and AGD_USR.1.

6.2.4.1 AGD Assurance Measures

The TFM is primarily intended for the security and administrative personnel responsible for managing the trusted aspects of the system. It instructs the Trusted RUBIX administrators on how to properly configure Trusted RUBIX for secure operations, describes functions and privileges to be controlled when running a trusted facility, details the security policy, its implementation procedures, audit mechanisms and various roles associated with the secure operation of Trusted RUBIX. The TFM provides guidelines on the consistent and effective use of the Trusted RUBIX protection features and its interaction with the trusted operating system.

The SFUG details the uses and interactions of the various security protections available to non-administrative users, describes the user-accessible security functions, and contains warnings to users about functions that should be controlled with examples. The Trusted RUBIX Reference Guide details the full syntax of all commands used in this document, including all security mechanisms.

6.2.4.2 Rationale for AGD_ADM.1 Assurance Measures

The TFM provides administrator guidance for system administrative personnel and describes the various administrative roles, functions, interfaces, and administrative security features necessary to administer the TOE in a secure manner. Warnings are distinguished with separated text and a centered "Warning" label, to highlight functions and privileges that should be controlled in a secure processing environment.

Assumptions regarding user behavior are included to advise the administrator of potential security risks from user actions. For example, the TOE assumes successful authentication to the underlying operating system (OS). The inherent security risk in this approach is the potential for the user to select a weak password for OS authentication.

The security parameters, and the corresponding possible values, associated with each interface are discussed in the appendix of security interfaces. For example, the *`rxdump`* command, which is used to perform a dump of the database, should be run at the `SYSHIGH` MAC level to ensure all records are read during processing.

The TFM discusses all security-relevant events relative to the administrative functions that need to be performed, including: selection of personnel to perform administrative roles, audit file capture and review, database backup & recovery, reclassification of database entities, and management of access control lists (ACLs) for database entities.

The interfaces listed in the TFM and SFUG documents are consistent with the interfaces identified in the Functional Specification, High-Level Design, and Low-Level Design.

The TFM provides additional security requirements for the IT environment that must be enforced to achieve a secure TOE configuration, including the physical secure countermeasures necessary to ensure backup media is safeguarded from destruction or theft.

6.2.4.3 Rationale for AGD_USR.1 Assurance Measures

The SFUG provides user guidance for all functions and interfaces available to the non-administrative user. The guidance includes description, syntax, options explanation, and example usage for each interface. Warnings associated with each security interface are offset in bold text to increase visibility.

The SFUG presents all user responsibilities for the secure operation of the TOE, including assumptions regarding user behavior. For example, the SFUG includes the assumption that most users will authenticate at the highest privilege permitted and record data at that level, even when the information does not require that level of protection.

In addition to describing the TOE security requirements, the SFUG describes all relevant security requirements for the IT environment. Specifically, the SFUG describes the dependency on the underlying OS for the available database security labels.

The interfaces listed in the SFUG documents are consistent with the interfaces identified in the TFM, Functional Specification, High-Level Design, and Low-Level Design.

6.2.5 Life Cycle Support (LC)

The Life Cycle Support Document provided by ITI for the Trusted RUBIX 5.0 is the Trusted RUBIX 5.0 Life Cycle Support Document.

The Security Assurance Requirements satisfied by this document are: ALC_DVS.1, ALC_LCD.1, and ALC_TAT.1.

6.2.5.1 ALC Assurance Measures

The Life Cycle Support Document is primarily intended to describe the software development model, security measures, and development tools used to construct the TOE. The processes and procedures described in the document provide the assurance that software is developed and maintained without adverse impact to TOE security functions.

6.2.5.2 Rationale for ALC_DVS.1 Identification of security measures

The Trusted RUBIX 5.0 Life Cycle Support Document provides a description of the physical and procedural countermeasures used at ITI to protect the TOE during development. Evidence proving compliance with these security measures is highlighted in the document.

6.2.5.3 Rationale for ALC_LCD.1 Developer defined life-cycle model

The traditional "Waterfall Model" is the life-cycle model used to develop Trusted RUBIX 5.0. The Life Cycle Support Document illustrates each stage in the model and describes the activities ITI performs during each stage. As an example, the Engineering Change Request (ECR) process describes the actions necessary to incorporate new functionality into Trusted RUBIX. Specifically, the process describes the individual roles and responsibilities in the ECR process as well as the tools used to track the request.

6.2.5.4 Rationale for ALC_TAT.1 Well-defined development tools

Software development tools provide additional assurance that TOE Security Functions operate as specified. ITI uses several tools for development and verification of software developed in the "C" language. For example, ITI use a tool named "Insure++" to verify correct memory usage and detect memory overruns and un-initialized variables. Additionally, the UNIX debugging tool "gdb" is used to trace software during runtime in an effort to locate defects and assure correct execution.

6.2.6 Security Testing (TE)

6.2.6.1 ATE Assurance Measures

Trusted RUBIX 5.0 Software Test Plan describes the plan for testing performed by the SQL regression test suite of Trusted RUBIX 5.0 relational database management system and associated internal interface specifications.

The Security Assurance Requirements satisfied by this test documents are: ATE_COV.2, ATE_DPT.1, and ATE_FUN.1.

6.2.6.2 Rationale for ATE_COV.2 Analysis of coverage

Section 1.3.2 of the Trusted RUBIX 5.0 Software Test Document presents a mapping between the Functional Specification sections that contain TSF's and the sections in the Trusted RUBIX 5.0 Software Test document that present the corresponding security tests.

6.2.6.3 Rationale for ATE_DPT.1 Testing: high-level design

Section 1.3.2 of the Trusted RUBIX 5.0 Software Test Document provides the analysis of the depth of testing. The depth analysis provides the mapping between Security Tests and the subsystems in High Level Design, and demonstrates that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.

6.2.6.4 ATE_FUN.1 Functional testing

The Trusted RUBIX 5.0 Software Test document consists of test plans, test procedure descriptions, expected test results and actual test results. Section 3 provides the test plan information of the required test environment in terms of software, hardware, and configuration. Section 4 provides test plan information of test tools to help execute automated and non-automated testing of the Trusted RUBIX™ 5.0 product. Section 5 of the test document identifies the security functions of the Trusted RUBIX server to be tested in the test suites and describe the goal of tests to be performed.

Sections 6, 7, and 9 of the Trusted RUBIX 5.0 Software Test document demonstrate the correspondence between the TSF as described in the functional specification and the test suites identified in the test documentation is complete by mapping each TSFI in the functional specification to one or more test suites.

Section 6 of the test document is the test procedure description of the TSFI of Interactive SQL (ISQL) for Trusted RUBIX Client application to be tested in these test suites. Section 7 of the test document is the test procedure description of the TSFI of Call Line Interface (CLI) for Trusted RUBIX Client application to be tested in these test suites. Section 8 of the test document describes the Kernel Subsystem test suite. Section 9 of the test document is the test procedure description of the TSFI of for Trusted RUBIX Commands to be tested in these test suites.

6.2.6.5 Rationale for ATE_IND.2 Independent testing – sample

The developer will provide the TOE that is suitable for testing. The developer will provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

6.2.7 Vulnerability Assessment (VA)

6.2.7.1 AVA Assurance Measures

The analysis documents provided by ITI for Trusted RUBIX 5.0 is the Misuse Analysis document, the Strength of Function Analysis, and the Trusted RUBIX 5.0 Vulnerability Analysis.

The Security Assurance Requirements satisfied by these analysis documents are AVA_MSU.2, AVA_SOF.1, and AVA_VLA.2.

6.2.7.2 Rationale for AVA_MSU.2 Validation of analysis

The Trusted RUBIX 5.0 Misuse Analysis analyzes whether the issue of misuse has been addressed in the user guidance, the administrator guidance, and the secure installation, generation, and start-up procedures.

6.2.7.3 Rationale for AVA_SOF.1 Strength of TOE security function evaluation

The developer will provide the Strength of Function Analysis document for Trusted RUBIX 5.0, for each security functions realized by a probabilistic or permutational mechanism.

6.2.7.4 Rationale for AVA_VLA.2 Independent vulnerability analysis

The developer Trusted RUBIX 5.0 Vulnerability Analysis documents an analysis of the TOE deliverables searching for obvious vulnerabilities and the disposition of these vulnerabilities, so that the vulnerability cannot be exploited in the intended environment for the TOE.

7 Rationale

This section provides the rationale for the selection of the IT security objectives, requirements, and TOE security functions.

Section 7.1 provides the rationale for the existence of the security objectives based upon the stated security policies and threats while Section 7.2 provides the lower-level rationale for the existence of functional and assurance components based upon the stated security objectives.

7.1 Rationale for Security Objectives

This section provides rationale for the security objectives demonstrating that security objectives are suitable to cover the intended environment.

7.1.1 Assumptions

A.ADMIN

It is assumed that database administrators, database operators, DBMS security administrators, DBMS audit administrators are competent, and merit trust place in them.

O.ENV_ADMIN is intended to ensure that A.ADMIN holds true in the environment of use by requiring that administrators be trained and provide evidence of their trustworthiness. This is an objective for the non-IT environment, and hence, no IT requirements are traced to it.

A.DISASTER

It is assumed that the DBMS is protected against disasters such as loss of power, fire, flood, and destruction of facilities.

O.ENV_DISASTER is intended to ensure A.DISASTER holds true in the environment of use by requiring that responsible persons provide the necessary protections. This is an objective for the non-IT environment, and hence, no IT requirements are traced to it.

A.PHYSICAL

It is assumed that the DBMS, host OS, and IT environment are protected from physical attack.

O.ENV_PHYSICAL is intended to ensure A.PHYSICAL holds true in the environment of use by requiring that responsible persons provide for the physical protection of the DBMS and its IT environment. This is an objective for the non-IT environment, and hence, no IT requirements are traced to it.

A.SECURE_COMMS

It is assumed that the environment protects information while it is in transit between the DBMS and components of the IT environment.

O.ENV_SECURE_COMMS is intended to ensure A.SECURE_COMMS holds true in the environment of use by requiring that responsible persons provide for secure communications between the TOE and external IT entities. This is an objective for the non-IT environment, and hence, no IT requirements are traced to it.

A.USERS

It is assumed that authorized DBMS users are familiar with applicable DBMS security policies and procedures, and merit the trust placed in them.

O.ENV_USERS is intended to ensure that A.USERS holds true in the environment of use by requiring that authorized users be trained, provide evidence of their trustworthiness, and acknowledge their responsibility to follow DBMS security policy and procedures. This is an objective for the non-IT environment, and hence, no IT requirements are traced to it.

7.1.2 Threats

T.ABUSE

An authorized DBMS user performs authorized actions that compromise (intentionally or otherwise) DBMS assets.

The TSF provides the capability to detect the compromise of DBMS assets by authorized DBMS users through the record of security-relevant events [O.TOE_AUDIT_GENERATION*] and the tools provided to review that record [O.TOE_AUDIT_REVIEW*]. DBMS audit administrators are trained to apply these review tools in accordance with appropriate policy and procedures [O.ENV_ADMIN] using the guidance documentation provided with the TOE [O.TOE_ADMIN_GUIDANCE]. In addition, authorized users are trained in the appropriate use of the DBMS [O.TOE_USER_GUIDANCE] and acknowledge their responsibility for their actions [O.ENV_USERS]. Positive identification of users [O.TOE_USER_IDENTIFICATION*, O.IT_ENV_USER_IDENTIFICATION, O.IT_ENV_USER_AUTHENTICATION*] also counters this threat since the knowledge that authorized DBMS users can be held individually to account for their actions does itself act as a deterrent.

The TSF relies on the IT environment to manage IT environment resources efficiently in order to mitigate unintentional exhaustive consumption of resources by authorized users [O.ENV_TRUST_IT].

T.ADMIN_ERROR*

An attacker may exploit vulnerabilities in the DBMS caused by improper administration of the DBMS in order to compromise DBMS assets.

The developer prevents this threat by providing adequate guidance for the administration of the DBMS [O.TOE_ADMIN_GUIDANCE]. Database administrators and DBMS security administrators are adequately trained to use the guidance and procedures [O.ENV_ADMIN] and follow any applicable security policies and procedures [O.ENV_CONFIG].

T.ADMIN_ROGUE*

A database administrator, database operators, DBMS security administrator, or DBMS audit administrator performs actions that intentionally compromise DBMS assets.

Requiring administrators to provide evidence of their trustworthiness [O.ENV_ADMIN] diminishes this threat. The TSF further mitigates the threat by limiting administrators' capabilities using roles [O.TOE_ADMIN_ROLE*].

