MarkLogic Server Enterprise Edition 6.0
Security Target

Version 1.0
December 20, 2013

Prepared for:
MarkLogic Corporation

999 Skyway Road
Suite 200
San Carlos, CA 94070

Prepared By:
Science Applications International Corporation

Common Criteria Testing Laboratory

6841 Benjamin Franklin Drive
Columbia, MD 21046
USA

TABLE OF CONTENTS

LIST OF TABLES

1 Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is MarkLogic Corporation MarkLogic Server Enterprise, Edition 6.0-4 provided by MarkLogic. MarkLogic Server Enterprise Edition is an enterprise-class database or "contentbase" that provides a set of services used to build both content and search applications which query, manipulate and render XML content.

The Security Target contains the following additional sections:

- Security Target Introduction (Section 1)
- TOE Description (Section 2)
- Security Problem Definition (Section 3)
- Security Objectives (Section 4)
- IT Security Requirements (Section 5)
- TOE Summary Specification (Section 6)
- Protection Profile Claims (Section 7)
- Rationale (Section 8)

1.1 Security Target, TOE and CC Identification

ST Title –MarkLogic Server Enterprise, Edition 6.0 Security Target

ST Version – Version 1.0

ST Date – December 20, 2013

TOE Identification –MarkLogic Server Enterprise, Edition 6.0-4

TOE Developer – MarkLogic Corporation

Evaluation Sponsor – MarkLogic Corporation

CC Identification – Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4, September 2012

1.2 Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements, Version 3.1 Revision 4, September 2012.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 4, September 2012.
    - Part 3 Conformant
    - Assurance Level: EAL2 augmented with ALC_FLR.3

1.3 Conventions

The following conventions have been applied in this document:

Extended requirements – Security Functional Requirements not defined in Part 2 of the CC are annotated with a suffix of _EXT.

Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.

Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is identified with a number in parentheses following the base component identifier. For example, iterations of FCS_COP.1 are identified in a manner similar to FCS_COP.1(1) (for the component) and FCS_COP.1.1(1) (for the elements).

Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).

Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).

Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").

Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.4 Glossary

Acronym	Description
API	Application Programming Interface
CC	Common Criteria
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
LAN	Local Area Network
EAL	Evaluation Assurance Level
NTP	Network Time Protocol
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL/TLS	Secure Session Layer/Transport Layer Security in FIPS mode
ST	Security Target
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Function
DBMS	Database Management System
GUI	Graphical User Interface
OS	Operating System
PP	Protection Profile

1.5 Terminology

The terminology below is described in order to clarify the terms used in the ST as well as those used in the TOE product documentation.

Amps — Amps are security objects that temporarily grant role membership to unprivileged users only for the execution of a given function. While executing an "amped" function, the user is temporarily part of the amped role which in turn temporarily grants the user the additional privileges and permissions given by the roles configured in the amp. Amps enable the effect of the additional permissions and privileges to be limited to a particular function.

Permissions	Permissions provide a role with the ability to perform capabilities (that is, read, insert, update, execute) on documents. A permission is a combination of role and capability. Permissions are assigned to documents. Users gain the authority to perform a capability on a document if they are members of the role the permission associates with the capability.
Capabilities	Capabilities are operations on documents: Read, Update, Insert or Execute.
Execute Privileges	Execute privileges allow developers to control authorization for the execution of an XQuery function. These privileges are assigned to a user through a role.
Role	MarkLogic Server implements a role-based security model. A role contains privileges and the privileges allow access to execute code on the system (for example, security management functions). A role also allows access to a documents based on permissions defined on the document.
URI Privileges	Uniform Resource Identifier privileges are used to control the creation of documents with a given URI prefix. In order to create a document with a prefix that has a URI privilege associated with it, a user must be part of a role to which the needed URI privilege is assigned.
Application Server Privileges	Application Server Privileges are Execute Privileges that can be configured to control access to each application server (that is, HTTP or XDBC server). If such a privilege is specified, any users that access the server must possess the specified privilege.

2 TOE Description

The Target of Evaluation (TOE) is MarkLogic Server Enterprise Edition 6.0-4, hereafter referred to as MarkLogic Server or the TOE.

2.1 TOE Overview

The TOE is MarkLogic Corporation's MarkLogic Server software. MarkLogic Server is an enterprise-class database that provides a set of services used to build content and search applications which query, manipulate and render Extensible MarkUp Language (XML) content.

The MarkLogic Server TOE is built with a blend of search engine and database architecture approaches specifically designed to index and retrieve XML content. The TOE's native data format is XML and XML is accepted in an 'as is' form, while content in other formats can be converted to an XML representation or stored as is (in binary or text formats) when loaded into MarkLogic Server. As an XML database, MarkLogic Server manages its own content repository and is accessed using the W3C standard XQuery language, just as a relational database is a specialized server that manages its own repository and is accessed through Structured Query Language (SQL).

The TOE is fully transactional, runs in a distributed environment and can scale to terabytes of indexed content. It is schema independent and all loaded documents can be immediately queried without normalizing the data in advance. MarkLogic Server provides developers with the functionality and programmability, using XQuery as its query language, to build content-centric applications. Developers build applications using XQuery both to search the content and as a programming language in which to develop applications. It is possible to create entire applications using only MarkLogic Server, and programmed entirely in XQuery. Application can also be created using Java or other programming languages that access MarkLogic Server.

The security management functions of the TOE are performed via the Admin Interface, which is a web based browser GUI implemented as a MarkLogic Server web application. This interface allows authorized administrators to manage audit events, user accounts, access control and TOE sessions.

Authorized administrators can also perform security management functions programmatically using the XQuery functions included in XQuery library modules that are included with MarkLogic Server. Theprogrammatic libraries that support security management are the Admin API, the Security API, and the PKI API. The Admin API enables the scripting of administrative tasks that would otherwise need the Admin Interface to perform, including TOE security management tasks (for example, management of TOE sessions, configuration of auditing, and so on). For example, you can write a program using the Admin API to create and configure App Servers, including setting the type of authentication that the App Servers use. Most functions in this library perform administrative tasks and therefore require the user who runs an XQuery program executing these functions to be an authorized administrator. The Security API provide functions for managing objects stored in the security database (users, roles, amps, and privileges). For example, you can use the Security API to create and modify users (including passwords), roles, amps, and privileges. The PKI API provides functions that manage private keys and other cryptographic management functions used with SSL/TLS (HTTPS) in FIPS mode.

Security management functions include the ability to control the creation, management, and configuration of databases, forests, servers, and hosts. Documents are stored in forests. The name forests comes from the fact that XML documents are tree structures and a collection of trees is a forest. One or more forests are gathered together to form a database. Databases are logical units against which you can assign HTTP and XDBC servers and set various runtime configuration options. A host is a single instance of MarkLogic Server running on a single machine. Databases exist as a logical abstraction because in a distributed environment it can be useful to have the same logical database spread across different hosts, perhaps one host with two forests and another with three.

2.2 TOE Architecture

The TOE consists of two subsystems, the Administration subsystem and the Server subsystem. The Administration subsystem provides the Admin Interface to the Server subsystem. The Admin Interface application manages all features of the Server subsystem. It is composed of XQuery programs which are evaluated inside of an HTTP server. The HTTP server evaluates each request and sends a response back as a web page to the requester. The Admin Interface is accessed through HTTPS only (i.e., HTTP over SSL/TLS in FIPS mode).

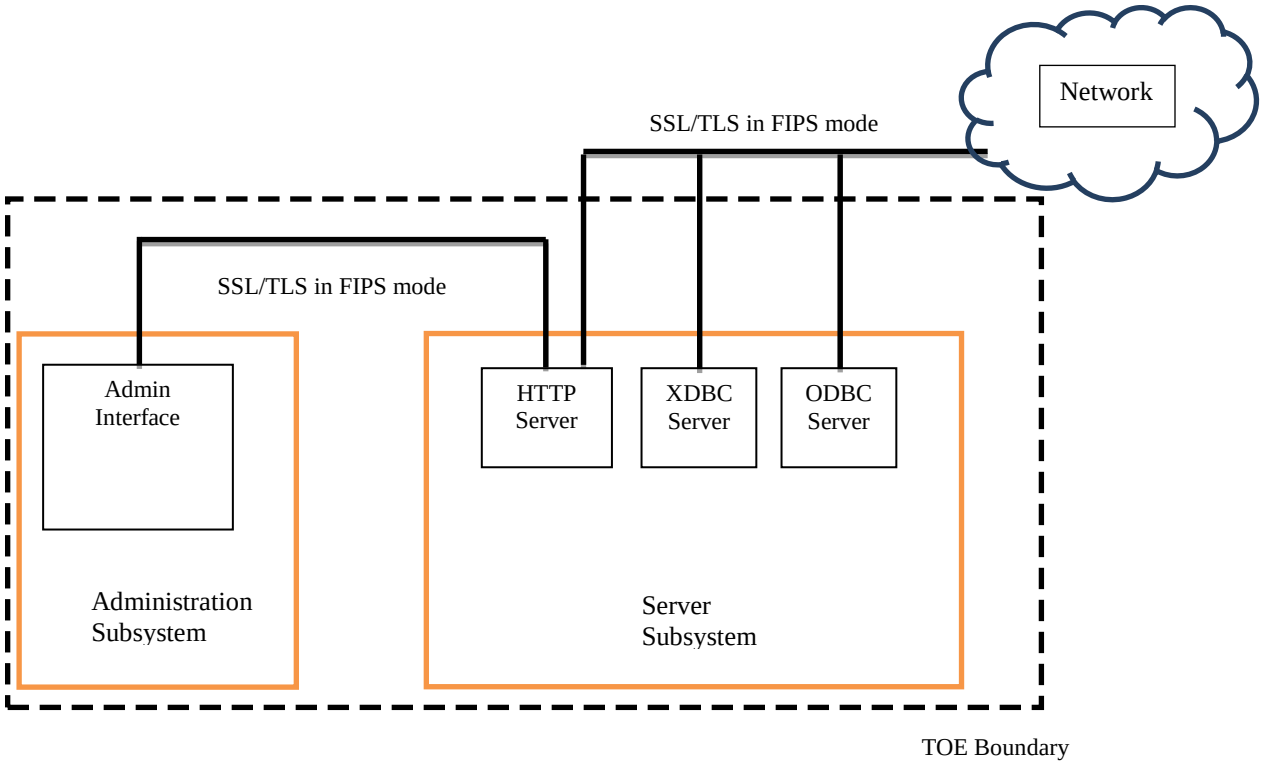


Figure 2-1 TOE Architecture

The TOE supports three interfaces that are available through a network. An HTTP Server offers connectivity for the administrative interface and for customer applications with the Server subsystem. The communication pathways to and from the Server subsystem are depicted in Figure 2-1 by the lines labeled as "SSL/TLS in FIPS mode". Two additional programmatic interfaces are provided by XDBC and ODBC protocols that can also use SSL/TLS in FIPS mode to protect the session. Developers write client applications to use these interfaces in a system that requires access to a backend XML database. In particular, the HTTP and XDBC servers each provide the Admin API, Security API, and PKI API, which are collections of XQuery functions. The API functions are evaluated inside the HTTP and XDBC servers. Consequently, the servers enforce TOE security policy (for example, authentication, security management restrictions, access control, and auditing).