T.AUDIT_CORRUPT*

An attacker may cause audit records to be lost or modified, or may prevent future records from being recorded by taking actions to exhaust audit storage capacity, thus masking an attacker's actions.

The TSF prevents unauthorized modifications to security audit data and functions [O.TOE_AUDIT_PROTECTION, O.IT_ENV_AUDIT_PROTECTION*, O.TOE_AUDIT_GENERATION *]. The training of administrators [O.ENV_ADMIN] and proper configuration of the TOE [O.ENV_CONFIG] support the TSF in preventing unauthorized modifications.

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*]. Nor can attackers bypass the preventive measures by physically tampering with the TOE itself [O.ENV_PHYSICAL].

T.DOS*

An attacker may exhaustively consume IT environment resources in order to deny DBMS assets to authorized DBMS users.

The TSF relies on the IT environment to manage IT environment resources efficiently in order to mitigate this threat [O.ENV_TRUST_IT].

T.EAVESDROP*

An attacker may gain unauthorized access to DBMS assets (e.g. authentication information or DBMS objects) when the data is transmitted to or from the DBMS.

In general, an attacker could gain unauthorized access to data when that data is in transit between the user and the TSF, between the TSF and an external IT product, or between the user and an external IT product. The IT environment prevents this attack in each of these cases [O.ENV_SECURE_COMMS, O.ENV_TRUST_IT]. In addition, the IT environment prevents disclosure of DBMS objects and DBMS assets (e.g., authentication information) in transit between parts of the TOE, when the TOE is physically distributed [O.IT_ENV_SELF_PROTECTION*].

T.EXPORT

An authorized DBMS user may send information (in soft or hard copy form) from DBMS objects to a recipient who is not authorized to see the information or who subsequently handles the information in a manner that is inconsistent with its sensitivity designation.

The environment mitigates this threat in that users are trained to follow appropriate information handling procedures [O.ENV_USERS] as described in the user guidance [O.TOE_USER_GUIDANCE].

T.IMPROPER_INSTALLATION*

An attacker may exploit vulnerabilities in the DBMS caused by improper delivery, installation, or configuration of the DBMS in order to compromise DBMS assets.

The developer prevents this threat by providing guidance and procedures for the secure delivery and installation of the DBMS [O.TOE_ADMIN_GUIDANCE]. Database administrators and DBMS security administrators are adequately trained to use the guidance and procedures [O.ENV_ADMIN] and follow any applicable security policies and procedures [O.ENV_CONFIG].

T.INSECURE_START*

An attacker may exploit vulnerabilities in the DBMS created during start up or restart of the DBMS or host OS in order to compromise DBMS assets.

Database administrators and/or DBMS security administrators mitigate this threat by following appropriate procedures [O.ENV_ADMIN, O.TOE_ADMIN_GUIDANCE] when starting or restarting [O.TOE_RECOVERY*] the DBMS.

T.MASQUERADE*

An attacker or external IT entity may masquerade as an authorized DBMS user or a external IT entity in order to gain unauthorized access to DBMS objects or DBMS resources.

The TSF and the security function of the host OS prevents masquerading by requiring users to identify themselves [O.TOE_USER_IDENTIFICATION*, O.IT_ENV_USER_IDENTIFICATION*] and verifying the identities claimed by users [O.IT_ENV_USER_AUTHENTICATION*]. In order to prevent masquerading in the form of session hijacking, communication between authorized DBMS users (both human and IT products) and the TSF are protected [O.ENV_SECURE_COMMS].

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*]. Nor can attackers bypass the preventive measures by physically tampering with the TOE itself [O.ENV_PHYSICAL].

T.POOR_DESIGN*

An attacker may exploit vulnerabilities in the DBMS caused by unintentional or intentional errors in requirements specification, design or development in order to compromise DBMS assets.

The processes used to develop the DBMS mitigate this threat by eliminating potential vulnerabilities through sound development practices [O.TOE_SOUND_DEVELOPMENT*] and management of the DBMS and its development evidence during development [O.TOE_DEVEL_CM*].

T.POOR_IMPLEMENTATION*

An attacker may exploit vulnerabilities in the DBMS caused by unintentional or intentional errors in implementing the design of the DBMS in order to compromise DBMS assets.

The process used to develop the TSF mitigates this threat by eliminating potential vulnerabilities through sound development practices [O.TOE_SOUND_DEVELOPMENT*] and management of the DBMS and its development evidence during development [O.TOE_DEVEL_CM*].

Testing of the TSF further mitigates this threat. The TSF is tested both by the developer and independently, including analysis and testing for vulnerabilities [O.TOE_TESTING*].

T.REPLAY*

An attacker may gain unauthorized access to DBMS assets by replaying authentication information corresponding to an authorized DBMS user.

The TSF and IT environment mitigate this threat by protecting authentication information in transit between the user and the TSF [O.ENV_SECURE_COMMS].

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the host OS [O.IT_ENV_SELF_PROTECTION*].

T.SPOOFING*

An attacker may masquerade as the DBMS or an external IT entity in the IT environment and communicate with authorized DBMS users who incorrectly believe they are communicating with the DBMS or external IT entity.

The IT environment prevents spoofing by providing users with secure communication to the TSF [O.ENV_SECURE_COMMS]. Users are trained to follow the appropriate procedures [O.ENV_USERS] as described in the user guidance [O.TOE_USER_GUIDANCE].

Attackers cannot bypass these preventive measures by circumventing the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*].

T.SYSACC*

An attacker may gain unauthorized access to the account of a database administrator, a database operators, a DBMS security administrator, a DBMS audit administrator, or other trusted personnel including IT environment administrators.

The TSF and the security function of host OS prevents unauthorized access to administrative accounts by requiring users to identify themselves [O.TOE_USER_IDENTIFICATION*, O.IT_ENV_USER_IDENTIFICATION*] and verifying the identities claimed by users [O.IT_ENV_USER_AUTHENTICATION*]. These preventive measures are supported by the applicable security policies and procedures [O.ENV_CONFIG]. Administrators can be relied on to follow these policies and procedures [O.ENV_ADMIN].

Auditing [O.TOE_AUDIT_GENERATION*, O.IT_ENV_TIME] and review of the audit trail [O.TOE_AUDIT_REVIEW*] provide a limited capability for detection of unauthorized access to administrative accounts.

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*]. Nor can attackers bypass the preventive measures by physically tampering with the TOE itself [O.ENV_PHYSICAL]. The security function of the TOE protects the DBMS audit trail [O.TOE_AUDIT_PROTECTION] and the security function of host OS protects the OS audit trail [O.IT_ENV_ AUDIT_PROTECTION*] in support of the measures to detect this attack.

T.UNATTENDED_SESSION*

An attacker may gain unauthorized access to DBMS assets using an unattended session of an authorized DBMS user.

Since the DBMS is a server application, the TSF relies on the IT environment (e.g., host OS of the client applications) to manage user sessions. Hence, the IT environment security functions [O.ENV_TRUST_IT] together with administrator and user cooperation [O.ENV_ADMIN, O.ENV_USERS, O.TOE_USER_GUIDANCE] mitigate this threat.

T.UNAUTH_ACCESS*

An attacker may gain unauthorized access to DBMS assets either via the DBMS itself or via the IT environment.

The TSF prevents unauthorized access to DBMS assets. The TSF enforces a discretionary access control policy to ensure that DBMS users gain access only to DBMS assets for which they are authorized [O.TOE_DISCRETIONARY_ACCESS*]. The TSF also enforces a mandatory access control policy to ensure that DBMS users gain access only to DBMS assets for which they have the necessary clearance [O.TOE_MANDATORY_ACCESS*]. The mandatory access control policy is based on labels that mark the security level of both information and subjects. The security administrators of the host OS manage user clearances [O.ENV_CONFIG] and the DBMS administrators manage DBMS object sensitivity labels [O.ENV_ADMIN] in accordance with security policy and procedures.

In addition, the TSF and security function of the host OS work together to clear residual information from storage objects to prevent unauthorized access to information when storage objects are reused. The host OS removes data from system objects before reallocating them for use by the TSF or other host OS subjects [O.ENV_TRUST_IT]. The TSF removes data from DBMS objects before reallocating them to DBMS users, including returning to a secure state after service discontinuity [O.TOE_RESIDUAL_INFORMATION*, O.TOE_RECOVERY*].

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*]. Nor can attackers bypass the preventive measures by physically tampering with the TOE itself [O.ENV_PHYSICAL].

More complex means of gaining unauthorized access to DBMS assets are expressed via the threats T.SYSACC*, T.UNATTENDED_SESSION*, T.MASQUERADE*, T.SPOOFING*, T.REPLAY*, T.EXPORT, and T.EAVESDROP*.

T.UNAUTH_MODIFICATION*

An attacker may make unauthorized modifications to the DBMS security policy data or unauthorized use of security functions.

The TSF prevents unauthorized modifications to DBMS security policy data or functions by limiting the ability to make modifications to administrative roles for access control [O.TOE_DISCRETIONARY_ACCESS*, O.TOE_MANDATORY_ACCESS*], identification [O.TOE_USER_IDENTIFICATION*], and security audit [O.TOE_AUDIT_PROTECTION*]. The training of administrators [O.ENV_ADMIN] and proper configuration of the TOE [O.ENV_CONFIG] support the TSF in preventing unauthorized modifications.

Auditing [O.TOE_AUDIT_GENERATION*, O.IT_ENV_TIME] and review of the audit trail [O.TOE_AUDIT_REVIEW*] provide a limited capability for detection of unauthorized modifications to security policy data or functions.

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*]. Nor can attackers bypass the preventive measures by physically tampering with the TOE itself [O.ENV_PHYSICAL].

The host OS security function supports the TSF in user identification and authentication [O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*] and protects the audit trail [O.IT_ENV_AUDIT_PROTECTION*] in support of the measures to detect this attack.

T.UNDETECTED_ACTIONS*

In order to compromise DBMS assets, an attacker may successfully introduce vulnerabilities into the DBMS, or repeatedly exploit vulnerabilities in the DBMS, without being detected by the DBMS.

The TSF mitigates the threat of undetected, unauthorized actions by providing the capability to record security-relevant events [O.TOE_AUDIT_GENERATION*], by providing DBMS audit administrators with tools to review the record of events [O.TOE_AUDIT_REVIEW*], and relying on the security function of the host OS to protect that record of events [O.IT_ENV_AUDIT_PROTECTION*]. DBMS audit administrators are trained to use the audit tools (i.e., select appropriate security-relevant events to audit) [O.ENV_ADMIN]. See also T.UNIDENTIFIED_ACTIONS*.

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*]. Nor can attackers bypass the preventive measures by physically tampering with the TOE itself [O.ENV_PHYSICAL].

T.UNIDENTIFIED_ACTIONS*

In order to compromise DBMS assets, an attacker may successfully introduce vulnerabilities into the DBMS, or repeatedly exploit vulnerabilities in the DBMS, without being identified by the DBMS audit administrator.

The TSF mitigates the threat of administrators failing to identify unauthorized actions by providing the capability to record security-relevant events [O.TOE_AUDIT_GENERATION*] and by providing DBMS audit administrators with tools to review the record of events [O.TOE_AUDIT_REVIEW*]. The TSF relies on the security function of the host OS to protect the audit trail [O.IT_ENV_AUDIT_PROTECTION*]. The DBMS audit administrators are trained to use these tools to follow appropriate policy and procedures [O.ENV_ADMIN, O.TOE_ADMIN_GUIDANCE]. See also T.UNDETECTED_ACTIONS*.

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*]. Nor can attackers bypass the preventive measures by physically tampering with the TOE itself [O.ENV_PHYSICAL].

T.UNKNOWN_STATE*

An attacker may exploit vulnerabilities in the DBMS created by a failure of the DBMS or host OS in order to compromise DBMS assets.

Database administrators and/or DBMS security administrators mitigate this threat by following appropriate procedures [O.ENV_ADMIN, O.TOE_ADMIN_GUIDANCE] when restarting [O.TOE_RECOVERY*] the DBMS after a failure or service discontinuity.

T.USER_CORRUPT*

An attacker may make unauthorized deletions or modifications to DBMS data.

The TSF prevents this attack by enforcing the discretionary and mandatory access control policies [O.TOE_DISCRETIONARY_ACCESS*, O.TOE_MANDATORY_ACCESS*], thereby limiting the capability to delete or modify DBMS data to authorized users [O.TOE_USER_IDENTIFICATION*]. The training of users [O.ENV_USERS, O.TOE_USER_GUIDANCE] helps mitigate inadvertent deletions or modifications by users with delete and/or modify authorization.

Auditing [O.TOE_AUDIT_GENERATION*, O.IT_ENV_TIME] and review of the audit trail [O.TOE_AUDIT_REVIEW*] provide a limited capability for detection of unauthorized deletions or modifications of DBMS data.

Attackers cannot interfere, tamper, or bypass these preventive measures by circumventing the security functions of the TSF [O.TOE_SELF_PROTECTION*] or the security functions of the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*]. Nor can attackers bypass the preventive measures by physically tampering with the TOE itself [O.ENV_PHYSICAL]. The security function of the host OS provides user identification and authentication [O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*] and protects the audit trail [O.IT_ENV__AUDIT_PROTECTION*] in support of the measures to detect this attack.

7.1.3 Organizational Security Policies

P.ACCOUNT*

The users of the DBMS shall be held individually accountable for their actions within the DBMS.

The TSF meets this policy by maintaining a record of security relevant events [O.TOE_AUDIT_GENERATION*] and providing DBMS audit administrators with tools to review that record [O.TOE_AUDIT_REVIEW*]. Each security relevant event is associated with the user responsible for the event [O.TOE_USER_IDENTIFICATION*, O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*]. Users acknowledge responsibility for their actions [O.ENV_USERS].

P.AUTHORIZATION*

The DBMS must limit the extent of each user’s abilities in accordance with the DBMS security policy.

The TSF meets this policy by limiting the capabilities of roles [O.TOE_ADMIN_ROLE*] with respect to access control [O.TOE_DISCRETIONARY_ACCESS*, O.TOE_MANDATORY_ACCESS*], identification [O.TOE_USER_IDENTIFICATION*, O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*], and security audit [O.TOE_AUDIT_PROTECTION*, O.IT_ENV_AUDIT_PROTECTION*]. The training of administrators [O.ENV_ADMIN] and proper configuration of the TOE [O.ENV_CONFIG] support the TSF in the correct implementation of this policy.

P.AUTHORIZED_USERS*

Only those users who have been authorized to access the information within the DBMS may access the DBMS and users will be granted authorization in accordance with the principle of least privilege.

The TSF meets this policy by restricting access to DBMS assets to authorized users whose identities [O.TOE_USER_IDENTIFICATION*] have been verified by the host OS [O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*]. The host OS security administrators follow applicable policies and procedures (e.g., least privilege) in managing user accounts and the database administrators follow applicable policies and procedures in managing the access control lists of DBMS objects [O.ENV_CONFIG].

P.CLEARANCE*

The DBMS must limit access to the protected DBMS assets to authorized users whose security level is appropriate for the labeled data.

The TSF meets this policy by restricting access to DBMS assets based on the sensitivity of the information and the clearance of the user requesting access [O.TOE_MANDATORY_ACCESS*]. The TSF obtains the clearance of a user’s identity [O.TOE_USER_IDENTIFICATION*], which has been verified by the host OS [O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*] before associating authorizations with the user and granting access.

P.INDEPENDENT_TESTING*

The DBMS must undergo independent security functional and vulnerability testing as part of an independent vulnerability analysis.

The developer meets this policy through the process used to develop the TOE [O.TOE_SOUND_DEVELOPMENT*] and through the testing and evaluation of the TOE [O.TOE_TESTING*].

P.NEED_TO_KNOW*

The DBMS must limit the access to, modification of, and destruction of the information in DBMS assets to those authorized users who have an explicated stated, “need to know” for that information.

The TSF meets this policy through discretionary access controls [O.TOE_DISCRETIONARY_ACCESS*], which allow authorized users to restrict access to DBMS assets based on “need to know” of the authorized user requesting access [O.TOE_USER_IDENTIFICATION*, O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*].

P.RESOURCE_LABELS*

All DBMS assets must have associated labels identifying the security level of the data contained therein.

The TSF meets this policy through a mandatory access control policy [O.TOE_MANDATORY_ACCESS*], which includes the marking of all DBMS assets with appropriate sensitivity labels.

P.ROLES*

The database administrator, database operator, DBMS security administrator, and DBMS audit administrator shall have separate and distinct roles associated with them.

The TSF meets this policy by implementing distinct roles for database administrators, database operator, DBMS security administrators, DBMS audit administrators, and authorized DBMS users [O.TOE_ADMIN_ROLE*]. These roles are distinguished by their capabilities with respect to access control [O.TOE_DISCRETIONARY_ACCESS*, O.TOE_MANDATORY_ACCESS*], identification [O.TOE_USER_IDENTIFICATION*], and security audit [O.TOE_AUDIT_PROTECTION*, O.IT_ENV_AUDIT_PROTECTION*].

The security function of the host OS [O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*] supports the TSF in associating roles to users. The training of administrators [O.ENV_ADMIN] and proper configuration of the TOE [O.ENV_CONFIG] support the TSF in the correct implementation of this policy.

P.TRUSTED_RECOVERY*

After a DBMS failure or other operational discontinuity, the DBMS shall recover without a protection compromise. In all cases DBMS operation must begin in a secure state.

The TSF meets this policy by recovering from failures or discontinuities [O.TOE_RECOVERY*], with support from the IT environment [O.IT_ENV_SELF_PROTECTION*].

Database administrators and/or DBMS security administrators support this policy by following appropriate procedures [O.ENV_ADMIN, O.TOE_ADMIN_GUIDANCE] when starting or restarting the DBMS.

P.USER_CLEARANCE*

Each user must have a clearance identifying the maximum security level of data that the user may access.

The TSF meets this policy by associating appropriate sensitivity labels with subjects acting on behalf of each user [O.TOE_MANDATORY_ACCESS*]. These sensitivity labels correspond to the security level of the user [O.ENV_USERS, O.TOE_USER_IDENTIFICATION*, O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*].

7.1.4 Tracings Between Security Objectives And TOE Security Environment

This section contains tables showing the tracings between security objectives and the assumptions, threats, and organizational security policies identified in section 3, TOE Security Environment.

Table 10 identifies, for each assumption, the security objectives that address it.

Table 10 Security Objectives to Assumptions Tracing

Assumption	Objectives
A.ADMIN	O.ENV_ADMIN
A.SECURE_COMMS	O.ENV_SECURE_COMMS
A.DISASTER	O.ENV_DISASTER
A.PHYSICAL	O.ENV_PHYSICAL
A.USERS	O.ENV_USERS

Table 11 identifies, for each threat, the security objectives that address it.

Table 11 Security Objectives to Threats Tracing

Threat	Objectives
T.ABUSE	O.TOE_AUDIT_GENERATION* O.TOE_AUDIT_REVIEW* O.TOE_ADMIN_GUIDANCE O.TOE_USER_GUIDANCE O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION* O.ENV_ADMIN O.ENV_USERS O.ENV_TRUST_IT
T.ADMIN_ERROR*	O.TOE_ADMIN_GUIDANCE O.ENV_ADMIN O.ENV_CONFIG

Threat	Objectives
T.ADMIN_ROGUE*	O.TOE_ADMIN_ROLE* O.ENV_ADMIN
T.AUDIT_CORRUPT*	O.TOE_AUDIT_PROTECTION O.IT_ENV_AUDIT_PROTECTION* O.TOE_AUDIT_GENERATION* O.TOE_SELF_PROTECTION* O.IT_ENV_SELF_PROTECTION* O.ENV_ADMIN O.ENV_CONFIG O.ENV_PHYSICAL O.ENV_TRUST_IT
T.DOS*	O.ENV_TRUST_IT
T.EAVESDROP*	O.IT_ENV_SELF_PROTECTION* O.ENV_SECURE_COMMS O.ENV_TRUST_IT
T.EXPORT	O.TOE_USER_GUIDANCE O.ENV_USERS
T.IMPROPER_INSTALLATION*	O.TOE_ADMIN_GUIDANCE O.ENV_ADMIN, O.ENV_CONFIG
T.INSECURE_START*	O.TOE_ADMIN_GUIDANCE O.TOE_RECOVERY* O.ENV_ADMIN
T.MASQUERADE*	O.TOE_SELF_PROTECTION* O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION* O.IT_ENV_SELF_PROTECTION* O.ENV_SECURE_COMMS O.ENV_TRUST_IT O.ENV_PHYSICAL
T.POOR_DESIGN*	O.TOE_DEVEL_CM* O.TOE_SOUND_DEVELOPMENT*

Threat	Objectives
T.POOR_IMPLEMENTATION*	O.TOE_DEVEL_CM* O.TOE_SOUND_DEVELOPMENT* O.TOE_TESTING*
T.REPLAY*	O.TOE_SELF_PROTECTION* O.IT_ENV_SELF_PROTECTION* O.ENV_SECURE_COMMS
T.SPOOFING*	O.TOE_USER_GUIDANCE O.IT_ENV_SELF_PROTECTION* O.ENV_TRUST_IT O.ENV_SECURE_COMMS O.ENV_USERS
T.SYSACC*	O.TOE_AUDIT_PROTECTION* O.IT_ENV_AUDIT_PROTECTION* O.TOE_AUDIT_REVIEW* O.TOE_AUDIT_GENERATION* O.TOE_SELF_PROTECTION* O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION* O.IT_ENV_TIME O.IT_ENV_SELF_PROTECTION* O.ENV_CONFIG O.ENV_ADMIN O.ENV_TRUST_IT O.ENV_PHYSICAL
T.UNATTENDED_SESSION*	O.TOE_USER_GUIDANCE O.ENV_ADMIN, O.ENV_TRUST_IT O.ENV_USERS

Threat	Objectives
T.UNAUTH_ACCESS*	O.TOE_DISCRETIONARY_ACCESS* O.TOE_MANDATORY_ACCESS* O.TOE_RESIDUAL_INFORMATION* O.TOE_RECOVERY* O.TOE_SELF_PROTECTION* O.IT_ENV_SELF_PROTECTION* O.ENV_CONFIG O.ENV_ADMIN O.ENV_TRUST_IT O.ENV_PHYSICAL
T.UNAUTH_MODIFICATION*	O.TOE_AUDIT_PROTECTION* O.IT_ENV_AUDIT_PROTECTION* O.TOE_AUDIT_REVIEW* O.TOE_AUDIT_GENERATION* O.TOE_DISCRETIONARY_ACCESS* O.TOE_MANDATORY_ACCESS* O.TOE_SELF_PROTECTION* O.IT_ENV_AUDIT_PROTECTION* O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION* O.ENV_TRUST_IT O.IT_ENV_SELF_PROTECTION* O.IT_ENV_TIME O.ENV_ADMIN O.ENV_CONFIG O.ENV_PHYSICAL

Threat	Objectives
T.UNDETECTED_ACTIONS*	O. IT_ENV _AUDIT_PROTECTION* O.TOE_AUDIT_GENERATION* O.TOE_AUDIT_REVIEW O.TOE_SELF_PROTECTION* O.IT_ENV_SELF_PROTECTION* O.ENV_ADMIN O.ENV_PHYSICAL O.ENV_TRUST_IT
T.UNIDENTIFIED_ACTIONS*	O.TOE_AUDIT_REVIEW* O.IT_ENV_AUDIT_PROTECTION* O.TOE_ADMIN_GUIDANCE O.TOE_AUDIT_GENERATION* O.TOE_SELF_PROTECTION* O.IT_ENV_SELF_PROTECTION* O.ENV_ADMIN O.ENV_PHYSICAL O.ENV_TRUST_IT
T.UNKNOWN_STATE*	O.TOE_ADMIN_GUIDANCE O.TOE_RECOVERY* O.ENV_ADMIN

Threat	Objectives
T.USER_CORRUPT*	O.IT_ENV_AUDIT_PROTECTION* O.TOE_AUDIT_REVIEW* O.TOE_AUDIT_GENERATION* O.TOE_USER_GUIDANCE O.TOE_DISCRETIONARY_ACCESS* O.TOE_MANDATORY_ACCESS* O.TOE_SELF_PROTECTION* O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION* O.IT_ENV_SELF_PROTECTION* O.IT_ENV_TIME O.ENV_PHYSICAL O.ENV_TRUST_IT O.ENV_USERS

Table 12 identifies, for each organizational security policy, the security objectives that address it.