MarkLogic Server includes REST APIs, a Java Client API, and XCC libraries. These libraries are for application development. They do not provide any security functionality. The REST APIs are implemented as XQuery programs that run on a MarkLogic HTTP App Server. The Java Client API is implemented in Java, and calls the REST APIs, which in turn run on a MarkLogic HTTP App Server. The HTTP App Server is an interface to the TOE that honors DAC policy. The XCC libraries run against a MarkLogic XDBC App Server, which is an interface to the TOE that honors DAC policy.

The TOE can be set up as a single instance of MarkLogic Server on a single machine or it can support large scale high-performance architectures through multi-host distributed architectures. The following terminology has been defined for consideration in a TOE distributed environment:

- Cluster – A cluster is a set of one or more instances (see hosts, below) of MarkLogic Server (i.e., the TOE's Server subsystem) that will work together as a unified whole to provide content services. Security management functions of the TOE are performed from the Administration subsystem by connecting to any cluster host.
- Host – A host is a single instance of MarkLogic Server running on a single machine. Even though each host in a cluster can be configured to perform a different task, the full MarkLogic Server software (Server

subsystem) runs on each host. MarkLogic Server Enterprise Edition enables multi-host configurations.[1]

- Cluster Management Group – A cluster management group is a set of hosts with uniform HTTP, XDBC and ODBC configurations (but not necessarily uniform forest configurations). Cluster Management Groups are used to simplify cluster management.
- Forest – A forest is a repository for documents. Each forest is managed by a single host. The mapping of which forest is managed by which host is transparent to queries, as queries are processed against databases, not forests.
- Database – A database is a set of one or more forests that appears as a single contiguous set of content for query purposes. Each forest in a database must be configured consistently. HTTP and XDBC servers evaluate queries against a single database. In addition to databases created by the administrator for user content, MarkLogic Server maintains databases for administrative purposes: *security* databases, which contain user authentication and permissions information; *schema* databases, which are used to store schemas used by the system; *modules* databases, which are used to store executable XQuery code; *last-login* databases, which are used to store session history and data and *triggers* databases, used to store trigger definitions.

2.2.1 TOE Physical Boundaries

The TOE consists of the software applications and network protocol interfaces (described and shown in the diagram above). The Administration subsystem, which provides the Admin Interface, runs using a supported browser, Firefox, Internet Explorer, or Chrome. The Server subsystem applications and network interfaces execute either on Sun Solaris or Linux operating systems. The TOE requires the following hardware and operating system (OS) platforms in the operational environment:

Memory, Disk Space, and Swap Space Requirements

Before installing the software, the system must meet the following minimum requirements:

- 512 MB of system memory, minimum.
- Three times the disk space of the source content to be loaded.
- Swap space at least equal to the amount of physical memory on the machine.

Supported Platforms – Server Subsystem

The server subsystem is supported on the following platforms for the evaluated configuration:

- Sun Solaris 10 (x64)
- Red Hat Enterprise Linux 5.0 (x64)
- Red Hat Enterprise Linux 6 (x64)

Supported Platforms – Administration Subsystem

The MarkLogic Server administration subsystem is supported on the following browsers for the evaluated configuration:

- Firefox 17 on Windows and Mac OS
- Internet Explorer 8 and 9 on Windows
- Chrome 23 on Windows and Mac OS

Other browser/platform combinations may work but are not as thoroughly tested by MarkLogic.

[1] The evaluated configuration only includes the MarkLogic Server Enterprise Edition which supports multi-host configurations.

As noted previously, the TOE can be deployed on a single machine or in a distributed environment across multiple machines. In a distributed environment, the TOE is a cluster of hosts as defined above. The hosts communicate using TLS to protect transmitted data from disclosure or undetected modification.

The TOE relies on the hosting OS to protect its applications, processes, and any locally stored data. The TOE itself maintains a security domain that protects it from interference and tampering by untrusted subjects within the TOE scope of control. Web browsers in the environment are used to access the Admin Interface and the HTTP server through its HTTPS interface, and to terminate a session. The Admin Interface prompts the user to authenticate with a valid username and password in order to log in for a session. As is standard in browser-based applications, the browser caches and automatically re-issues the login credentials for each request throughout the browser session. These credentials are valid until the browser is closed, which terminates the session. When the browser is restarted, the user will once again be prompted to authenticate with a valid username and password.

A customer application on the network can also communicate with the TOE's App Servers (HTTP, XDBC or ODBC). The TOE supports the use of SSL/TLS version 1.0 in FIPS mode sessions by a customer applications on the network that communication with the TOE's App Servers (HTTP server, XDBC server or ODBC server). The TOE requires applications that use the Admin API, Security API, and PKI API to communicate with the HTTP App Server and XDBC App Server using TLS. Customer client applications are not part of the TOE.

2.2.2 TOE Logical Boundaries

This section identifies the security functions that MarkLogic Server Enterprise Edition 6.0-4 provides. The logical boundaries of the TOE include the security functions of the TOE interfaces. The TOE logically supports the following security functions:

- Security Audit
- Cryptographic Support
- User Data Protection
- Identification & Authentication
- Security Management
- Protection of the TSF
- TOE Access

2.2.2.1 **Security Audit**

The TOE generates audit records that include date and time of the event, subject identity and outcome for security events. The TOE provides authorized administrators with the ability to include and exclude auditable events based on user identity, role, event type, object identity and success and failure of auditable security events. When appropriate, the TOE also associates audit events with the identity of the user that caused the event. The environment stores the audit records and also provides the system clock information that is used by the TOE to timestamp each audit record.

2.2.2.2 **Cryptographic Support**

The Secure Sockets Layer protocol or Transport Layer Security (TLS 1.0) protocol (referred to in this document as SSL/TLS in FIPS mode) is used to provide protection of the communications surrounding the remote administrative sessions from disclosure and from modification (referred to as SSL/TLS in FIPS mode in this security target). For communication between a customer application on a network and the HTTP server, XDBC server, or ODBC server of the TOE, the TOE offers the use of a SSL/TLS session in FIPS mode to protect these communications. Finally, the TOE uses an SSL/TLS in FIPS mode protected channel to distribute TSF data when it is transmitted between distributed parts of the TOE (that is, hosts within a cluster).

The TOE uses OpenSSL object module version 2.0 which has undergone a FIPS 140-2 certification (certificate #1747). The TOE includes an OpenSSL object module built without modification from the source code of the OpenSSL FIPS certification. All references to "the TOE" performing cryptographic operations in this security target are indicating that the TOE is performing the operation through its use of the OpenSSL object module.

2.2.2.3 User data protection

The TOE enforces a Discretionary Access Control (DAC) policy which restricts access to TOE-controlled object(s). Users of the TOE are identified and authenticated by the TOE before any access to the system is granted. Once access to the system is granted, authorization provides the mechanism to control what functions a user is allowed to perform based on the user's role membership. Access to all TOE-controlled objects is denied unless access, based on role membership, is explicitly allowed. The authorized administrator role shall be able to access any object regardless of the object's permissions. The TOE also provides amplifications or "amps" which temporarily grant roles to a user only for the execution of a specific function. Therefore, the DAC policy can also be extended by a user who is temporarily granted the privileged role in order to perform a specific "amped" function. The TOE also ensures that any previous information content of a resource is made unavailable upon the allocation of the resource to an object. Memory or disk space is only allocated when the size of the new data is first known, so that all previous data is overwritten by the new data.

2.2.2.4 Identification & Authentication

The TOE requires users to provide unique identification and authentication data before any access to the system is granted and further restricts access to DBMS-controlled objects based on role membership. The TOE maintains the following security attributes belonging to individual users: role membership, and password. The TOE uses these attributes to determine access.

The TOE provides a password plug-in functionality that allows administrators to write custom code to require passwords to conform to specific rules (e.g., the number of characters, special characters, last change date).

2.2.2.5 Security Management

The security functions of the TOE are managed by authorized administrators via the web-based Admin Interface, or application written using the Admin API, Security API, PKI API, and built-in admin functions. The ST defines the security role of 'authorized administrator'. Authorized administrators perform all security functions of the TOE including managing audit events, user accounts, access control and TOE sessions.

2.2.2.6 Protection of the TSF

The TOE provides protection mechanisms for its security functions. One of the protection mechanisms is that users must authenticate and have the appropriate permissions before any administrative operations or access to TOE data and resources can be performed on the system. The TOE also maintains a security domain that protects it from interference and tampering by untrusted subjects within the TOE scope of control.

Communication with remote administrators is protected by SSL/TLS in FIPS mode, protecting against the disclosure and undetected modification of data exchanged between the TOE and the administrator. Communication with remote customer applications can also utilize SSL/TLS in FIPS mode to protect against the disclosure and undetected modification of data exchanged between the TOE and the customer application. Customer applications must determine whether the use of SSL/TLS in FIPS mode is necessary for that specific customer application's data.

The TOE ensures that TSF data is encrypted and remains consistent when transmitted between parts of the TOE. The TOE provides consistency of TSF data between distributed parts of the TOE by regularly monitoring the configuration file and security database for changes and distributing the updated configuration file or security database to all parts of the cluster. The TOE utilizes a TLS protected channel to distribute TSF data among a cluster.

2.2.2.7 TOE Access

The TOE restricts the maximum number of concurrent sessions that belong to the same user by enforcing an administrator configurable number of sessions per user. The TOE also denies session establishment based on attributes that can be set explicitly by authorized administrators including role identity, time of day and day of week.

Upon successful session establishment, the TOE stores and retrieves the date and time of the last successful session establishment to the user. It also stores and retrieves the date and time of the last unsuccessful session establishment and the number of unsuccessful attempts since the last successful session establishment. This information is collected by the TOE Access security function, because the information pertains to user's attempts to access the TOE. The information gathered by the TOE pertains to historical session establishment actions by a user.

2.3 TOE Documentation

Mark Logic has a number of administration and configuration guides for the TOE which include the following:

- *MarkLogic Server Administrator's Guide*, MarkLogic 6, September 2012 (Last Revised: 6.0-4, July 2012)
- *MarkLogic Server Understanding and Using Security*, MarkLogic 6, September 2012 (Last Revised: 6.0-1, September 2012)
- *MarkLogic Server Scalability, Availability, and Failover Guide*, MarkLogic 6, September 2012 (Last Revised: 6.0-1, September 2012)
- *MarkLogic Common Criteria Evaluated Configuration Guide*, MarkLogic 6, September 2012 (Last Revised: 6.0-4, July, 2013)
- *MarkLogic Server Installation Guide for All Platforms*, MarkLogic 6, September 2012 (Last Revised: 6.0-1, September 2012)

3 Security Problem Definition

This section describes the security environment in which the TOE operates. The security environment is defined in terms of supported organizational policies, assumptions made by the TOE and threats to the TOE.

3.1 Assumptions

The following conditions are assumed to exist in the operational environment.

Assumption	Definition
A.NO_EVIL	TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.
A.OS_TIME	The OS in the environment shall be able to provide reliable time stamps for use by the TOE.
A.TRUSTED_OS	The underlying OS is trusted to provide protection of the DBMS processes and stored data from other processes running on the underlying OS.
A.NO_GENERAL_PURPOSE	It is assumed that there are no general-purpose computing capabilities (e.g., compilers or user applications) available on the DBMS, other than those services necessary for the operation, administration and support of the DBMS.
A.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the environment.
A.AUTH	Passwords are encrypted during the authentication process.
A.CLIENT	The web browsers used to access the Admin Interface perform correctly such that when the browser is closed, the active Admin session is terminated. Client applications used to access the Admin API, Security API, and PKI API will perform correctly and when the application is closed, the active Admin session will be terminated.

3.2 Threats

Threat	Definition
T.UNDETECTED_ACTIONS	Malicious remote users or external IT entities may take actions that adversely affect the security of the TOE. These actions may remain undetected and thus their effects cannot be effectively mitigated.
T.UNAUTHORIZED_ACCESS	A user may gain unauthorized access to the TSF data and TSF executable code. A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to TSF data or TSF resources. A malicious user, process, or external IT entity may misrepresent itself as the TSF to obtain identification and authentication data.
T.TSF_COMPROMISE	A user may cause, through an unsophisticated attack, TSF data, or executable code to be inappropriately accessed (viewed, modified, or deleted).

4 Security Objectives

This chapter identifies the security objectives of the TOE and its environment. Security objectives identify the responsibilities of the TOE and the support need by the TOE from its environment.

4.1 Security Objectives for the TOE

Objective	Definition
O.ACCESS_HISTORY	The TOE will store and retrieve information (to authorized users) related to previous attempts to establish a session.
O.AUDIT_GENERATION	The TOE will provide the capability to detect and create records of security relevant events associated with users.
O.MANAGE	The TOE will provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.
O.MEDIATE	The TOE must protect user data in accordance with its security policy.
O.RESIDUAL_INFORMATION	The TOE will ensure that any information contained in a protected resource within its Scope of Control is not released when the resource is reallocated.
O.TOE_ACCESS	The TOE will provide mechanisms that control a user's logical access to the TOE.
O.PROTECTED_COMMUNICATIONS	The TOE will provide protected communication channels for administrators and support protected communication channels for non-administrative users.

4.2 Security Objectives for the Environment

Objective	Definition
OE.AUTH	Password encryption during the authentication process is provided by the web browser.
OE.CLIENT	The web browsers used to access the Admin Interface will perform correctly and when the browser is closed, the active Admin session will be terminated. Client applications used to access the Admin API, Security API, and PKI API will perform correctly and when the application is closed, the active Admin session will be terminated.
OE.NO_GENERAL_ PURPOSE	There will be no general-purpose computing capabilities (e.g., compilers or user applications) available on the TOE, other than those services necessary for the operation, administration and support of the DBMS.
OE.PHYSICAL	Physical security will be provided within the domain for the value of the IT assets protected by the TOE and the value of the stored, processed, and transmitted information.
OE.PROCESS	The environment provides one or more dedicated processes for the exclusive use of the TOE that isolates the TOE from non-TOE processes which allow the TOE

	to use real and virtual resources offered in the environment.
OE.STORAGE	The environment provides a protected data storage mechanism (e.g., files) that allows the TOE to store information such that the environment prevents unauthorized modification or deletion of TOE data.
OE.TIME	The environment provides a reliable time source for use by the TOE.
OE.TRUSTED_ADMIN	TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

5 IT Security Requirements

The security requirements for the TOE have been drawn from Parts 2 and 3 of the Common Criteria. The security functional requirements have been selected to correspond to the actual security functions implemented by the TOE while the assurance requirements have been selected to offer a low to moderate degree of assurance that those security functions are properly realized.

5.1 Extended Component Definition

This Security Target includes Security Functional Requirements (SFR) that are not drawn from CC Part 2. These Extended SFRs are identified by having a label '_EXT' after the requirement name for TOE SFRs. The structure of the extended SFRs is modeled after the SFRs included in CC Part 2. The structure is as follows:

A. Class – The extended SFRs included in this ST are part of the identified classes of requirements.

B. Family – The extended SFRs included in this ST are part of several SFR families including the new families defined below.

C. Component – The extended SFRs are not hierarchical to any other components, though they may have identifiers terminating on other than "1". The dependencies for each extended component are identified in the TOE SFR Dependencies section of this ST (Section 8.3, Requirement Dependency Rationale).

5.1.1 Extended Family Definitions

5.1.1.1 **FCS_CKM_EXT**

Family Behavior

This SFR is intended to be another requirement in the FCS_CKM family. It is an alternative to using the FCS_CKM.4 requirement.

Management: FCS_CKM_EXT.4

There are no management activities foreseen.

Audit: FCS_CKM_EXT.4

The following actions should be auditable if FAU_GEN Security audit data generation is included:

Basic level:

- Failure on invoking functionality.

5.1.1.1.1 FCS_CKM_EXT.4 – Cryptographic Key Zeroization

Hierarchical to: No other components.

Dependencies: (FDP_ITC.1 or FDP_ITC.2) or FCS_CKM.1

FCS_CKM_EXT.4.1 The TSF shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

5.1.1.2 **FCS_HTTPS_EXT**

Family Behavior

This family identifies the behavior of the TOE when the HTTPS protocol is implemented.

Management: FCS_HTTPS_EXT.1

There are no management activities foreseen.

Audit: FCS_HTTPS_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included:

Basic Level:

- Failure to establish an HTTPS Session
- Establishment/Termination of an HTTPS session

5.1.1.2.1 FCS_HTTPS_EXT.1 – HTTPS Protocol

Hierarchical to: No other components.

Dependencies: FCS_TLS_EXT.1

FCS_HTTPS_EXT.1.1 The TSF shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2 The TSF shall implement HTTPS using TLS as specified in FCS_TLS_EXT.1.

5.1.1.3 FCS_TLS_EXT

Family Behavior

This family identifies the behavior of the TOE when the Transport Layer Transport Layer Security (TLS) protocol is implemented.

Management: FCS_TLS_EXT.1

There are no management activities foreseen.

Audit: FCS_TLS_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included:

Basic Level:

- Failure to establish an TLS Session
- Establishment/Termination of an TLS session

5.1.1.3.1 FCS_TLS_EXT.1 – Transport Layer Security Protocol

Hierarchical to: No other components.

Dependencies: FCS_COP.1

FCS_TLS_EXT.1.1 The TSF shall implement one or more of the following protocols [selection: TLS 1.0 (RFC 2346), TLS 1.1 (RFC 4346), TLS 1.2 (RFC 5246)] supporting the following ciphersuites:

Mandatory Ciphersuites:
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
Optional Ciphersuites:
[selection:
None
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
].

5.1.1.4 FIA_PMG_EXT

Family Behavior

This family defines the password management capabilities that are required for administrator passwords.

Management: FIA_PMG_EXT.1

There are no management activities foreseen.

Audit: FIA_PMG_EXT.1

There are no auditable events foreseen.

5.1.1.4.1 FIA_PMG_EXT.1 – Password Management

Hierarchical to: No other components.

Dependencies: None

FIA_PMG_EXT.1 The TSF shall provide the following password management capabilities for administrative passwords:

- Passwords shall be able to be composed of any combination of upper and lower case letters, numbers, and the following special characters: [selection: "!", "@", "#", "$", "%", "^", "&", "*", "(", ")", [assignment: other characters]].

5.1.1.5 FPT_APW_EXT

Family Behavior

This family defines the protections required for administrator passwords used by the TOE for authentication.

Management: FPT_APW_EXT.1

There are no management activities foreseen.

Audit: FPT_APW_EXT.1

There are no auditable events foreseen.

5.1.1.5.1 FPT_PTD_EXT.1 – Protection of Administrator Passwords

Hierarchical to: No other components.

Dependencies: None

FPT_APW_EXT.1.1 The TSF shall store passwords in non-plaintext form..

FPT_APW_EXT.1.2 The TSF shall prevent reading of plaintext passwords.

5.1.1.6 FPT_SKP_EXT

Family Behavior

This family defines the protections required for critical security parameters used by the TOE for authentication or cryptography.

Management: FPT_SKP_EXT.1

There are no management activities foreseen.

Audit: FPT_SKP_EXT.1

There are no auditable events foreseen.

5.1.1.6.1 FPT_SKP_EXT.1 – Protection of TSF Data

Hierarchical to: No other components.

Dependencies: None

FPT_SKP_EXT.1.1 The TSF shall prevent reading of [assignment: critical security parameters].

5.1.1.7 FPT_TRC_EXT

Family Behavior

This family requires that the TOE provide a mechanism to ensure TSF data is consistent between distributed parts of the TOE.

Management: FPT_TRC_EXT.1

There are no management activities foreseen.

Audit: FPT_TRC_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included:

Basic Level:

- Restoring consistency

5.1.1.7.1 Internal TSF consistency (FPT_TRC_EXT.1)

Hierarchical to: No other components.

Dependencies: FPT_ITT.1 Basic internal TSF data transfer protection

FPT_TRC_EXT.1.1 The TSF shall ensure that TSF data is consistent between parts of the TOE by providing a mechanism to bring inconsistent TSF data into a consistent state in a timely manner.

5.1.1.8 **FTA_TAH_EXT**

Family Behavior

This family requires that the TOE store and retrieve information about the user's prior attempts at session establishment.

Management: FTA_TAH_EXT.1

There are no management activities foreseen.

Audit: FTA_TAH_EXT.1

There are no auditable events foreseen.

5.1.1.8.1 TOE access history (FTA_TAH_EXT.1)

Hierarchical to: No other components.

Dependencies: None

FTA_TAH_EXT.1.1 Upon successful session establishment, the TSF shall store and retrieve the [assignment: list of saved information pertaining to session establishment such as date, time or location] of the last successful session establishment to the user.