Table 12 Security Objectives to Organizational Security Policies Tracing

Policy	Objectives
P.ACCOUNT*	O.TOE_AUDIT_GENERATION* O.TOE_AUDIT_REVIEW* O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION* O.ENV_USERS

Policy	Objectives
P.AUTHORIZATION*	O.TOE_AUDIT_PROTECTION* O.IT_ENV_AUDIT_PROTECTION* O.TOE_ADMIN_ROLE* O.TOE_DISCRETIONARY_ACCESS* O.TOE_MANDATORY_ACCESS* O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION* O.ENV_ADMIN O.ENV_CONFIG
P.AUTHORIZED_USERS*	O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION* O.ENV_CONFIG
P.CLEARANCE*	O.TOE_MANDATORY_ACCESS* O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION*
P.INDEPENDENT_TESTING*	O.TOE_SOUND_DEVELOPMENT* O.TOE_TESTING*
P.NEED_TO_KNOW*	O.TOE_DISCRETIONARY_ACCESS* O.IT_ENV_USER_AUTHENTICATION* O.IT_ENV_USER_IDENTIFICATION* O.TOE_USER_IDENTIFICATION*
P.RESOURCE_LABELS*	O.TOE_MANDATORY_ACCESS*

Policy	Objectives
P.ROLES*	O.TOE_ADMIN_ROLE*
	O.TOE_AUDIT_PROTECTION*
	O.IT_ENV_AUDIT_PROTECTION*
	O.TOE_DISCRETIONARY_ACCESS*
	O.TOE_MANDATORY_ACCESS*
	O.IT_ENV_USER_AUTHENTICATION*
	O.TOE_USER_IDENTIFICATION*
	O.IT_ENV_USER_IDENTIFICATION*
	O.ENV_ADMIN
	O.ENV_CONFIG
P.TRUSTED_RECOVERY*	O.TOE_RECOVERY*
	O.TOE_ADMIN_GUIDANCE
	O.IT_ENV_SELF_PROTECTION*
	O.ENV_ADMIN
P.USER_CLEARANCE*	O.TOE_MANDATORY_ACCESS*
	O.IT_ENV_USER_AUTHENTICATION*
	O.IT_ENV_USER_IDENTIFICATION*
	O.TOE_USER_IDENTIFICATION*
	O.ENV_USERS

Table 13 identifies, for each TOE security objective, the assumptions, organizational security policies and threats addressed by that objective.

Table 13 Assumptions, Threats, and Organizational Security Policies to Security Objectives Tracing

Objectives	Policy/Threat/Assumptions

Objectives	Policy/Threat/Assumptions
O.TOE_ADMIN_GUIDANCE	P.TRUSTED_RECOVERY* T.ABUSE T.ADMIN_ERROR* T.IMPROPER_INSTALLATION* T.INSECURE_START* T.UNIDENTIFIED_ACTIONS* T.UNKNOWN_STATE*
O.TOE_ADMIN_ROLE	T.ADMIN_ROGUE* P.AUTHORIZATION* P.ROLES*
O.TOE_AUDIT_GENERATION*	P.ACCOUNT* T.ABUSE, T.AUDIT_CORRUPT* T.SYSACC* T.UNAUTH_MODIFICATION* T.UNDETECTED_ACTIONS* T.UNIDENTIFIED_ACTIONS* T.USER_CORRUPT*
O.TOE_AUDIT_REVIEW*	T.ABUSE T.SYSACC* T.UNAUTH_MODIFICATION* T.UNDETECTED_ACTIONS* T.UNIDENTIFIED_ACTIONS* T.USER_CORRUPT* P.ACCOUNT*

Objectives	Policy/Threat/Assumptions
O.TOE_AUDIT_PROTECTION*	T.AUDIT_CORRUPT* T.SYSACC* T.UNAUTH_MODIFICATION* P.AUTHORIZATION* P.ROLES*
O.TOE_USER_IDENTIFICATION*	T.ABUSE T.MASQUERADE* T.SYSACC* T.UNAUTH_MODIFICATION* T.USER_CORRUPT* P.ACCOUNT* P.AUTHORIZATION* P.AUTHORIZED_USERS* P.CLEARANCE* P.NEED_TO_KNOW* P.ROLES* P.USER_CLEARANCE*
O.TOE_DEVEL_CM*	T.POOR_DESIGN* T.POOR_IMPLEMENTATION*
O.TOE_DISCRETIONARY_ACCESS*	P.AUTHORIZATION* P.NEED_TO_KNOW* P.ROLES* T.UNAUTH_ACCESS* T.UNAUTH_MODIFICATION* T.USER_CORRUPT

Objectives	Policy/Threat/Assumptions
O.TOE_MANDATORY_ACCESS*	P.AUTHORIZATION* P.CLEARANCE* P.RESOURCE_LABELS* P.ROLES* P.USER_CLEARANCE* T.UNAUTH_ACCESS* T.UNAUTH_MODIFICATION* T.USER_CORRUPT*
O.TOE_RECOVERY*	P.TRUSTED_RECOVERY* T.INSECURE_START* T.UNAUTH_ACCESS* T.UNKNOWN_STATE*
O.TOE_RESIDUAL_INFORMATION*	T.UNAUTH_ACCESS*
O.TOE_SELF_PROTECTION	T.AUDIT_CORRUPT T.MASQUERADE* T.REPLAY* T.SYSACC* T.UNAUTH_ACCESS* T.UNAUTH_MODIFICATION* T.UNDETECTED_ACTIONS* T.UNIDENTIFIED_ACTIONS* T.USER_CORRUPT*
O.TOE_SOUND_DEVELOPMENT*	P.INDEPENDENT_TESTING* T.POOR_DESIGN* T.POOR_IMPLEMENTATION*
O.TOE_TESTING*	P.INDEPENDENT_TESTING* T.POOR_IMPLEMENTATION*

Objectives	Policy/Threat/Assumptions
O.TOE_USER_GUIDANCE	T.ABUSE T.EXPORT T.SPOOFING* T.UNATTENDED_SESSION* T.USER_CORRUPT*

7.2 Rationale for Security Requirements

7.2.1 Security Functional Requirements Rationale

This section provides rationale for the security functional requirements demonstrating that security functional requirements are suitable to address the security objectives.

7.2.1.1 Security Objectives for the TOE

O.TOE_ADMIN_ROLE*

To isolate administrative capabilities by providing separate and distinct security management roles associated with authorized DBMS users.

The TSF provides database administrator, DBMS security administrator, database operator, and DBMS audit administrator roles [FMT_SMR.1 Security Roles, FMT_SMF.1 Specification of Management Functions]. The TSF enforces the discretionary access control by restricting the DBMS security administrative functions to users associated with the authorized roles [FDP_ACC.2 Complete access control, FDP_ACF.1 Security attribute based access control].

O.TOE_AUDIT_GENERATION*

To provide the capabilities to detect security relevant events, create records of such events, and associate each record with the individual user who causes the event.

The TSF provides the capability to detect security relevant events and create an audit record of each such event [FAU_GEN.1 Audit data generation]. The audit record of an event can be associated with the individual user who caused the event [FAU_GEN.2 User identity association, FIA_ATD.1 User attribute definition, FIA_USB.1 User-subject binding]. The time a security relevant event occurs can be reliably determined based on information supplied by the IT environment [O.IT_ENV_TIME, O.ENV_TRUST_IT]. The TSF provides DBMS audit administrators [FMT_SMR.1 Security Roles, FMT_SMF.1 Specification of Management Functions] the capability to select which security relevant events are audited [FAU_SEL.1 Selective Audit, FMT_MOF.1 Management of security functions behavior, FMT_MTD.1 Management of TSF data].

O.TOE_AUDIT_PROTECTION*

To provide the capability to protect audit information.

The TSF protects the stored audit records from unauthorized deletion and/or modification. [FAU_STG.1 Protected audit trail storage]. The TSF prevents audit data loss if the audit trail is full [FAU_STG.4 Prevention of audit data loss]. The TSF protects audit information [O.TOE_AUDIT_GENERATION*] from unauthorized viewing [FAU_SAR.2 Restricted audit review].

O.TOE_AUDIT_REVIEW*

To provide the capability to selectively view audit information associated with individual DBMS users.

The TSF provides DBMS audit administrators [FMT_SMR.1 Security Roles, O.ENV_ADMIN, O.AUDIT_PROTECTION*] with tools to view and interpret audit information associated with DBMS users [FAU_SAR.1 Audit review] including the capability to search the audit information [FAU_SAR.3 Selectable audit review].

O.TOE_DISCRETIONARY_ACCESS*

To control accesses to DBMS assets based upon the identity of DBMS users and groups of DBMS users.

The TSF governs access to DBMS assets according to a discretionary access control policy [FDP_ACC.2 Complete access control] as enforced by access control functions [FDP_ACF.1 Security attribute based access control]. This policy restricts access to DBMS assets to authorized DBMS users [O.TOE_USER_IDENTIFICATION*].

It is also enforced on DBMS assets imported to [FDP_ITC.1 Import of user data without security attributes, FDP_ITC.2 Import of user data with security attributes] or exported from [FDP_ETC.1 Export of user data without security attributes, FDP_ETC.2 Export of user data with security attributes] the TSC. Access control decisions are based on the identity of the DBMS user, DBMS group identifier associated with that user, and access control attributes associated with the DBMS asset.

The policy is discretionary in that authorized DBMS users and database administrators have the capability to change the access control attributes associated with some DBMS assets [FMT_SMR.1 Security roles, FMT_MSA.2 Secure security attributes, FMT_MSA.1 Management of security attributes, FMT_MSA.3 Static attribute initialization, and FMT_REV.1 Revocation].

The discretionary access control policy is supported by the host OS [O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*, O.IT_ENV_SELF_PROTECTION*].

O.TOE_MANDATORY_ACCESS*

To control accesses to DBMS assets based upon the security level of users and the sensitivity of the DBMS assets.

The TSF governs access to DBMS assets in accordance with a mandatory access control policy [FDP_IFC.2 Complete information flow control] as enforced by information flow control functions [FDP_IFF.2 Hierarchical security attributes]. This policy restricts information flows among DBMS assets based on the clearances of authorized DBMS users [O.TOE_USER_IDENTIFICATION*]. It is also enforced on DBMS assets imported to [FDP_ITC.1 Import of user data without security attributes, FDP_ITC.2 Import of user data with security attributes] or exported from [FDP_ETC.1 Export of user data without security attributes, FDP_ETC.2 Export of user data with security attributes] the TSC. Information flow control decisions are based on the clearance level of the DBMS user and the sensitivity of the DBMS asset.

The policy is mandatory in that only DBMS security administrators have the capability to change the sensitivity attributes associated with DBMS assets [FMT_SMR.1 Security management roles, FMT_MSA.1 Management of security attributes, FMT_MSA.3 Static attribute initialization]. In particular, DBMS security administrators manage user security attributes. The mandatory access control policy is consistent with the policy enforced by the IT environment [O.ENV_TRUST_IT] and is supported by the host OS [O.IT_ENV_USER_IDENTIFICATION*, O.IT_ENV_USER_AUTHENTICATION*, O.IT_ENV_SELF_PROTECTION*].

O.TOE_RECOVERY*

To provide procedures and/or mechanisms to assure that, after a DBMS failure, host OS failure, or other discontinuity, the DBMS recovers without a protection compromise.

Since the TSF is an application, the TSF and IT environment cooperate to provide the services needed for recovery to a consistent and secure state after a failure or discontinuity [FPT_RCV.4 Function recovery, FDP_ROL.2 Advanced rollback, O.IT_ENV_SELF_PROTECTION*]. Database administrators use these services to restore the TSF in accordance with applicable policy and procedures [O.ENV_ADMIN, O.ENV_TRUST_IT].

O.TOE_RESIDUAL_INFORMATION*

The TSF and the IT environment cooperatively ensure that any information contained in a protected DBMS object or resource is not released when the DBMS object or resource is reallocated [FDP_RIP_DB.2 TOE full residual information protection and FDP_RIP_OS.2 Environment full residual information protection].

After specified failures, TOE security functions either complete successfully or the TSF recovers to a secure state [O.TOE_RECOVERY*], which ensures that DBMS objects and resources are cleared of residual information.

O.TOE_SELF_PROTECTION*

To ensure that the security functions of the DBMS cannot be bypassed, and to ensure that the security functions have a separate execution domain, which protects from interference and tampering by untrusted subjects.

The TOE Security Requirement ensures that the TOE security functions are always invoked and enforced [FPT_RVM_DB.1 TOE non-bypassability of the TSP]. The TOE SFR ensures the DBMS provides advance rollback for data integrity [FDP_ROL.2 Advanced rollback]. The TSF and the IT environment cooperatively protect the TSF from interference and tampering by providing the security functions with a separate execution domain [FPT_SEP_DB.1 Partial TSF domain separation and FPT_SEP_OS.1 Environment TSF domain separation].

O.TOE_USER_IDENTIFICATION*

To uniquely identify DBMS users.