FTA_TAH_EXT.1.2 Upon successful session establishment, the TSF shall store and retrieve the [assignment: list of saved information pertaining to session establishment such as date, time or location] of the last unsuccessful attempt to session establishment and the number of unsuccessful attempts since the last successful session establishment.

5.1.2 Extended Requirements Rationale:

5.1.2.1 **Legacy Extended Requirements**

The following SFRs are modeled from requirements defined by an old (now sunset) protection profile for Database Management Systems. Earlier versions of this TOE satisfied these requirements prior to the PP being sunset and the SFRs are being retained in this ST to indicate a continuity of product functionality.

- FPT_TRC_EXT.1:

FPT_TRC_EXT.1 has been created to require timely consistency of replicated TSF data. Although there is a Common Criteria Requirement that attempts to address this functionality, it falls short of the needs of the environment in this security target.

Specifically, FPT_TRC.1.1 states "The TSF shall ensure that TSF data is consistent when replicated between parts of the TOE." In the widely distributed environment of this TOE, this is an infeasible requirement. For TOEs with a very large number of components, 100 percent TSF data consistency is not achievable and is not expected at any specific instant in time.

Another concern lies in FPT_TRC.1.2 that states that when replicated parts of the TSF are "disconnected", the TSF shall ensure consistency of the TSF replicated data upon "reconnection". Upon first inspection, this seems reasonable, however, when applying this requirement it becomes clear that it dictates specific mechanisms to determine when a component is "disconnected" from the rest of the TSF and when it is "reconnected". This is problematic in this TOE's environment in that it is not the intent of the authors to dictate that distributed TSF components keep track of connected/disconnected components.

Thus, this extended requirement is intended to only require a mechanism that provides TSF data consistency in a timely manner after it is determined that it is inconsistent.

- FTA_TAH_EXT.1:

  The TOE cannot force a client browser to display a message. Thus, this requirement is based upon FTA_TAH.1, but has been modified to require the TOE to store and retrieve the access history instead of displaying it.

5.1.2.2 **Network Device PP Influenced**

The following SFRs are modeled from those found in the NDPP.

- FCS_CKM_EXT.4 Cryptographic Key Zeroization

  This requirement was copied exactly from the NDPP where it is used as a replacement for FCS_CKM.4.

- FCS_HTTPS_EXT.1: Explicit HTTPS

  This requirement was copied exactly from the NDPP where it is used as a requirement specific to the HTTPS protocol.

- FCS_TLS_EXT.1: Explicit TLS

  This requirement was copied exactly from the NDPP where it is used as a requirement specific to the TLS protocol.

- FIA_APW_EXT.1: Protection of Administrator Passwords

  - This requirement was copied exactly from the NDPP where it is used as a requirement to define constraints on administrator password management.

- FIA_SKP_EXT.1: Password Management

  This requirement was modeled from the NDPP where it is used as a requirement to define protection of cryptographic keys. The NDPP version was changed to accommodate identification of key types, since not all TOE use the keys identified in the NDPP requirement.

- FPT_PTD_EXT.1: Protection of TSF Data

  This Requirement was copied from the NDPP where it was stated as two similar requirements: FPT_PTD.1(1) and FPT_PTD.1(2). The naming and numbering of the requirement has been changed to more accurately reflect its nature as an extended requirement. It has also been changed to show assignments which allow the NDPP requirements to be included in this ST without modification.

5.2 TOE Security Functional Requirements

The following table describes the SFRs that are candidates to be satisfied by MarkLogic MarkLogic Server Enterprise.

Requirement Class	Requirement Component
FAU: Security Audit	FAU_GEN.1: Audit data generation
	FAU_GEN.2: User identity association
	FAU_SEL.1: Selective Audit
FCS: Cryptographic support	FCS_CKM.1: Cryptographic key generation
	FCS_CKM_EXT.4: Cryptographic Key Zeroization
	FCS_COP.1(1): Cryptographic Operation (for data encryption/decryption)
	FCS_COP.1(2): Cryptographic Operation (for cryptographic signatures using ECDSA)
	FCS_COP.1(3): Cryptographic Operation (for cryptographic hashing)
	FCS_COP.1(4): Cryptographic Operation (for keyed-hash message authentication)
	FCS_COP.1(5): Cryptographic Operation (for cryptographic signatures using rDSA)
	FCS_HTTPS_EXT.1: HTTPS Protocol
	FCS_TLS_EXT.1: Transport Layer Security Protocol
FDP: User data protection	FDP_ACC.1: Subset access control
	FDP_ACF.1: Security attribute based access control
	FDP_RIP.2: Full residual information protection
FIA: Identification and authentication	FIA_ATD.1: User attribute definition
	FIA_UAU.2: Timing of authentication
	FIA_UID.2: User identification before any action
	FIA_PMG_EXT.1: Password Management
FMT: Security management	FMT_MSA.1: Management of security attributes
	FMT_MSA.3: Static attribute initialization
	FMT_MTD.1(1): Management of TSF data (TSF data)
	FMT_MTD.1(2): Management of TSF data (audit selection)
	FMT_REV.1(1): Revocation (users)
	FMT_REV.1(2): Revocation (objects)
	FMT_SMF.1: Specification of Management Functions
	FMT_SMR.1: Security roles
FPT: Protection of the TSF	FPT_APW_EXT.1: Protection of Administrator Passwords
	FPT_SKP_EXT.1: Protection of TSF Data
	FPT_ITT.1: Basic internal TSF data transfer protection
	FPT_TRC_EXT.1: Internal TSF Consistency
TOE Access	FTA_MCS.1: Basic limitation on multiple concurrent sessions
	FTA_TAH_EXT.1: TOE Access History
	FTA_TSE.1: TOE Session Establishment
FTP: Trusted path/channels	FTP_TRP.1(1): Trusted Path for administrators
	FTP_TRP.1(2): Trusted Path for users

Table 5-1 TOE Security Functional Components

5.2.1 Security audit (FAU)

5.2.1.1 **Audit data generation (FAU_GEN.1)**

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 [***not specified***] level of audit; and [

c) **all administrative actions;**

d**) successful use of an amp;**

e) The specifically defined auditable events listed in Table 5-2 Auditable Events].

FAU_GEN.1.2 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, 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, [**information specified in column three of Table 5-2 Auditable Events**].

Table 5-2 Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None	
FAU_GEN.2	None	
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	The identity of the authorized administrator that made the change to the audit configuration.
FDP_ACC.1	None	
FDP_ACF.1	All requests to perform an operation on an object covered by the SFP.	The identity of object and the subject performing the operation.
FDP_RIP.2	None	
FIA_ATD.1	None	
FIA_UAU.2	Unsuccessful use of the authentication mechanisms	Provided user identity, origin of the attempt (e.g., IP address).
FIA_UID.2	Unsuccessful use of the user identification mechanism, including the user identity provided.	The user identity provided.
FIA_PMG_EXT.1	None	
FMT_MSA.1	None	
FMT_MSA.3	None	
FMT_MTD.1(1)	None	
FMT_MTD.1(2)	None	
FMT_REV.1(1)	Unsuccessful revocation of security attributes.	Identity of individual attempting to revoke security attributes.
FMT_REV.1(2)	Unsuccessful revocation of security attributes.	Identity of individual attempting to revoke security attributes.
FMT_SMF.1	Use of the management functions.	Identity of the administrator performing these functions.
FMT_SMR.1	Modifications to the group of users that are part of a role.	Identity of authorized administrator modifying the role definition.
FPT_APW_EXT.1	None	
FPT_SKP_EXT.1	None	
FTA_MCS.1	Rejection of a new session based on the limitation of multiple concurrent sessions.	None
FTA_TAH_EXT.1	None	
FTA_TSE.1	Denial of a session establishment due to the session establishment mechanism.	Identity of the individual attempting to establish the session.

5.2.1.2 User identity association (FAU_GEN.2)

FAU_GEN.2.1 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.

5.2.1.3 Selective audit (FAU_SEL.1)

FAU_SEL.1.1 The TSF shall be able to select the set of events to be audited from the set of all audited events based on the following attributes:

a) [***object identity,***

b) ***user identity,***

c) ***event type***]
d) [**role,**
e) success of auditable security events, and
f) failure of auditable security events].

5.2.2 Cryptographic Support (FCS)

5.2.2.1 Cryptographic key generation (FCS_CKM.1)

FCS_CKM.1.1 The TSF shall generate **asymmetric** cryptographic keys **used for key establishment** in accordance with ~~a specified cryptographic key generation algorithm~~ [

NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for elliptic curve-based key establishment schemes and implementing]

and specified cryptographic key sizes [**equivalent to, or greater than, a symmetric key strength of 112 bits**] that meet the following [

NIST Special Publication 800-56A implementing "NIST curves" P-256, P-384 and P-521 defined in FIPS PUB 186-3, "Digital Signature Standard"].

5.2.2.2 Cryptographic Key Zeroization (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1 The TSF shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

5.2.2.3 Cryptographic Operation (for data encryption/decryption) (FCS_COP.1(1))

FCS_COP.1.1(1) The TSF shall perform [**encryption and decryption**] in accordance with a specified cryptographic algorithm [**AES operating in CBC modes**] and cryptographic key sizes [**128-bits, and 256-bits**] that meets the following: [

- **FIPS PUB 197, 'Advanced Encryption Standard (AES)'**
- **NIST SP 800-38A**].

5.2.2.4 Cryptographic Operation (for cryptographic signatures using ECDSA) (FCS_COP.1(2))

FCS_COP.1.1(2) The TSF shall perform [**cryptographic signature services**] in accordance with a specified cryptographic algorithm [**Elliptic Curve Digital Signature Algorithm (ECDSA)**] and cryptographic key sizes [**of 256 bits or greater**] that meet the following: [**FIPS PUB 186-3, "Digital Signature Standard" and the TSF shall implement "NIST curves" P-256, P-384 and P-521 (as defined in FIPS PUB 186-3, "Digital Signature Standard")**].

5.2.2.5 Cryptographic Operation (for cryptographic hashing) (FCS_COP.1(3))

FCS_COP.1.1(3) The TSF shall perform [**cryptographic hashing services**] in accordance with a specified cryptographic algorithm [**SHA-1, , SHA-256**] and ~~cryptographic key~~ **message digest** sizes [**160 or 256 bits**] that meet the following: [**FIPS Pub 180-3, 'Secure Hash Standard'**].

5.2.2.6 Cryptographic Operation (for keyed-hash message authentication) (FCS_COP.1(4))

FCS_COP.1.1(4) The TSF shall perform [**keyed-hash message authentication**] in accordance with a specified cryptographic algorithm [**SHA-1 and SHA-256**] and cryptographic key sizes **[160 or 256 bits], and message digest sizes 160 or 256 bits** that meet the following:

[**FIPS Pub 198-1, 'The Keyed-Hash Message Authentication Code', and FIPS Pub 180-3, 'Secure Hash Standard**'].

5.2.2.7 Cryptographic Operation (for cryptographic signatures using rDSA) (FCS_COP.1(5))

FCS_COP.1.1(5) The TSF shall perform [**cryptographic signature services**] in accordance with a specified cryptographic algorithm [**RSA Digital Signature Algorithm (rDSA)**] and cryptographic key sizes [**of 2048 bits or greater**] that meet the following: [**FIPS PUB 186-2 or FIPS Pub 186-3, "Digital Signature Standard"**].

5.2.2.8 HTTPS Protocol (FCS_HTTPS_EXT.1)

FCS_HTTPS_EXT.1.1 The TSF shall implement the HTTPS protocol that complies with RFC 2818.
FCS_HTTPS_EXT.1.2 The TSF shall implement HTTPS using TLS as specified in FCS_TLS_EXT.1.

5.2.2.9 Transport Layer Security Protocol (FCS_TLS_EXT.1)

FCS_TLS_EXT.1.1 The TSF shall implement one or more of the following protocols [**TLS 1.0 (RFC 2246)**] supporting the following ciphersuites:
Mandatory Ciphersuites:
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
Optional Ciphersuites: [
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA _WITH_AES_128_GCM_SHA256
].

5.2.3 User data protection (FDP)

5.2.3.1 Subset access control (FDP_ACC.1)

FDP_ACC.1.1 The TSF shall enforce the [**Discretionary Access Control policy**] on [**all subjects, documents, and operations read, update, insert, and execute (called capabilities) and create**].

5.2.3.2 Security attribute based access control (FDP_ACF.1)

FDP_ACF.1.1 The TSF shall enforce the [**Discretionary Access Control policy**] to objects based on the following: [

- **Subject attributes:**
  - **the authorized user identity and**
  - **role membership associated with a subject and**
- **Object attributes:**
  - **object identity,**
  - **permissions consisting of (capability, role) pairs where roles may have associated compartments, and**
  - **protected collections where a protected collection has associated permissions**].

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [

a) The TSF shall allow a subject a capability to a document if the user identity is assigned to or inherits roles that satisfy all the following conditions:
i) A role is permitted the requested capability for the document.
ii) If the capability is update and the document belongs to one or more protected collection, then
- ***for each protected collection, at least one subject role is permitted the update capability for the collection.***

iii) If the document belongs to one or more Compartment Security compartments, then

- ***for each document permission with a compartmented role, the subject must have that role in order to perform the capability.***

Otherwise, the TSF shall deny the capability.

b) The TSF shall allow a subject to create a document within a given URI if the user identity is assigned to or inherits roles that satisfy all the following conditions:

i) If the document URI does not include a URI prefix with a URI privilege, then either a role has the any-uri privilege or a role has the unprotected-uri privilege.

ii) If the document URI includes one or more URI prefixes with a URI privilege, then either a role has the any-uri privilege or for each URI privilege a role has the URI privilege.

].

FDP_ACF.1.3 The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [

- **The TSF shall grant a subject with the admin role all modes of access to any object regardless of the object's permissions.**
- **Amps[2] can be used to temporarily grant the a privileged role to an unprivileged user, thereby extending the DAC policy to allow them to evaluate specific functions**].

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the [**no additional explicit denial rules**].

5.2.3.3 Full residual information protection (FDP_RIP.2)

FDP_RIP.2.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the [**allocation of the resource to**] all objects.

5.2.4 Identification and authentication (FIA)

5.2.4.1 User attribute definition (FIA_ATD.1)

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

- **Database user identifier;**
- **role membership; and**
- **Password**].

5.2.4.2 User authentication before any action (FIA_UAU.2)

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

5.2.4.3 User identification before any action (FIA_UID.2)

FIA_UID.2.1 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

5.2.4.4 Password Management (FIA_PMG_EXT.1)

FIA_PMG_EXT.1.1 The TSF shall provide the following password management capabilities for administrative passwords:

[2] For further information on amplifications or "amps", please refer to Section 6.1.2.

1. Passwords shall be able to be composed of any combination of upper and lower case letters, numbers, and the following special characters: [**“!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)”]**].

5.2.5 Security management (FMT)

5.2.5.1 Management of security attributes (FMT_MSA.1)

FMT_MSA.1.1 The TSF shall enforce the [**Discretionary Access Control policy**] to restrict the ability to [***[manage]***] the security attributes to [**authorized administrators with the required privileges and database users as allowed by Discretionary Access Control policy**].

5.2.5.2 Static attribute initialization (FMT_MSA.3)

FMT_MSA.3.1 The TSF shall enforce the [**Discretionary 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 [**authorized administrators with the required privileges and database users as allowed by the Discretionary Access Control policy**] to specify alternative initial values to override the default values when an object or information is created.

5.2.5.3 Management of TSF data (TSF data) (FMT_MTD.1(1))

FMT_MTD.1.1(1) The TSF shall restrict the ability to [***[manage]***] the [**TSF data**] to [**authorized administrators**].

5.2.5.4 Management of TSF Data (Audit Selection) (FMT_MTD.1(2))

FMT_MTD.1.1(2) The TSF shall restrict the ability to [***[determine the selection criteria for]***] the [**set of events to be audited**] to [**authorized administrators**].

5.2.5.5 Revocation (users) (FMT_REV.1(1))

FMT_REV.1.1(1) The TSF shall restrict the ability to revoke **[role membership, password]** associated with the [***users***] under the control of the TSF to [**the authorized administrator**].

FMT_REV.1.2(1) The TSF shall enforce the rules **[**

- **On the revocation host, revocation is effective on the next session that starts after the revocation request is committed.**
- **On other hosts in a cluster, revocation is effective no later than the receipt of the next heartbeat received from the revocation host**].

5.2.5.6 Revocation (objects) (FMT_REV.1(2))

FMT_REV.1.1(2) The TSF shall restrict the ability to revoke **[access operations]** associated with the [***objects***] under the control of the TSF to [**the authorized administrator and database users as allowed by the Discretionary Access Control policy**].

FMT_REV.1.2(2) The TSF shall enforce the rules [

- **On the revocation host, revocation is effective on the next session that starts after the revocation request is committed.**
- **On other hosts in a cluster, revocation is effective no later than the receipt of the next heartbeat received from the revocation host**].

5.2.5.7 Specification of Management Functions (FMT_SMF.1)

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

- **Configure the cryptographic functionality,**
- **Configure the auditing functionality,**
- **manage user accounts,**
- **manage password constraints,**
- **manage SSL/TLS in FIPS mode configuration,**

- **manage access controls, and**
- **manage TOE sessions**].

5.2.5.8 Security roles (FMT_SMR.1)

FMT_SMR.1.1 The TSF shall maintain the roles: [**authorized administrator**].
FMT_SMR.1.2 The TSF shall be able to associate users with roles.

5.2.6 Protection of the TSF (FPT)

5.2.6.1 Protection of Administrator Passwords (FPT_APW_EXT.1)

FPT_APW_EXT.1.1 The TSF shall store passwords in non-plaintext form..

FPT_APW_EXT.1.2 The TSF shall prevent reading of plaintext passwords.

5.2.6.2 Protection of TSF Data (FPT_SKP_EXT.1)

FPT_SKP_EXT.1.1 The TSF shall prevent reading of [**all symmetric keys, and private keys**].

5.2.6.3 Basic internal TSF data transfer protection (FPT_ITT.1)

FPT_ITT.1.1 The TSF shall protect TSF data from **[*disclosure and modification*]** when it is transmitted between separate parts of the TOE.

5.2.6.4 Internal TSF consistency (FPT_TRC_EXT.1)

FPT_TRC_EXT.1.1 The TSF shall ensure that TSF data is consistent between parts of the TOE by providing a mechanism to bring inconsistent TSF data into a consistent state in a timely manner.

5.2.7 TOE Access (FTA)

5.2.7.1 Basic limitation on multiple concurrent sessions (FTA_MCS.1)

FTA_MCS.1.1 The TSF shall restrict the maximum number of concurrent sessions that belong to the same user.
FTA_MCS.1.2 The TSF shall enforce, by default, a limit of [***an admin configurable number of***] sessions per user.

5.2.7.2 TOE access history (FTA_TAH_EXT.1)

FTA_TAH_EXT.1.1 Upon successful session establishment, the TSF shall store and retrieve the [**date and time**] of the last successful session establishment to the user.
FTA_TAH_EXT.1.2 Upon successful session establishment, the TSF shall store and retrieve the [**date and time**] of the last unsuccessful attempt to session establishment and the number of unsuccessful attempts since the last successful session establishment.

5.2.7.3 TOE session establishment (FTA_TSE.1)

FTA_TSE.1.1 The TSF shall be able to deny session establishment based on [**attributes that can be set explicitly by authorized administrator(s), including user identity and/or role memberhsip, time of day, day of the week**], **and [application server privilege].**

5.2.8 Trusted path/channels (FTP)

5.2.8.1 Trusted Path (for administrators) (FTP_TRP.1(1))

FTP_TRP.1.1(1) The TSF shall **use *TLS/HTTPS* to** provide a **trusted** communication path between itself and **[*remote*] administrators** ~~users~~ that is logically distinct from other communication paths and provides assured identification of its end points and

protection of the communicated data from **[*disclosure [and detection of modification of the communicated data]*].**

FTP_TRP.1.2(1) The TSF shall permit [**remote administrative users]** to initiate communication via the trusted path.

FTP_TRP.1.3(1) The TSF shall require the use of the trusted path for [***initial administrator authentication [and all remote administration actions]***].

5.2.8.2 Trusted Path (for user) (FTP_TRP.1(2))

FTP_TRP.1.1(2) The TSF shall provide a communication path between itself and [***remote***] **non-administrator** users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from [***modification***, disclosure, ***[and detection of modification of the communicated data]***].

FTP_TRP.1.2(2) The TSF shall permit [***remote users***] to initiate communication via the trusted path.

FTP_TRP.1.3(2) The TSF shall require the use of the trusted path for [***[all services]***].

5.3 TOE Security Assurance Requirements

The security assurance requirements for the TOE are the EAL 2 augmented with ALC_FLR.3 components as specified in Part 3 of the Common Criteria. No operations are applied to the assurance components.