The TSF uses a defined set of security attributes associated with DBMS users [FIA_ATD.1 User attribute definition, FMT_SMR.1 Security roles]. The TSF associates security attributes with DBMS subjects acting on behalf of that user [FIA_USB.1 User subject binding].

7.2.1.2 Security Objectives for the IT Environment

This section presents the rationale that the security requirements for the IT environment meet the security objectives for the IT environment.

O.IT_ENV_SELF_PROTECTION*

To protect the host operating system and its assets from external interference, tampering, or unauthorized disclosure.

The host OS prevents interference and tampering by maintaining separate domains of execution for itself and its subjects [FPT_SEP_OS.1 Environment TSF domain separation] and by ensuring that its security policy enforcing functions cannot be bypassed [FPT_RVM_OS.1 Environment non-bypassability of the TSP]. In addition, the host OS protects DBMS objects and authentication information transmitted between physically separated parts of the TOE [FDP_ITT.1 Basic internal transfer protection, FPT_ITT.1 Basic internal TSF data transfer protection]. The host OS ensures there is no external tampering and its operations are correct by providing TSFs for recovery from failure [FPT_RCV. 2-NIAP-0406 Recovery from Failure] and for running a suite of tests during start-up and at the request of an authorized user [FPT_AMT.1 Abstract machine testing]. The host OS also provides the ability to test and ensure the correct operation of the SF [FPT_TST.1 TSF testing].

O.IT_ENV_TIME

To provide reliable time stamps.

The host OS provides the DBMS TSF with reliable time stamps [FPT_STM.1 Reliable time stamps].

O.IT_ENV_ AUDIT_PROTECTION*

To provide the capability to protect audit information associated with individual users.

The audit records are stored in audit files, which are being treated as user data files by the host OS. The host OS protects user data by providing TSFs for identifying and authenticating users [FIA_UID.2 User identification before any action, FIA_UAU.2 User authentication before any action, FIA_ATD.1 User attribute definition, FIA_USB.1 User-subject binding]. The host OS protects the audit files from unauthorized access [FDP_ACC.1 Access control policy, FDP_ACF.1 Access control functions, FDP_ITT.1 Internal TOE transfer, FDP_RIP_OS.2 Environment full residual information protection]. The host OS protects the audit information from unauthorized access by provides various user roles for separation of authorization and duty [FMT_SMR.1 Security roles].

O. IT_ENV_USER_AUTHENTICATION*

To verify the claimed identity of a DBMS user.

In order to use the DBMS, a DBMS user identifies himself or herself to the host OS TSF [O.IT_ENV_USER_IDENTIFICATION*]. The DBMS user then provides authentication information to the TSF of the host OS providing authentication functions. The host OS TSF verifies the user's identity [FIA_UAU.2 User authentication before any action]. The host OS TSF provides DBMS users, database administrators, and DBMS security administrators [FMT_SMR.1 Security Roles] with the capabilities to manage authentication data [FMT_MTD.1 Management of TSF data, O.ENV_USERS, O.ENV_ADMIN, O.ENV_CONFIG]. The host OS authentication functions are supported by the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*].

O. IT_ENV_USER_ IDENTIFICATION*

To uniquely identify DBMS users.

The TSF of the host OS and the TOE use a defined set of security attributes associated with DBMS users [FIA_ATD.1 User attribute definition]. In order to use the DBMS, a DBMS user identifies himself or herself to the TSF of the host OS [FIA_UID.2 User identification before any action]. Once the host OS verifies the user's identity [O.IT_ENV_USER_AUTHENTICATION*], the host OS TSF associates security attributes with DBMS subjects acting on behalf of that user [FIA_USB.1: User subject binding]. The IT environment protects communication between the DBMS user and the host OS TSF during the identification process [O.ENV_SECURE_COMMS]. The TSF of the host OS provides OS security administrators [FMT_SMR.1 Security Roles] with the capabilities to manage identification data [FMT_MTD.1: Management of TSF data], manage security attributes [FMT_MSA.1 Management of security attributes], and identification functions [O.ENV_ADMIN, O.ENV_CONFIG]. The host OS identification functions are supported by the host OS in the IT environment [O.ENV_TRUST_IT, O.IT_ENV_SELF_PROTECTION*].

7.2.2 Security Assurance Requirements Rationale

The selection of EAL4 is based on guidance from the *Information Assurance Technical Framework* ([5] section 4.5.2) on determining the degree of robustness. EAL4 was selected because of the value of the information assets and the presumed threat agents. Violation of the information protection policy would cause serious damage to the security, safety, financial posture, or infrastructure of the organization. The most capable threat agents are presumed to be sophisticated adversaries with moderate resources who are willing to take little risk, e.g., organized crime or sophisticated hackers.

O.TOE_ADMIN_GUIDANCE

To provide database administrators, database operators, DBMS security administrators, database administrators, and DBMS audit administrators with the guidance documentation necessary for secure management of the DBMS.

The developer documents the procedures used for delivering the TOE, which ensure that any modifications made to the TOE in transit would be detected [ADO_DEL.2 Detection of modification]. In addition, the developer provides procedures for installing, generating, and starting up the TOE in a secure manner [ADO_IGS.1 Installation, generation, and start-up procedures] and the evaluator will have confirmed that the guidance is complete, clear, consistent and reasonable [AVA_MSU.2, Validation of Analysis]. Database administrators and DBMS security administrators are adequately trained [O.ENV_ADMIN] to use the documentation [AGD_ADM.1 Administrator guidance] and follow any applicable security policies and procedures [O.ENV_CONFIG].

O.TOE_DEVEL_CM*

To track and control all changes to the DBMS and its development evidence during development.

The developer controls the development and maintenance of the TOE through a life-cycle model [ALC_LCD.1 Developer defined life-cycle model] and adequately protects the confidentiality and integrity of the TOE during its development and maintenance [ALC_DVS.1 Identification of security measures]. The developer uses a CM system, whose use is described in a CM plan. The CM system includes automated CM tools to control the implementation representation and generation of the DBMS TOE [ACM_AUT.1 Partial CM automation]. The CM system ensures the integrity of the TOE from its design stages through TOE generation [ACM_CAP.4 Generation support and acceptance procedures]. The CM system tracks the TOE representation implementation, development evidence, documentation, and security flaws [ACM_SCP.2 Problem tracking CM coverage]. In addition, the developer identifies the development tools used to produce the TOE [ALC_TAT.1 Well-defined development tools].

O.TOE_SOUND_DEVELOPMENT*

To apply, and accurately document, sound design and implementation principles and techniques to the development of the DBMS.

Through step-wise refinement, the developer creates representations of the TSF with successive TSF representations providing finer-grained details [ADV_FSP.2 Fully defined external interfaces, ADV_HLD.2 Security enforcing high-level design, ADV_LLD.1 Descriptive low-level design, ADV_IMP.1 Subset of the implementation of the TSF]. Moreover, the developer demonstrates that the most detailed TSF representation is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST [ADV_RCR.1 Informal correspondence demonstration].

In addition, the design documentation includes a security policy model, strength of function analysis, and vulnerability analysis. The security policy model describes the rules and characteristics of TOE security policies and demonstrates that the model corresponds to TOE security functions [ADV_SPM.1 Informal TOE security policy model]. The strength of function analysis covers TOE security functions realized by probabilistic or permutational mechanisms identified in the ST. This analysis demonstrates that each such mechanism satisfies the corresponding strength of function claim in the ST [AVA_SOF.1 Strength of TOE security function evaluation]. The developer analyzes the TOE and its design for security vulnerabilities, shows that all identified vulnerabilities cannot be exploited, and justifies that the TOE is resistant to obvious penetration attacks. Evaluators perform an independent vulnerability analysis, including penetration testing to show that the security functions are not bypassable [AVA_VLA.2 Independent vulnerability analysis, O.TOE_TESTING*].

The developer provides guidance documentation for the TOE that is complete, clear, consistent and reasonable together with an analysis that the guidance documentation is complete [AVA_MSU.2 Validation of analysis].

Finally, the developer tracks and controls all changes to the TOE representation implementation, development evidence, and documentation during its development including changes to the design [O.TOE_DEVEL_CM*].

O.TOE_TESTING*

To independently test the DBMS security functions both for conformance to the DBMS design and for vulnerabilities.

The developer tests the TSF against its design in a systematic manner and provides an analysis of the test coverage [ATE_COV.2 Analysis of coverage]. The developer performs tests to demonstrate that all security functions perform as specified and provides supporting test documentation [ATE_FUN.1 Functional testing]. The testing of the TSF is done to the level of detail contained in the high-level design [ATE_DPT.1 Testing: high-level design, O.TOE_SOUND_DEVELOPMENT*].

Evaluators independently verify developer testing by executing a sample of the developer’s tests [ATE_IND.2 Independent testing – sample]. Furthermore, the evaluators use the development evidence provided by the developer in their analysis and penetration testing [O.TOE_SOUND_DEVELOPMENT*]. The evaluators’ analysis includes an independent vulnerability analysis.

O.TOE_USER_GUIDANCE

To provide authorized DBMS users with the guidance documentation necessary for secure use of the DBMS.

The developer provides documentation for the secure use of the TOE [AGD_USR.1 User guidance]. Authorized DBMS users are adequately trained [O.ENV_USERS] to use this documentation.

7.2.3 Requirements to Security Objectives Tracings

This section contains tables showing the tracings between security requirements and the security objectives identified in section 4 Security Objectives.

Table 14 identifies, for each TOE security objective, the security functional requirements and security assurance requirements that meet that objective.

Table 14 Security Requirements to TOE Security Objectives Tracings

Objective	Requirements
O.TOE_ADMIN_GUIDANCE	ADO_DEL.2 Detection of modification ADO_IGS.1 Installation, generation, and start-up procedures AGD_ADM.1 Administrator guidance AVA_MSU.2 User guidance
O.TOE_ADMIN_ROLE	FMT_SMR.1 Security roles FMT_SMF.1 Specification of Management Functions FDP_ACC.2 Complete access control FDP_ACF.1 Security attribute based access control
O.TOE_AUDIT_GENERATION	FAU_GEN.1 Audit data generation FAU_GEN.2 User identity association FIA_ATD.1 User attribute definition FIA_USB.1 User-subject binding FAU_SEL.1 Selective audit FMT_SMR.1 Security roles FMT_MOF.1 Management of security functions behavior FMT_MTD.1 Management of TSF data FMT_SMF.1 Specification of Management Functions

Objective	Requirements
O.TOE_AUDIT_PROTECTION	FAU_SAR.2 Restricted audit review FAU_STG.1 Protect audit trial storage FAU_STG.4 Prevention of audit data loss
O.TOE_AUDIT_REVIEW	FAU_SAR.1 Audit review FAU_SAR.3 Selectable audit review FMT_SMR.1 Security Roles
O.TOE_DEVEL_CM	ACM_AUT.1 Partial CM automation ACM_CAP.4 Generation support and acceptance procedures ACM_SCP.2 Problem tracking CM coverage ALC_DVS.1 Identification of security measures ALC_LCD.1 Developer defined life-cycle model ALC_TAT.1 Well-defined development tools
O.TOE_DISCRETIONARY_ACCESS	FDP_ACC.2 Complete access control FDP_ACF.1 Security attribute based access control FDP_ITC.1 Import of user data without security attributes FDP_ITC.2 Import of user data with security attributes FDP_ETC.1 Export of user data without security attributes FDP_ETC.2 Export of user data with security attributes FMT_MSA.3 Static Attribute initialization FMT_MSA.1 Management of security attributes FMT_MSA.2 Secure security attributes FMT_SMR.1 Security roles FMT_REV.1 Revocation

Objective	Requirements
O.TOE_MANDATORY_ACCESS	FDP_IFC.2 Complete information flow control FDP_IFF.2 Hierarchical security attributes FDP_ETC.1 Export of user data without security attributes FDP_ETC.2 Export of user data with security attributes FDP_ITC.1 Import of user data without security attributes FDP_ITC.2 Import of user data with security attributes FMT_MSA.1 Management of security attributes FMT_MSA.3 Static attribute initialization FMT_SMR.1 Security roles
O.TOE_RECOVERY	FPT_RCV.4 Function recovery FDP_ROL.2 Advanced rollback
O.TOE_RESIDUAL_INFORMATION	FDP_RIP_DB.2 TOE full residual information protection FPT_RCV.4 Function recovery
O.TOE_SELF_PROTECTION	FPT_RVM_DB.1 TOE non-bypassability of the TSP FPT_SEP_DB.1 Partial TSF domain separation FDP_ROL.2 Advanced rollback
O.TOE_SOUND_DEVELOPMENT	AVA_MSU.2 Validation of analysis AVA_SOF.1 Strength of TOE security function evaluation AVA_VLA.2 Independent vulnerability analysis ADV_FSP.2 Fully defined external interfaces ADV_HLD.2 Security-enforcing high-level design ADV_IMP.1 Subset of the implementation of the TSF ADV_LLD.1 Descriptive low-level design ADV_RCR.1 Informal correspondence demonstration ADV_SPM.1 Informal TOE security policy model

Objective	Requirements
O.TOE_TESTING	ATE_COV.2 Analysis of coverage ATE_DPT.1 Testing: high-level design ATE_FUN.1 Functional testing ATE_IND.2 Independent testing - sample
O.TOE_USER_IDENTIFICATION	FIA_ATD.1 User attribute definition FIA_USB.1 User-subject binding FMT_SMR.1 Security roles
O.TOE_USER_GUIDANCE	AGD_USR.1 User guidance

Table 15 identifies, for each objective for the IT environment, the security functional requirements that meet that objective.