Requirement Class	Requirement Component
ADV: Development	ADV_ARC.1: Security architecture description
	ADV_FSP.2: Security-enforcing functional specification
	ADV_TDS.1: Basic design
AGD: Guidance documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures
ALC: Life-cycle support	ALC_CMC.2: Use of a CM system
	ALC_CMS.2 Parts of the TOE CM coverage
	ALC_DEL.1: Delivery procedures
	ALC_FLR.3: Systematic flaw remediation
ATE: Tests	ATE_COV.1: Evidence of coverage
	ATE_FUN.1: Functional testing
	ATE_IND.2: Independent testing - sample
AVA: Vulnerability assessment	AVA_VAN.2 Vulnerability analysis

Table 5-3 EAL2 Augmented with ALC_FLR.3 Assurance Components

5.3.1 Development (ADV)

5.3.1.1 Security architecture description (ADV_ARC.1)

ADV_ARC.1.1d The developer shall design and implement the TOE so that the security features of the TSF cannot be bypassed.

ADV_ARC.1.2d The developer shall design and implement the TSF so that it is able to protect itself from tampering by untrusted active entities.

ADV_ARC.1.3d The developer shall provide a security architecture description of the TSF.

ADV_ARC.1.1c The security architecture description shall be at a level of detail commensurate with the description of the SFR-enforcing abstractions described in the TOE design document.

ADV_ARC.1.2c The security architecture description shall describe the security domains maintained by the TSF consistently with the SFRs.

ADV_ARC.1.3c The security architecture description shall describe how the TSF initialization process is secure.

ADV_ARC.1.4c The security architecture description shall demonstrate that the TSF protects itself from tampering.

ADV_ARC.1.5c The security architecture description shall demonstrate that the TSF prevents bypass of the SFR-enforcing functionality.

ADV_ARC.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.1.2 Security-enforcing functional specification (ADV_FSP.2)

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

ADV_FSP.2.2d The developer shall provide a tracing from the functional specification to the SFRs.

ADV_FSP.2.1c The functional specification shall completely represent the TSF.

ADV_FSP.2.2c The functional specification shall describe the purpose and method of use for all TSFI.

ADV_FSP.2.3c The functional specification shall identify and describe all parameters associated with each TSFI.

ADV_FSP.2.4c For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions associated with the TSFI.

ADV_FSP.2.5c For each SFR-enforcing TSFI, the functional specification shall describe direct error messages resulting from processing associated with the SFR-enforcing actions.

ADV_FSP.2.6c The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

ADV_FSP.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.2.2e The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

5.3.1.3 Basic design (ADV_TDS.1)

ADV_TDS.1.1d The developer shall provide the design of the TOE.

ADV_TDS.1.2d The developer shall provide a mapping from the TSFI of the functional specification to the lowest level of decomposition available in the TOE design.

ADV_TDS.1.1c The design shall describe the structure of the TOE in terms of subsystems.

ADV_TDS.1.2c The design shall identify all subsystems of the TSF.

ADV_TDS.1.3c The design shall describe the behaviour of each SFR-supporting or SFR-non-interfering TSF subsystem in sufficient detail to determine that it is not SFR-enforcing.

ADV_TDS.1.4c The design shall summarise the SFR-enforcing behaviour of the SFR-enforcing subsystems.

ADV_TDS.1.5c The design shall provide a description of the interactions among SFR-enforcing subsystems of the TSF, and between the SFR-enforcing subsystems of the TSF and other subsystems of the TSF.

ADV_TDS.1.6c The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that they invoke.

ADV_TDS.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_TDS.1.2e The evaluator shall determine that the design is an accurate and complete instantiation of all security functional requirements.

5.3.2 Guidance documents (AGD)

5.3.2.1 Operational user guidance (AGD_OPE.1)

AGD_OPE.1.1d The developer shall provide operational user guidance.

AGD_OPE.1.1c The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2c The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3c The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4c The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5c The operational user guidance 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.

AGD_OPE.1.6c The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfil the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7c The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.2.2 **Preparative procedures (AGD_PRE.1)**

AGD_PRE.1.1d The developer shall provide the TOE including its preparative procedures.

AGD_PRE.1.1c The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2c The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

AGD_PRE.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2e The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

5.3.3 Life-cycle support (ALC)

5.3.3.1 **Use of a CM system (ALC_CMC.2)**

ALC_CMC.2.1d The developer shall provide the TOE and a reference for the TOE.

ALC_CMC.2.2d The developer shall provide the CM documentation.

ALC_CMC.2.3d The developer shall use a CM system.

ALC_CMC.2.1c The TOE shall be labelled with its unique reference.

ALC_CMC.2.2c The CM documentation shall describe the method used to uniquely identify the configuration items.

ALC_CMC.2.3c The CM system shall uniquely identify all configuration items.

ALC_CMC.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.3.2 **Parts of the TOE CM coverage (ALC_CMS.2)**

ALC_CMS.2.1d The developer shall provide a configuration list for the TOE.

ALC_CMS.2.1c The configuration list shall include the following: The TOE itself; the evaluation evidence required by the SARs; and the parts that comprise the TOE.

ALC_CMS.2.2c The configuration list shall uniquely identify the configuration items.

ALC_CMS.2.3c For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

ALC_CMS.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.3.3 **Delivery procedures (ALC_DEL.1)**

ALC_DEL.1.1d The developer shall document and provide procedures for delivery of the TOE or parts of it to the consumer.

ALC_DEL.1.2d The developer shall use the delivery procedures.

ALC_DEL.1.1c The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to the consumer.

ALC_DEL.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.3.4 **Systematic flaw remediation (ALC_FLR.3)**

ALC_FLR.3.1d The developer shall document and provide flaw remediation procedures addressed to TOE developers.

ALC_FLR.3.2d The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.

ALC_FLR.3.3d The developer shall provide flaw remediation guidance addressed to TOE users.

ALC_FLR.3.1c The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.3.2c The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.3.3c The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.3.4c The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.3.5c The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ALC_FLR.3.6c The flaw remediation procedures shall include a procedure requiring timely response and the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw.

ALC_FLR.3.7c The procedures for processing reported security flaws shall ensure that any reported flaws are remediated and the remediation procedures issued to TOE users.

ALC_FLR.3.8c The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.3.9c The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

ALC_FLR.3.10c The flaw remediation guidance shall describe a means by which TOE users may register with the developer, to be eligible to receive security flaw reports and corrections.

ALC_FLR.3.11c The flaw remediation guidance shall identify the specific points of contact for all reports and enquiries about security issues involving the TOE.

ALC_FLR.3.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.4 Tests (ATE)

5.3.4.1 **Evidence of coverage (ATE_COV.1)**

ATE_COV.1.1d The developer shall provide evidence of the test coverage.

ATE_COV.1.1c The evidence of the test coverage shall show the correspondence between the tests in the test documentation and the TSFIs in the functional specification.

ATE_COV.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.4.2 **Functional testing (ATE_FUN.1)**

ATE_FUN.1.1d The developer shall test the TSF and document the results.

ATE_FUN.1.2d The developer shall provide test documentation.

ATE_FUN.1.1c The test documentation shall consist of test plans, expected test results and actual test results.

ATE_FUN.1.2c The test plans shall identify the tests to be performed and describe the scenarios for performing each test. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.1.3c The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.1.4c The actual test results shall be consistent with the expected test results.

ATE_FUN.1.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.3.4.3 **Independent testing - sample (ATE_IND.2)**

ATE_IND.**2.1d** The developer shall provide the TOE for testing.

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.

ATE_IND.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2e The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

ATE_IND.2.3e The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5.3.5 Vulnerability assessment (AVA)

5.3.5.1 **Vulnerability analysis (AVA_VAN.2)**

AVA_VAN.2.1d The developer shall provide the TOE for testing.

AVA_VAN.2.1c The TOE shall be suitable for testing.

AVA_VAN.2.1e The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.2.2e The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.2.3e The evaluator shall perform an independent vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design and security architecture description to identify potential vulnerabilities in the TOE.

AVA_VAN.2.4e The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

6 TOE Summary Specification

This chapter describes the following security functions:

- Security audit
- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Protection of the TSF
- TOE Access

6.1 Security audit

The TOE generates audit records for the following auditable events:

- Start-up and shutdown of the DBMS,
- All administrative actions,
- Successful use of an amp,
- All auditable events as specified in the table below:

Requirement	Auditable Events	Additional Audit Record Contents
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	The identity of the authorized administrator that made the change to the audit configuration.
FDP_ACF.1	All requests to perform an operation on an object covered by the SFP.	The identity of the object and the subject performing the operation.
FIA_UAU.2	All use of the authentication mechanisms	Provided user identity, origin of the attempt (e.g., IP address).
FIA_UID.2	All use of the user identification mechanism, including the user identity provided.	The user identity provided.
FMT_REV.1(1)	Unsuccessful revocation of security attributes.	Identity of individual attempting to revoke security attributes.
FMT_REV.1(2)	Unsuccessful revocation of security attributes.	Identity of individual attempting to revoke security attributes.
FMT_SMF.1	Use of the management functions.	Identity of the administrator performing these functions.
FMT_SMR.1	Modifications to the group of users that are part of a role.	Identity of authorized administrator modifying the role definition.
FTA_MCS.1	Rejection of a new session based on the limitation of multiple concurrent sessions.	None
FTA_TSE.1	Denial of a session establishment due to the session establishment mechanism.	Identity of the individual attempting to establish the session.

Each audit record will include the date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event. In some cases, auditing can be configured to audit successful, unsuccessful, or both types of events; however, some events specifically audit either the success or failure of the event. The Operating System in the environment provides protection, storage and the ability to view the audit records. The OS administrator can configure conditions under which the TOE start using a “new” audit log file with values such as Never, day of the week (Mon-Sun), or monthly. The OS administrator can also specify how many log files to keep.

Thus, the operating system in the environment is responsible for providing sufficient storage space to hold TOE audit data.

The Operating System in the environment also provides the system clock information that is used by the TOE to timestamp each audit record. The audit records are stored on the local file system of the host. In a multi-host distributed architecture, where the Server subsystem of the TOE is run on a number of hosts, the audit records are stored on the local file system of the host on which the related auditable event is detected. Consequently, the aggregate audit record for an entire cluster may be distributed across multiple hosts, rather than being stored in a single location.

The TOE provides the Admin Interface, a web based browser GUI, through which an authorized administrator has the ability to configure the audit function to include or exclude auditable events based on user identity, role, event type, object identity and success or failure of the auditable security event. The Admin API provides XQuery functions for managing audit settings.

Since the TOE is an application running within a process context, it relies upon the environment to provide typical operating system services including protected data storage. The TOE relies upon the environment to provide a mechanism (e.g., files) which allows the TOE to store information. The TOE relies upon the environment to prevent unauthorized modification or deletion of this stored TOE data.

The TOE offers the ability to start and stop the audit mechanism independently from the starting and stopping of the entire DBMS. The TOE is capable of generating audit events recording the starting and stopping of the DBMS or the starting and stopping of the audit mechanism.

The Security audit function is designed to satisfy the following security functional requirements:

- FAU_GEN.1: Audit records are generated for the appropriate security relevant events and include the date and time of the event, type of event, subject identity (if applicable) and outcome of the event.
- FAU_GEN.2: The TOE associates each auditable event resulting from actions of identified users with the identity of the user that caused the event.
- FAU_SEL.1: The TOE allows administrators to include or exclude auditable events based on user identity, role, event type, object identity and success and failure of auditable security events.

6.2 Cryptographic support

The TOE uses cryptography to support the protection of the following types of communication pathways:

- Administrative login and management sessions,
- TOE to TOE communication, and
- Customer application to TOE sessions

An administrative management session is initiated by a login and occurs only over HTTPS using SSL/TLS in FIPS mode (TLS version 1.0). The TOE does not distinguish between an administrator that is on a local host or a remote network host. All administrator connectivity is handled like a remote session and requires the use of HTTPS over SSL/TLS in FIPS mode. A remote administrative session occurs using a GUI provided by the TOE HTTP server using HTTPS. TOE to TOE communication occurs for the purpose of propagating TSF data from one instance of the TOE to another. Each instance of the TOE ensures that such communication occurs only over a SSL/TLS in FIPS mode protected communication pathway.

Additionally, management functions may be performed on an App Server that runs the Admin API, Security API, or PKI API. Any App Server where Admin API, Security API, or PKI API functions are run protects sessions with SSL/TLS in FIPS mode.

The TOE provides the capability for customer applications (i.e., non-administrative user's) to communicate with the TOE from network hosts through SSL/TLS in FIPS mode protected communication pathways. Such communication allows a customer application to communicate with either the HTTP server, ODBC server, or XDBC server of the TOE.

The TOE uses the same implementation of SSL/TLS in FIPS mode for each of these communication pathways. SSL/TLS in FIPS mode provides protection of the communications pathways from disclosure and from modification.

The TOE uses a FIPS capable OpenSSL object module consisting of OpenSSL version 1.0.1b with OpenSSL FIPS Object Module v2.0, which has undergone a FIPS 140-2 validation. See certificate #1747 (http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm#1747). The TOE includes the FIPS object module built without modification from the source code that has undergone FIPS validation. All references to "the TOE" performing cryptographic operations in this section are indicating that the TOE is performing the operation through its use of the OpenSSL FIPS object module.

The TOE uses the FIPS object module implementation of AES to support the HTTPS/TLS protocol. The TOE also uses ECDSA from the FIPS object module as part of the verification and use of certificates. Refer to Table 6-1 OpenSSL FIPS Object Module Certificates for references to the specific FIPS certificate number covering these algorithms as well as algorithms described below for cryptographic hash, keyed hash, and signature services.

Algorithm	FIPS Certification Number
AES	1884
ECDSA	264
rDSA	960
SHS	1655
HMAC	1126

Table 6-1 OpenSSL FIPS Object Module Certificates

The TOE includes OpenSSL which implements the SSL/TLS protocol (this is OpenSSL version 1.0.1b). The TOE is required to run in FIPS mode. This OpenSSL object module is distinct from the OpenSSL FIPS object module. The FIPS Object Module is a special monolithic object module built from the special source distribution identified in its Security Policy. It is not the same as the OpenSSL product or any specific official OpenSSL distribution release. A version of the OpenSSL product that is suitable for reference by an application along with the FIPS Object Module is a FIPS compatible OpenSSL. When the FIPS Object Module and a FIPS compatible OpenSSL are separately built and installed on a system, the combination is referred to as a FIPS capable OpenSSL. The TOE includes a FIPS capable OpenSSL.

The TOE generates random numbers in a manner that is consistent with FIPS 140-2 for use in the generation of cryptographic keys with bit sizes from 128 bits to 256 bits.

The TOE implements the AES algorithm as defined by FIPS PUB 197 and consistent with NISP SP 800-38A. The TOE uses AES for encryption and decryption of data as part of the support for the SSH and TLS protocols. The TOE uses AES in CBC (cipher-block chaining) mode. The TOE supports the use of 128-bit and 256-bit AES keys.

The TOE also provides cryptographic hashing services using the SHA-1 and SHA-256 algorithms as defined by FIPS Pub 180-3 'Secure Hash Standard'. The TOE supports message digest sizes of 160-bits and 256 bits for this hashing service. These cryptographic hashing services are used by the TOE implementation of TLSv1.0.

The TOE provides keyed-hash authentication using HMAC-SHA-1 and HMAC-SHA-256 with a keys size and message digest sizes of 160-bits and 256-bits respectively. The TOE implementation of HMAC-SHA-1 is built to meet FIPS Pub 198-1 and FIPS Pub 180-3. These crypto keyed-hash authentication services are used by the TOE implementation of TLSv1.0.

Finally, when the TOE erases a plaintext secret and private key from disk that is no longer need, it overwrites the storage space used by that key with zeros.

The TOE implements HTTPS as specified by RFC 2818. The TOE does not support HTTP connections for administration. The TOE implements TLS version 1.0 as specified by RFC 2246 using the following ciphersuites.

- TLS_RSA_WITH_AES_128_CBC_SHA

- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

The Cryptographic support function is designed to satisfy the following security functional requirements:

- FCS_CKM.1: The TOE generates asymmetric cryptographic keys for use in key establishment. These keys meet the recommendations of SP 800-56A for elliptic curve-based key establishment schemes. The TOE also implements ECDSA using key sizes of 256-bits or greater and implements support for all NIST 'P' curves (including P-256, P-384 and P-521).
- FCS_CKM_EXT.4: The TOE clears, by overwriting with zeros, plaintext secret and private keys when no longer needed.
- FCS_COP.1(1): The TOE implements AES for encryption and decryption of data as described above to meet FIPS PUB 197 and NISP SP 800-38A with the bit sizes and mode described above and in the OpenSSL security policy.
- FCS_COP.1(2): The TOE also implements ECDSA using key sizes of 256-bits or greater. The TOE implementation of ECDSA both meet FIPS 186-3. The TOE implementation of ECDSA meets all NIST defined P curves with all SHA sizes.
- FCS_COP.1(3): The TOE implements SHA-1 and SHA-256 for hashing services as described above to meet FIPS Pub 180-3 with the required message digest sizes.
- FCS_COP.1(4): The TOE implements HMAC-SHA-1 and HMAC-SHA-256 for keyed-hash authentication as described above to meet FIPS Pub 198-1 and FIPS Pub 180-3 with the required key sizes.
- FCS_COP.1(5): The TOE implements rDSA using key sizes of 2048 bits or greater. The TOE implementation of RSA Digital Signature Algorithm meets FIPS 186-3 by using the OpenSSL FIPS object module for rDSA cryptographic signature operations.
- FCS_HTTPS_EXT.1: The TOE implements HTTPS as described in the text above. The TOE uses TLS version 1.0 as cryptographic protection for HTTPS.
- FCS_TLS_EXT.1: The TOE implements TLS Version 1.0 protocol which is used as described in the text above. The TOE implementation of TLS provides the ciphersuites listed above.
- FTP_TRP.1(2): The TOE provides SSL/TLS in FIPS mode support for secure communication between non-administrative users (that is, customer application) and the HTTP, XDBC, and ODBC servers of the TOE. Non-administrative users are not required to use trusted communication channels.

6.3 User data protection

The TOE enforces a Discretionary Access Control (DAC) Policy on all subjects, all Database Management System (DBMS) controlled objects and all operations among them. The DBMS controlled objects implemented by the TOE are *documents*. The operations are create a document and capabilities (that is, read, update, insert, and execute). Documents are made up of one or more of the following:

- Content – XML, character, or binary content stored in a TOE database.
- Properties – XML describing the document properties.
- Locks – System-maintained XML describing the document locks. A lock can be exclusive or shared and can prevent update or deletion of content by other users.
- Other metadata – For example, collections and document permissions.

Documents can be organized into *collections*, which are groups of related documents that enable queries to target subsets of content within the TOE. A collection is either unprotected or protected. A document may belong to any number of collections simultaneously. An unprotected collection is implicitly created and exists in the system when a document in the system states that it is part of that collection. A protected collection is explicitly created using the Admin Interface. Collections are not related to directories. They do not require member documents to conform to any

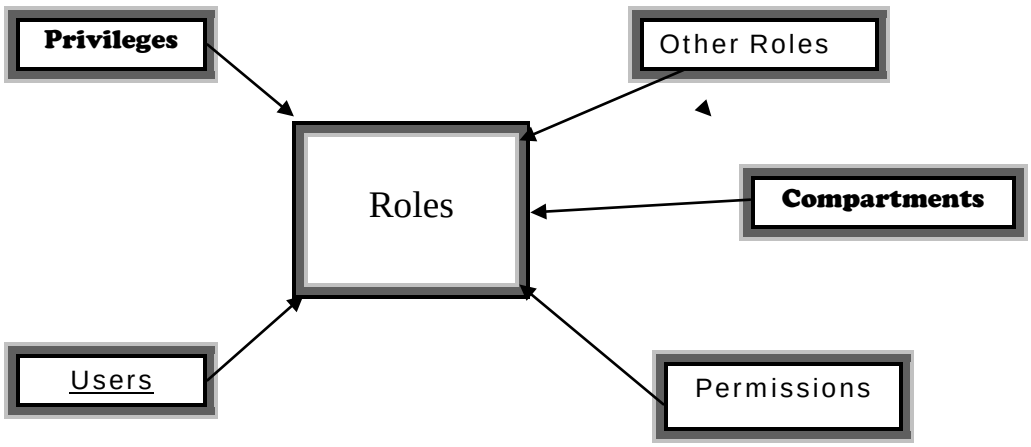
URI patterns, they are not hierarchical and they cannot have properties set on them. The URIs that are used to name collections serve only as identifiers to the server. Unprotected collections do not have any security attributes associated to them; therefore the access control policy for them is the access control policy for the individual documents that are part of the collection. Access to each of the individual documents that belong to the specified collection is governed by that individual document's permissions. An authorized administrator associates permissions with a protected collection using either the Admin Interface or Security API. In order to access a document that belongs to a protected collection, a user must have role(s) to satisfy both document permissions and protected collection permissions.

The DAC policy restricts access operations implemented for documents based on the document's identity (its Uniform Resource Identifier (URI)) and the user's authorized role membership. Users of the TOE are identified and authenticated by the TOE before any access to the system is granted. Once access to the system is granted, authorization provides the mechanism to control what functions a user is allowed to perform based on the user's role membership.