Table 15 Functional Requirements to IT Environment Security Objectives Tracings

Objective	Requirements
O.IT_ENV_AUDIT_PROTECTION	FIA_UID.2 User identification before any action FIA_UAU.2 User authentication before any action FIA_ATD.1 User attribute definition FIA_USB.1 User-subject binding FDP_ACC.1 Access control policy FDP_ACF.1 Access control functions FDP_ITT.1 Internal TOE transfer FDP_RIP_OS.2 Environment full residual information protection FMT_SMR.1 Security roles
O. IT_ENV_USER_AUTHENTICATION	FIA_UAU.2 User authentication before any action FMT_SMR.1 Security roles FMT_MTD.1 Management of TSF data

Objective	Requirements
O. IT_ENV_USER_IDENTIFICATION	FIA_ATD.1 User attribute definition FIA_UID.2 User identification before any action FIA_USB.1 User-subject binding FMT_SMR.1 Security roles FMT_MTD.1 Management of TSF data FMT_MSA.1 Management of security attributes
O.IT_ENV_SELF_PROTECTION	FPT_RVM_OS.1 Environment non-bypassability of the TSP FPT_SEP_OS.1 Environment TSF domain separation FDP_ITT.1 Basic internal transfer protection FPT_ITT.1 TSF domain separation FPT_RCV.2-NIAP-0406 Recovery from Failure FPT_AMT.1 Abstract machine testing FPT_TST.1 TSF testing
O.IT_ENV_TIME	FPT_STM.1 Reliable time stamps

Table 16 identifies, for each security requirement, the security objectives that are met by that requirement.

Table 16 Security Requirement to Security Objectives mapping

Requirements	Objectives
Security Functional Requirements	
FAU_GEN.1	O.TOE_AUDIT_GENERATION
FAU_GEN.2	O.TOE_AUDIT_GENERATION
FAU_SAR.1	O. TOE_AUDIT_REVIEW
FAU_SAR.2	O. TOE_AUDIT_PROTECTION
FAU_SAR.3	O. TOE_AUDIT_REVIEW
FAU_SEL.1	O.TOE_AUDIT_GENERATION
FAU_STG.1	O.TOE_AUDIT_PROTECTION
FAU_STG.4	O.TOE_AUDIT_PROTECTION
FDP_ACC.2	O.TOE_DISCRETIONARY_ACCESS

Requirements	Objectives
	O.TOE_ADMIN_ROLE
FDP_ACF.1	O.TOE_DISCRETIONARY_ACCESS O.TOE_ADMIN_ROLE
FDP_ETC.1	O.TOE_DISCRETIONARY_ACCESS O.TOE_MANDATORY_ACCESS
FDP_ETC.2	O.TOE_MANDATORY_ACCESS O.TOE_DISCRETIONARY_ACCESS
FDP_IFC.2	O.TOE_MANDATORY_ACCESS
FDP_IFF.2	O.TOE_MANDATORY_ACCESS
FDP_ITC.1	O.TOE_MANDATORY_ACCESS O.TOE_DISCRETIONARY_ACCESS
FDP_ITC.2	O.TOE_MANDATORY_ACCESS O.TOE_DISCRETIONARY_ACCESS
FDP_RIP_DB.2	O.TOE_RESIDUAL_INFORMATION
FDP_ROL.2	O.TOE_RECOVERY* O.TOE_SELF_PROTECTION*
FIA_ATD.1	O.TOE_AUDIT_GENERATION O.TOE_USER_IDENTIFICATION
FIA_USB.1	O.TOE_AUDIT_GENERATION O.TOE_USER_IDENTIFICATION
FMT_MOF.1	O.TOE_AUDIT_GENERATION
FMT_MSA.1	O.TOE_MANDATORY_ACCESS O.TOE_DISCRETIONARY_ACCESS
FMT_MSA.2	O.TOE_DISCRETIONARY_ACCESS
FMT_MSA.3	O.TOE_DISCRETIONARY_ACCESS O.TOE_MANDATORY_ACCESS
FMT_MTD.1	O.TOE_AUDIT_GENERATION
FMT_SMR.1	O.TOE_DISCRETIONARY_ACCESS O.TOE_MANDATORY_ACCESS O.TOE_USER_IDENTIFICATION O.TOE_AUDIT_GENERATION O.TOE_AUDIT_REVIEW

Requirements	Objectives
	O.TOE_ADMIN_ROLE
FMT_SMF.1	O.TOE_ADMIN_ROLE O.TOE_AUDIT_GENERATION
FPT_RCV.4	O.TOE_RECOVERY O.TOE_RESIDUAL_INFORMATION
FMT_REV.1	O.TOE_DISCRETIONARY_ACCESS
FPT_RVM_DB.1	O.TOE_SELF_PROTECTION
FPT_SEP_DB.1	O.TOE_SELF_PROTECTION
Security Assurance Requirements	
ACM_AUT.1	O.TOE_DEVEL_CM
ACM_CAP.4	O.TOE_DEVEL_CM
ACM_SCP.2	O.TOE_DEVEL_CM
ADO_DEL.2	O.TOE_ADMIN_GUIDANCE
ADO_IGS.1	O.TOE_ADMIN_GUIDANCE
ADV_FSP.2	O.TOE_SOUND_DEVELOPMENT
ADV_HLD.2	O.TOE_SOUND_DEVELOPMENT
ADV_IMP.1	O.TOE_SOUND_DEVELOPMENT
ADV_LLD.1	O.TOE_SOUND_DEVELOPMENT
ADV_RCR.1	O.TOE_SOUND_DEVELOPMENT
ADV_SPM.1	O.TOE_SOUND_DEVELOPMENT
AGD_ADM.1	O.TOE_ADMIN_GUIDANCE
AGD_USR.1	O.TOE_USER_GUIDANCE
ALC_DVS.1	O.TOE_DEVEL_CM
ALC_LCD.1	O.TOE_DEVEL_CM
ALC_TAT.1	O.TOE_DEVEL_CM
ATE_COV.2	O.TOE_TESTING
ATE_DPT.1	O.TOE_TESTING
ATE_FUN.1	O.TOE_TESTING
ATE_IND.2	O.TOE_TESTING

Requirements	Objectives
AVA_MSU.2	O.TOE_SOUND_DEVELOPMENT O.TOE_ADMIN_GUIDANCE
AVA_SOF.1	O.TOE_SOUND_DEVELOPMENT
AVA_VLA.2	O.TOE_SOUND_DEVELOPMENT

7.2.4 Dependency Rationale

7.2.4.1 Justification of Unsupported Dependencies

(A) Dependencies of FAU_GEN.1

The FAU_GEN.1 has a dependency of FPT_STM.1. This dependency is levied on the host OS, instead of on the TOE. The host OS provides the TOE with a reliable timestamp upon request from the TOE.

(B) Dependencies of FDP_ITC.2

The unsupported dependencies of FDP_ITC.2 are:

- [FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path], and
- FPT_TDC.1 Inter-TSF basic TSF data consistency

The TOE SF does not communicate or share data with another trusted IT product in the IT environment for importing user data. Therefore, the TOE FDP_ITC.2 security functional requirement does not have any dependency requirement on FTP_ITC.1, FTP_TRP.1, or FPT_TDC.1, which is required while the TOE communicates or shares user or TSF data with another trusted IT product.

Dependency on FTP_ITC.1

The TOE imports user data only from a previously created export file or dump/backup file on the host OS by the DBMS roles defined in the FMT_SMR.1. While the host OS is not part of the TOE, both are collocated on the same hardware platform in a physically secure environment. Hence, a trusted channel between the TOE and the host OS is unnecessary.

Dependency on FPT_TDC.1

The TOE uses host OS input devices (e.g., tape drive, disk drive) for importing user data with security attributes. The data (user data and sensitivity labels) are read from the device through host OS system calls and are treated as input data of an application by the OS. Hence, the host OS does not interpret the input data and the requirement for consistent data interpretation is unnecessary.

(C) Dependencies of FMT_SMR.1

The SFR FMT_SMR.1 has a dependency of FIA_UID.1. This dependency is levied on the host OS, instead of on the TOE. The host OS treats the TOE as an application process acting on the behalf of the logged on user. The TOE views the host OS as a trusted IT product in the TOE environment, which provides the user identification and authentication and other security functions.

(D) Dependencies of FPT_RCV.4

The dependency of FPT_RCV.4 is ADV_SPM.1. This dependency is levied on the host OS, instead of on the TOE. The TOE views the host OS as a trusted IT product in the TOE environment, which provides security function to provide the reliable time stamps upon the TOE's request.

7.2.4.2 Security Requirements Dependencies

Table 17 identifies, for each TOE security requirement, the security functional and assurance dependencies.

Table 17 Security Functional and Assurance Requirements Dependencies

Requirement	Dependencies
Functional Requirements	
FAU_GEN.1	FPT_STM.1 (see Dependency Rationale)
FAU_GEN.2	FAU_GEN.1, FIA_UID.1
FAU_SAR.1	FAU_GEN.1
FAU_SAR.2	FAU_SAR.1
FAU_SAR.3	FAU_SAR.1
FAU_SEL.1	FAU_GEN.1, FMT_MTD.1
FAU_STG.1	FAU_GEN.1
FAU_STG.4	FAU_STG.1
FDP_ACC.2	FDP_ACF.1
FDP_ACF.1	FDP_ACC.1, FMT_MSA.3
FDP_ETC.1	FDP_ACC.1, FDP_IFC.1
FDP_ETC.2	FDP_ACC.1, FDP_IFC.1
FDP_IFC.2	FDP_IFF.2
FDP_IFF.2	FDP_IFC.1, FMT_MSA.3
FDP_ITC.1	FDP_ACC.1, FDP_IFC.1, FMT_MSA.3
FDP_ITC.2	FDP_ACC.1, FDP_IFC.1 FPT_TDC.1 (see Dependency Rationale)

Requirement	Dependencies
FDP_RIP_DB.2	None
FDP_RIP_OS.2	None
FDP_ROL.2	None
FIA_ATD.1	None
FIA_USB.1	FIA_ATD.1
FMT_MOF.1	FMT_SMR.1, FMT_SMF.1
FMT_MSA.1	FDP_ACC.1, FDP_IFC.1, FMT_SMR.1, FMT_SMF.1
FMT_MSA.2	ADV_SPM.1, FDP_ACC.1, FDP_IFC.1, FMT_MSA.1 FMT_SMR.1
FMT_MSA.3	FMT_MSA.1, FMT_SMR.1
FMT_MTD.1	FMT_SMR.1, FMT_SMF.1
FMT_REV.1	FMT_SMR.1
FMT_SMF.1	None
FMT_SMR.1	FIA_UID.1 (see Dependency Rationale)
FPT_RCV.4	ADV_SPM.1 (see Dependency Rationale)
FPT_RVM_DB.1	None
FPT_RVM_OS.1	None
FPT_SEP_DB.1	None
FPT_SEP_OS.1	None
Assurance Requirements	
ACM_AUT.1	ACM_CAP.3
ACM_CAP.4	ACM_SCP.1, ALC_DVS.1
ACM_SCP.2	ACM_CAP.3
ADO_DEL.2	ACM_CAP.3
ADO_IGS.1	AGD_ADM.1
ADV_FSP.2	ADV_RCR.1
ADV_HLD.2	ADV_FSP.1, ADV_RCR.1
ADV_IMP.1	ADV_LLD.1, ADV_RCR.1, ALC_TAT.1
ADV_LLD.1	ADV_HLD.2, ADV_RCR.1

Requirement	Dependencies
ADV_SPM.1	ADV_FSP.1
AGD_ADM.1	ADV_FSP.1
AGD_USR.1	ADV_FSP.1
ALC_TAT.1	ADV_IMP.1
ATE_COV.2	ADV_FSP.1, ATE_FUN.1
ATE_DPT.1	ADV_HLD.1, ATE_FUN.1
ATE_IND.2	ADV_FSP.1, AGD_ADM.1, AGD_USR.1, ATE_FUN.1
AVA_MSU.2	ADO_IGS.1, ADV_FSP.1, AGD_ADM.1, AGD_USR.1
AVA_SOF.1	ADV_FSP.1, ADV_HLD.1
AVA_VLA.2	ADV_FSP.1, ADV_HLD.2, ADV_IMP.1, ADV_LLD.1, AGD_ADM.1, AGD_USR.1

7.2.5 Strength of Function Rationale

The evaluated TOE is intended to operate in DoD medium robustness environments processing a restricted range of DoD classified information. The TOE is intended to counter attacks by attackers with moderate attack potential. Hence, the SOF-medium is appropriate for the TOE.

7.3 Rationale for TOE Summary Specification

7.3.1 Rationale for TOE Security Functions

This section demonstrates that the TOE security functions completely and accurately meet the TOE security functional requirements. Table 18 provides the mapping between them. Table 18 is followed by the rationale for each of the security functional requirements addressing how the requirements are being met.

Table 18 Mapping of TOE Security Functions to TOE Security Functional Requirements

TOE Security Functions		TOE Security Requirement
Function Name	Security Function Full Name	
AU.RGEN	Audit Record Generation	FAU_GEN.1 FAU_GEN.2

TOE Security Functions		**TOE Security Requirement**
Function Name	**Security Function Full Name**	
AU.SELT	Audit Selection from Auditable Events	FMT_SMF.1 FAU_SEL.1
AU.ANLY	Audit Selective Review and Analysis	FMT_SMF.1 FAU_SAR.1 FAU_SAR.2 FAU_SAR.3
AU.RLOG	Audit Record Logging	FAU_STG.1 FAU_STG.4
IA.UATD	User Security Attribute Definitions	FIA_ATD.1
IA.USBD	User-subject binding	FIA_USB.1
DP.DACL	Discretionary access control list	FDP_ACC.2 FDP_ACF.1
DP.DACM	Discretionary access control enforcement	FDP_ACC.2 FDP_ACF.1
DP.MACL	Subject and object MAC label	FDP_IFC.2 FDP_IFF.2 FMT_MSA.1 FMT_MSA.3
DP.MACM	Mandatory access control enforcement	FDP_IFC.2 FDP_IFF.2 FMT_MSA.1 FMT_MSA.3

TOE Security Functions		TOE Security Requirement
Function Name	**Security Function Full Name**	

DP.EXPT	Export of user data	FMT_SMF.1
		FDP_ETC.1
		FDP_ETC.2
		FDP_ACC.2.1(1)
		FDP_ACC.2.1(3)
		FDP_ACC.2.2(3)
		FDP_ACC.2.1(4)
		FDP_ACC.2.2(4)
		FDP_ACF.1.2(3)
		FDP_ACF.1.3(3)
		FDP_ACF.1.4(3)
		FDP_ACF.1.2(4)
		FDP_ACF.1.3(4)
		FDP_ACF.1.4(4)
		FDP_ACF.1.2(5)
		FDP_ACF.1.3(5)
		FDP_ACF.1.4(5)
		FDP_IFC.2
		FDP_IFF.2

TOE Security Functions		TOE Security Requirement
Function Name	**Security Function Full Name**	

DP.IMPT	Import of user data	FMT_SMF.1
		FDP_ITC.1
		FDP_ITC.2
		FDP_ACC.2.1(3)
		FDP_ACC.2.2(3)
		FDP_ACC.2.1(4)
		FDP_ACC.2.2(4)
		FDP_ACF.1.2(3)
		FDP_ACF.1.3(3)
		FDP_ACF.1.4(3)
		FDP_ACF.1.2(4)
		FDP_ACF.1.3(4)
		FDP_ACF.1.4(4)
		FDP_ACF.1(3,4)
		FDP_IFC.2
		FDP_IFF.2
		FMT_SMR.1

DP.URIP	Residual Information Protection	FDP_RIP_DB.2
MT.AUDM	Audit Administrator Management	FAU_SEL.1 FMT_MTD.1 FMT_SMR.1 FMT_SMF.1 FMT_MOF.1(1)
MT.DBAM	Database Administrator Management	FDP_ACF.1 FMT_MSA.1(1) FMT_MSA.1(3) FMT_REV.1 FMT_SMR.1 FMT_SMF.1 FMT_MOF.1(3)

TOE Security Functions		TOE Security Requirement
Function Name	**Security Function Full Name**	
MT.OPRM	Database Operator Management	FPT_RCV.4 FMT_SMR.1 FMT_SMF.1
MT.SADM	Security Administrator Management	FPT_RCV.4 FDP_ITC.2 FDP_ROL.2 FMT_MSA.1(2) FMT_SMR.1 FMT_SMF.1 FMT_MOF.1(2)
MT.USRM	Non-admin. DBMS User Management	FDP_ETC.1 FDP_ETC.2 FDP_ITC.1 FDP_ITC.2 FDP_ROL.2 FMT_MSA.1(1) FMT_MSA.1(3) FMT_REV.1 FMT_SMR.1
MT.ROLE	System role assignment to users	FMT_SMR.1
MT.MSA	Security Attribute Protection	FMT_MSA.1, FMT_MSA.2, FMT_MSA.3
PT.TRCV	Trusted backup, restore, recovery	FPT_RCV.4 FMT_SMF.1
PT.RFMN	Reference Monitor	FPT_RVM_DB.1
PT.SEPT	Domain separation	FPT_SEP_DB.1

FAU_GEN.1 Audit data generation

Trusted RUBIX Audit Record Generation (AU.RGEN) function generates all the audit events listed in Table 3 FAU_GEN.1 Auditable Events. Within each audit record is the common information listed in Table 5, which includes event timestamp for date and time, event ID for event type, user ID, group ID, process ID for subject identity, and status for the outcome of the event. The event specific audit information saved in audit records is listed in Table 6. The information includes event name, object level, subject level, subject user ID, transaction timestamp, database name and other container names of the target DBMS object. For auditable events of the trusted administrative programs, the administrator action as well as any security-relevant attribute changes are audited by Trusted RUBIX.

FAU_GEN.2 User Identity Association

The Trusted RUBIX Audit Record Generation (AU.RGEN) function ensures each audit record has the common information, which includes user ID, group ID, session ID, subject sensitivity label, and object name associated with the audited event.

FAU_SAR.1 Audit review

The Trusted RUBIX Audit Selective Review and Analysis (AU.ANLY) function provides the DBMS audit administrators with the *rxauditrpt* utility that converts the audit trail into a human readable report so that he may read the entire contents of the audit trail.

FAU_SAR.2 Restricted audit review

The Trusted RUBIX audit files are stored at the system high sensitivity label and are owned by the special user NOBODY and group RUBIXTP. The discretionary access control security function of the operating system ensures that by default, audit files are owned by the special user NOBODY and group RUBIXTP and set to allow read-write access for the group, null access for user and world. The DBMS audit administrator role is responsible for administering the Trusted RUBIX audit subsystem. The role membership is explicitly checked through its associated authorization by *rxauditrpt*. The *rxauditrpt* program executes with the set GID bit set to the RUBIXTP on its execution file. Only the Trusted RUBIX executables may use the RUBIXTP group. The Trusted RUBIX *rxauditrpt* utility, therefore, can only be run on the server by those in the role of DBMS audit administrator to examine the recorded audit data.

FAU_SAR.3 Selectable audit review

Using *rxauditrpt* utility in the audit function, Trusted RUBIX permits the DBMS audit administrator to perform searches of the audit data based on event status (success/failure) and database name.

FAU_SEL.1 Selective audit

Trusted RUBIX *rxauditset* utility permits the DBMS audit administrator to select which audit events are to be recorded in the audit trail from the auditable events based on system-wide basis, specific event type, object identity, user identity, subject sensitivity label, objects with a specific sensitivity label, and objects within a range of sensitivity labels.

FAU_STG.1 Protected audit trail storage

The Trusted RUBIX server creates the audit file as needed with its sensitivity label set to system high, its owner/owning group set to NOBODY/RUBIXTP, and with read/write permissions set for the RUBIXTP group only. The host OS protects the audit files as OS file objects by the Discretional Access Control of file objects. The Trusted RUBIX command, *`rxlogs`*, restricts the use of the command to list the files and to delete audit files to DBMS Audit Administrators.

FAU_STG.4 Prevention of audit data loss

Trusted RUBIX automatically terminates the Trusted RUBIX server process upon detection of storage failure on newly generated audit records for prevention of audit data loss.

FDP_ACC.2 Complete Access Control

Trusted RUBIX enforces the Discretionary Access Control (DAC) Policy and restricts access to the DBMS named objects based on the identity of the subjects and/or groups to which they belong. Trusted RUBIX relies on Trusted Solaris 8 to enforce the Discretionary Access Control (DAC) Policy and to restrict access to the operating system named objects, such as directories and files, based on the identity of the subjects and/or groups to which they belong.

Trusted RUBIX Discretionary Access Control (DAC) Security Function is integrated with the underlying Trusted Solaris 8 kernel to form a single DAC Policy. Within Trusted RUBIX, all DAC protected data are stored in single-level operating system file objects. Each database is stored in multiple OS files. All OS files for a unique database are stored in a single OS directory. The directories for all databases are stored in a single `databases` directory. The DAC function of Trusted Solaris 8 supports a DAC policy between processes and the protected database directories and files objects. The Trusted RUBIX DAC security function enforces a DAC policy between processes acing on behalf of users and the following objects: database, catalog, schema, table, view, and column. The DAC security functions ensure the DAC policy is enforced on all operations among subjects and objects.

FDP_ACF.1 Access Control Functions

The named objects protected by Trusted RUBIX DAC are databases, catalogs, schemas, tables, views, and columns. Each Trusted RUBIX named object is associated with an *access control list* (ACL). The ACL information consists of a bitmap of privileges and a bitmap of the privileges associated WITH GRANT options. Each bitmap contains a single bit for each privilege on each object type.

The Discretionary Access Control Security Function enforces access restriction to Trusted RUBIX object based on user identity and group identity associated with a subject, and Access Control Lists associated with an object. The Discretionary Access Control List sub-function (DP.DACL) sets the default access privileges in the object's ACL when the object is created. The default access privileges give the creator all access privileges to propagate and/or modify the access privileges in the ACL. The Discretionary Access Control Enforcement Mechanism (DP.DACM) identifies all the valid operations on each type of Trusted RUBIX objects and enforces the various access control rules governing each valid operation to different types of objects, which include database, catalog, schema, table, view, and column.

FDP_ETC.1 Export of user data without security attributes

Authorized DBMS users can use the `rxexport` command to export user data without security attributes. All MAC and DAC rules apply to `rxexport` as if the user were selecting the table using an SQL statement.

FDP_ETC.2 Export of user data with security attributes

Authorized DBMS users can use the `rxexport` command to export user data with security attributes. All MAC and DAC rules apply to `rxexport` as if the user were selecting the table using an SQL statement. Authorized DBMS users may specify an option to list the data with the row sensitivity label as the first column, so that the sensitivity labels are unambiguously associated with the exported row data.

FDP_IFC.2 Complete information flow control and

FDP_IFF.2 Hierarchical security attributes

The MAC security function ensures that the MAC policy is applied on all operations between processes and the following Trusted RUBIX objects: database, catalog, schema, index, table, view, and row. The MAC policy is also applied on the sensitivity label of the resultant combined information (from multiple pieces of information) and extracted information (from another piece of information).

The MAC security function enforces the MAC policy between subjects and objects. Trusted RUBIX assigns mandatory sensitivity labels to all subjects and objects under its control.

The mandatory access control policy enforced by Trusted RUBIX is based on Bell and LaPadula's sensitivity policy. The sensitivity policy is based on a dominance relationship between the subject and the object. For any two MAC labels A and B, the label A dominates the label B if and only if the hierarchical classification of label A is greater than or equal to that of label B and the non-hierarchical categories of label A include all those of label B as a subset. The sensitivity policy prevents a subject from reading information that is too sensitive for the subject's clearance. Trusted RUBIX requires a subject to write a piece of information at a sensitivity label equal to the session sensitivity label (no write up policy).

A mandatory sensitivity label in Trusted RUBIX contains a hierarchical classification and a set of non-hierarchical categories. This mandatory sensitivity label associated with subjects and objects form the basis of the Trusted RUBIX MAC policy. The Subject and Object MAC Label function (DP.MACL) ensures that the newly created Trusted RUBIX objects are properly labeled and attached based on the subject's session sensitivity label. The DP.MACL function meets the following SFRs: FDP_IFF.2.1, FDP_IFF.2.2.(b), and FDP_IFF.2.3.(a), (b), and (c).

The mandatory access control enforcement function (DP.MACM) ensures that the subject's session has the required sensitivity label for all attempted database operations on the Trusted RUBIX objects. The MAC rules in Table 7 of DP.MACM meet the following SFRs: FDP_IFF.2.2 (a) and FDP_IFF.2.2 (c).

Trusted RUBIX allows the use of multilevel database import (*rximport* with *–m* option) and reclassification (*rxrecl*) trusted utilities by the Trusted RUBIX Security Administrator. These trusted utilities operate on multilevel data and cause information with multiple sensitivity labels to flow to and from the subject acting as the Security Administrator. These implementations meet the FDP_IFF.2.5 SFR.

Trusted RUBIX allows the use of Standard SQL syntax to perform label comparisons. The label comparison functions supported by Trusted RUBIX meet the FDP_IFF.2.7 SFR.

FDP_ITC.1 Import of user data without security attributes

The trusted *rximport* utility supports importing unlabeled user data. All user data imported into the targeted database table via *rximport* utility without the use of "*-m*" option, are labeled as the subject session sensitivity label. The discretionary access control (DAC) for restricting user executing the *rximport* utility is enforced by the underlying OS and Trusted RUBIX. All MAC and DAC rules apply to *rxuimport* as if the user were inserting into the table or view using an SQL statement. Trusted RUBIX requires the following MAC rules: (1) The Trusted Solaris 8 session sensitivity label of the invoker must dominate the object label of the Trusted Solaris 8 input files, and (2) The Trusted RUBIC session sensitivity label of the user must dominate the sensitivity label of the targeted database, catalog, schema, and table containing objects for import.

FDP_ITC.2 Import of user data with security attributes

The trusted *rximport* utility supports importing labeled user data. Only the Security Administrator (SA) can load multilevel data using the *rximport* command by specifying the *-m* flag. Data is loaded with the row sensitivity label at the first column. If no format file is provided, *rximport* reads the data file in free form mode. Otherwise, input data are interpreted according to the content of the format file, which indicates the position and length of each input column. The Security Administrator is further constrained to raise, lower, or “across” the label of the new row as compared to the session label by individual authorizations.

FDP_RIP_DB.2 TOE full residual information protection

The object reuse security function ensures that before any resource is made available to a subject, it is cleared upon allocation. The Trusted RUBIX administrative utilities (e.g., *`rxauditset, rxauditrpt, rximport, rxexport, rxdump, rxrestore`*, and *`rxrecl`*) are trusted programs executing on the Trusted RUBIX server by administrative personnel. The residual information of the file system objects are cleared and created empty by Trusted Solaris. The Trusted RUBIX server is a trusted process that has subject identity of `RUBIX/RUBIXTP`. The Trusted RUBIX server views the database space as its own address space and ensures that when tables, views, and schemas are dropped, there is no direct and/or indirect information flow from one user to another.

FDP_ROL.2 Advanced rollback

The Discretionary Access Control policy and Mandatory Access Control policy are enforced during trusted recovery of all the operations on all DBMS objects. Trusted RUBIX implements Logging (i.e., rollback/recovery log and the restore log), Transaction Failure Recovery, System Failure Recovery, and Media Failure Recovery to rollback or restore databases to a consistent and secure state after a failure. Trusted RUBIX provides users with the ROLLBACK statement to complete the current transaction by applying none of the database changes. Trusted RUBIX performs rollback from transaction failures during SQL operation and transaction control, *rxexport*, and *rximport*.

FIA_ATD.1 User attribute definition

Trusted RUBIX relies solely on the Identification and Authentication (I&A) mechanism of Trusted Solaris 8, which maintains the security attributes of authorized individual users and groups. These security attributes include user identifier, user clearance, authentication data, group memberships, and security-relevant roles defined as having authorizations. When the client process communicates with a Trusted RUBIX server process, the Trusted RUBIX server uses the user security attributes supplied by Trusted Solaris 8 to identify the user. User Security Attribute Definitions function (IA.UATD) of the Trusted RUBIX server stores and maintains the user’s security attributes, which include the user ID, group ID, user authorizations, and user session sensitivity label during process initialization. The security-relevant role is correlated to the user’s authorization set.

FIA_USB.1 User-subject binding

The client process is an application process initiated by an authenticated user and created by the operating system. The client process is associated with the user attributes of user identity (i.e., user ID and group ID) and the sensitivity label of the user session. When the client process, acting on behalf of the user, initiates connection to a specific database, a Trusted RUBIX server process is created on the server machine to process and respond to client SQL requests. There is one instantiation of the server for each active client.

User-Subject Binding function (IA.USBD) of Trusted RUBIX uses the stored user security attributes (e.g., User ID, Group ID, user authorizations, session sensitivity label) for reference monitor purposes. Even though the Trusted RUBIX server is operating at `SYSHIGH` level under the user-ID `RUBIX` and group-ID `RUBIXTP`, it mediates the access of the DBMS object and the DBMS information flow using the user security attributes obtained from the client process for making the DAC and MAC decisions.

FMT_MOF.1 Management of security functions behavior

Each of the Trusted RUBIX commands ensures the subject is authorized to perform the requested function by checking the user group ID against the authorized security role(s) for executing this command. The Trusted RUBIX *`rxauditset`* utility provides the capability to specify criteria to enable and disable individual audit events to be generated and recorded in the audit trail. This *`rxauditset`* utility can only be run on the server by the DBMS Audit Administrators. The Trusted RUBIX *`rximport`* command with multilevel (i.e., *`-m`*) option provides the capability to import multilevel data into the database table. The *`rximport`* multilevel option can only be used by the DBMS Security Administrators.

FMT_MSA.1 Management of object security attributes

When a user attempts to change the content of an ACL associated with a DMBS object, Trusted RUBIX DAC Security Function ensures the user must have the required privilege (grant or revoke) in the ACL to change the content of that ACL. The MAC Security Function ensures that the session sensitivity label equals the object sensitivity label.

The Trusted RUBIX Security Administrator Role Management Functions, which include *`rxrecl`* command to reclassify the security level of rows of data, the SQL command *`ALTER SESSION SET LABEL`* change session label, and *`rximport`* command with "*`-m`*" option to import multilevel data into the database, are restricted to users in the Security Administrator. For *`rxrecl`* and *`rximport`* the Security Administrator is further constrained to raise, lower, or "across" the label of the new row by individual authorizations. For the *`ALTER SESSION SET LABEL`* command the Security Administrator is constrained to raise, lower, or across the new session label and to submit read-only or read-write operations by individual authorizations.

FMT_MSA.2 Secure security attributes

Trusted RUBIX Grant statement and Revoke statement ensure that every privilege to be granted or revoked is a defined privilege prior to updating the ACL. The *`rxrecl`* utility validates each new sensitivity label to be associated with a specified table row and ensures it is a valid sensitivity label prior to row re-classification (downgrade or upgrade).

FMT_MSA.3 Static attribute initialization

The DAC Security Function provides for a restrictive default access to all objects. By default, objects access privileges defined in the ACL are restricted to the DBMS object creator. When the object is created, the sensitivity label of the object is set to the sensitivity label of the subject; i.e., the session sensitivity label of the user who created the object by the MAC Security Function. Trusted RUBIX does not allow any user to specify alternative initial values to override the default value.

FMT_MTD.1 Management of the TSF data

The Trusted RUBIX DAC Security Function ensures that only the DBMS audit administrator can use *`rxauditset`* and *`rxauditrpt`* commands to select auditing events from the auditable events and to query and review audit records from the audit trails.

FMT_REV.1 Revocation

The revocation of access privilege attributes in an Access Control List of a DMBS object is enforced by the DAC security function. Only database administrators or DBMS users who have the privilege that include *`WITH GRANT OPTION`* may revoke that access privilege associated with an Access Control List entry of a DMBS object. The user must be operating at the same MAC label as the object.

FMT_SMR.1 Security roles

The Trusted RUBIX Security Roles Security Function allows authorized DBMS users to be assigned separate authorizations. There are different sets of security management utilities available to each authorization. Each of these utilities ensures that the user is associated with the utility designated role(s) before any functions are allowed to proceed.

FMT_SMF.1 Specification of Management Functions

The Trusted RUBIX restricts the users to the specified authorizations associated with roles for performing the security management functions.

FPT_RCV.4 Function recovery

Trusted RUBIX implements Logging (i.e., rollback/recovery log and the restore log), Transaction Failure Recovery, System Failure Recovery, and Media Failure Recovery to rollback or restore databases to a correct state after a failure. Trusted RUBIX provides a set of utilities to recover from catastrophic loss, e.g., database backup, restore, and archive using tape or disk. Trusted RUBIX maintains security attributes of its protected objects in a database. Sensitivity labels of objects are kept as the value of "`rowlabel`" column. To modify the sensitivity label of an object, the Security Administrator user uses `rxrecl` utility to update "`rowlabel`" column to the new sensitivity label. The ACLs of objects are kept as tables in a database container. An ACL is created as part of the object creation process and is removed as part of the object deletion process. To modify privileges on an object, the authorized user uses a grant or revoke database operation.

Since Trusted RUBIX stores the TSF data (e.g., ACLs) the same way as user data in databases, it recovers the TSF data lost due to transaction failure the same way as user data. When the Trusted RUBIX server detects a database transaction failure, system failure, or media failure, it either completes successfully or uses the above trusted recovery security functions to rollback or restore database to a consistent and secure state.

FPT_RVM_DB.1 TOE non-bypassability of the TSP

The reference monitor security function ensures that the TSF is always invoked before any functions are allowed to proceed. The Trusted RUBIX client/server architecture is based on the Trusted RUBIX server receiving requests from the client for operation on the database and sending responses to the client. The Reference Monitor security function (PT.RFMN) of the Trusted RUBIX server ensures that the TSF is always invoked before any functions are allowed to proceed. Within the Trusted RUBIX server, the Trusted RUBIX Server Interface subsystem is the client's only entry point into the TOE security functions of Trusted RUBIX. Therefore a client cannot bypass this interface layer. The Server Interface subsystem submits requests from untrusted client to the SQL engine subsystem for checking needed DAC privileges and operation on the database. All the SQL engine subsystem DAC checks and Kernel subsystem MAC checks are invoked, directly or indirectly, by the Server Interface subsystem.

FPT_SEP_DB.1 Domain Separation and Protection Mechanism

The Domain Separation and Protection Mechanisms security function (PT.SEP) ensures that TSF executes in its separated security domain, which is protected from interference and tampering by untrusted subjects. The PT.SEP security function of Trusted RUBIX utilizes the domain separation security function of the underlying OS for process domain separation. Trusted RUBIX database directories and files are owned by user and group `RUBIX/RUBIXTP` and labeled at system high sensitivity and protected by OS MAC and DAC protection mechanisms. Trusted RUBIX Server executable files are owned by user and group `RUBIX/RUBIXTP`, labeled at system low, and protected by the OS MAC and DAC protection mechanisms. Upon execution the process label is set to system high and effective group set to RUBIXTP, allowing access to Trusted RUBIX directories and files. The user client process and the Trusted RUBIX Server process can reside on the same computer or on two physically separated computers. Trusted programs provided for

DBMS Security Administrators can only be executed on the same computer where Trusted RUBIX Server is running.

References

[1] Bell, D. Elliot and Leonard J. LaPadula, *Secure Computer Systems: Unified Exposition and Multics Interpretation,* ESD-TR-75-306, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, Massachusetts, March 1976.

[2] Common Criteria Implementation Board, Common Criteria for Information Technology Security Evaluation, CCIMB-99-031, 032, 033, Version 2.1, August 1999.

[3] Department of Defense Standard, Department of Defense Trusted Computer System Evaluation Criteria (Orange Book), December 1985 (TCSEC).

[4] Protection Profile for Multilevel Operating Systems in Environments Requiring Medium Robustness (MOS-MED PP), version 1.2, 23 May 2001, Information Assurance Directorate, National Security Agency

[5] Security Agency Information Assurance Solutions Technical Directors, Information Assurance Technical Framework (IATF), Release 3.0, National September 2000