Access to all documents is denied unless access, based on role membership, is explicitly allowed. Documents are assigned permissions which are a combination of a role and a capability. Each permission associates a role with one of the following capabilities: Read, Update, Insert and Execute. Users assigned the role corresponding to the permission have the ability to perform the capability.

There are two types of privileges: Uniform Resource Identifier (URI) privileges and Execute privileges. URI privileges are used to control the creation of documents with certain URIs. Execute privileges are used to protect the execution of functions in XQuery code and to protect access to specific application servers.

Roles are the central point of authorization in the TOE. As shown in the diagram below, privileges (both execute and URI), permissions, compartments, users and roles are all assigned to zero or more roles.



** Permissions = Role + Capability (read, insert, update, executes).

The DAC policy restricts access to documents based on roles, which may entail protected collections and Compartment Security compartments. A compartment is a name associated with a role, and when a role is compartmented, documents that have this role (with a capability) require an additional check to pass before the capability is allowed. By default, the DAC policy allows a user a capability to a document when the user is a member of a role specified as part of a permission for the document. If the document is in a protected collection, then the user also must be a member of a role specified as part of a permission for the protected collection. If document permissions are paired with a compartmented role, the user also must be assigned those roles specified (for each permission paired with a compartmented role) in order to perform the permission's capability (read, insert, update, or execute) on the document. The DAC restricts document creation to a user with an Execute Privilege (any-uri or unprotected-uri) as well as URI privileges for URI prefixes of the document's URI. See section 6.5 regarding default document permissions for documents.

Authorized administrators with the `admin` role have explicitly authorized access to all documents. Additionally, the TOE provides amplifications (referred to as *amps*) which allow users to assume additional privileges and permissions through temporary assumption of additional user roles during the execution of specified XQuery library functions. Amps can therefore be used to temporarily grant administrator privileged role to an unprivileged user, thereby extending the DAC policy while performing a specific functions. The effect of any additional permissions and privileges is limited to the specific function. Amps can only be configured and assigned by authorized administrators via access to the Admin Interface and the Security API. Additionally, amplified functions are only located in either a designated administrator-controlled location in a directory on the MarkLogic Server Subsystem or in the database where they would be subject to the DAC policy and no user would have the ability to update or modify the function.

The TOE also ensures that any previous information content of a resource is made unavailable upon the allocation of the resource to TOE objects. Memory or disk space is only allocated when the size of the new data is first known, so that all previous data is overwritten by the new data.

The User data protection function is designed to satisfy the following security functional requirements:

- FDP_ACC.1: The TOE will enforce the DAC policy on all subjects, all documents and all operations among them.
- FDP_ACF.1: The TOE will enforce the DAC policy on documents based on the authorized user's role membership, the object identity and the access operations implemented for the documents. Documents will be protected from unauthorized access according to a set of ordered rules.
- FDP_RIP.2: The TOE will ensure that any previous information content of a resource is made unavailable upon the allocation of the resource to document objects.

6.4 Identification and authentication

The TOE maintains user accounts for the authorized users of the system and a list of security attributes for each user which includes the user's identifier, role membership, and password. The TOE maintains the security relevant database role of authorized administrator. Authorized administrators are the only users that have privileges to manage the TOE security functions as described in this Security Target.

The TOE requires users to provide unique identification and authentication data (ie. passwords) before any access to the system is granted. The TOE uses the digest authentication scheme, a commonly used web application authentication protocol, to provide encryption for passwords which are sent across the network as an MD5 hash using this scheme.[3] Digest authentication uses the browser's username and password prompt to obtain user credentials. The MarkLogic Server subsystem then authenticates the user credentials against the security database. Authentication simply verifies user credentials, associates that session with the authenticated user and determines their role membership. It does not grant any access or authority to perform any actions on the system. When a user logs into the TOE, their user id and password are validated against the security database.

All security attributes are stored in the security database of the MarkLogic Server subsystem. A single security database is associated with each HTTP, XDBC, or ODBC server. Where the TOE is configured with multiple servers, the same security database can be associated with the server or servers regardless of the number. The security database is accessed to authenticate users and to control user actions against the server.

Once access to the system is granted, authorization to access functions and data is implemented via the user's role membership. User roles are the central point of authorization in the TOE's security model. User roles are created with a specific set of privileges and permissions which apply to all users assigned to the role.

The TOE provides a password plug-in functionality that allows administrators to write custom code to require passwords to conform to specific rules (e.g., the number of characters, special characters, last change date). The TOE invokes password plug-in when a user's password is set or modified through the Admin Interface or Security API. The special characters shown in Table 6-2 are all available as characters in a password policy. The password policy plug-in can also be used (by writing code in the form of a password plug-in) to require a minimum password length and to support passwords of up to whatever limit the administrator implements within the plug-in. By default, MarkLogic does not require any particular strength for passwords. The actual plug-in is custom code and is not subject to

[3] For further information on digest authentication, please refer to RFC 2617. All Application Servers in the evaluated configuration must use digest based authentication.

evaluation; the only aspects that will be tested are those that cover special characters shown in Table 6-2 (required by the password management SFR).

Password Special Characters			
!	#	$	%
^	&	*	(
)	@		

Table 6-2 Special Characters available for Passwords

The Identification and authentication function is designed to satisfy the following security functional requirements:

- FIA_ATD.1: The TOE maintains a list of security attributes for individual users.
- FIA_UAU.2: The TOE requires all users to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.
- FIA_UID.2: The TOE requires all users to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.
- FIA_PMG_EXT.1: The password plug-in functionality can be used to define a policy that ensures that passwords are composed of any combination of upper and lower case letters, numbers and special characters mentioned in Table 6-2.

6.5 Security management

The ST defines the concept of an authorized administrator to distinguish users who can perform security management functions from normal database users. An authorized administrator is a user that performs security management functions. The TOE protects security management functions with privileges, and any user that has any of the privileges required to perform any security management function is considered an authorized administrator. Users with the admin role can perform any security management function. The functions in the Admin API, the Security API, the PKI API, or the built-in Admin XQuery functions, are protected by privileges; in order to evaluate them, a user must have privileges for each function that is evaluated, otherwise a security exception is thrown. Any user that has any of the privileges required to run any of the functions in these libraries is therefore considered an authorized administrator. Upon installation, multiple roles are installed into the TOE. These fine-grained product roles provide additional utility, and in some cases, allow access to certain security management functions (which would make users granted such a role authorized administrators). The only pre-defined administrative role assigned to a user at installation is the `admin` role. The other roles are defined with a default set of privileges, however, no users are assigned those roles until an authorized administrator assigns any of them to a user. The use of any of the other roles is optional. However, the ST makes no claim about further role separation within the ST authorized administrator role. As stated in *MarkLogic Common Criteria Evaluated Configuration Guide*, any user that has any of the privileges needed to run functions in the Admin API, the Security API, the PKI API, or the built-in Admin XQuery functions is considered an authorized administrator because it is possible for them to write and evaluate code that might perform a security management function.

Only authorized administrators can perform TOE security management functions. The built-in product role admin corresponds directly to the ST authorized administrator role. Other built-in product roles (such as the security role) and privileges provide partial access to security management functions and are considered part of the ST authorized administrator role. An authorized administrator can define roles and grant privileges to them. An administrator-defined role that is granted privileges for security management functions is considered part to the authorized administrator role.

The Admin Interface provides the interface through which the authorized administrator manages the security functions of the TOE. The Admin Interface provides administrator access to the following TOE security management functions:

- Management of User Accounts
  - Create, view, delete, and modify user accounts, including revoking security attributes associated with users.

    - Create, view, delete and modify privileges
    - Create, view, delete and modify user roles.
- Manage Cryptographic configuration
    - Configure the algorithm and key sizes used by SSL/TLS in FIPS mode
    - Configure the certificates used by SSL/TLS in FIPS mode for the HTTPS servers
- Configure the auditing functionality
    - Enable and disable the audit configuration function.
    - Configure the audit function to include or exclude auditable events.
- Management of Access Control
    - Create, view and delete amps.
    - Create, query, modify or delete all the user and DBMS-controlled object security attributes associated with the DAC policy.
- Management of TOE sessions
    - Configure the limit on maximum number of concurrent sessions belonging to the individual user.
    - Configure the rules for denying session establishment.

The Admin API, Security API, and PKI provide XQuery functions which can be used to write code which performs security management function. For example, the Admin Interface uses these functions, and with the XQuery functions, it is possible to write an application that provides the same functionality for managing security management functions as the Admin Interface. The APIs provide low-level programmatic access, so it is not the APIs themselves that provide the SFRs, rather it is programs that use the APIs that provide the SFRs. Roughly speaking, the following shows the XQuery libraries that one might use to provide TOE security management functions:

Management of User Accounts	Security API
Manage Cryptographic Configuration	PKI API
Configure Auditing	Admin API
Manage Access Control (roles, amps)	Security API
Management of TOE Sessions	Admin API
Management of TOE that involves server actions such as restart, shutdown, and so on.	Admin Built-Ins

The TOE provides a programmatic solution which the administrator can use to implement composition rules for passwords (that is, manage password constraints). An authorized administrator manages password composition rules through the Security API.

The TOE provides administrators with the ability to revoke security attributes associated with users and objects. User security attributes are role membership and password. DBMS-controlled object (that is , document) security attributes are the access operations, or permissions that are implemented for the document.

Revocation of both object and user security attributes is enforced at all TOE interfaces based on the following rules:

- On the *revocation host*, revocation is effective on the next session that starts after the revocation request is committed.
- On other hosts in a cluster, revocation is effective no later than the receipt of the next *heartbeat* received from the *revocation host*.

The revocation hosts for object and user security attributes are different. The revocation host associated with revocation of user security attributes is the host on which the security forest resides. The revocation host associated with revocation of document security attributes is the host on which the document resides locally. A heartbeat is a cluster synchronization message and occurs once per second.

When a document is created within the TOE, the document is initialized with a set of permissions. If permissions are not explicitly set during creation, then the TOE applies default permissions. The default permissions are determined

based on the roles assigned (both assigned explicitly and inherited from roles assigned to other roles) to the user who creates the document and on any default permissions assigned directly to the user. If users will be creating document in a database, it is very important to set up default permissions for roles to which that user is assigned. Without default permissions, it is easy to create document that no users (except those who are part of the admin role) can read, update, or delete.

The Security management function is designed to satisfy the following security functional requirements:

- FMT_MSA.1: The TOE enforces the DAC policy to restrict the ability to manage the security attributes to authorized administrators (with the `admin` role or the appropriate privileges).
- FMT_MSA.3: The TOE enforces the DAC policy to provide restrictive default values for security attributes.
- FMT_MTD.1(1): The TOE ensures that only "Authorized Administrator" (with the `admin` role or the appropriate privileges) can configure the TSF through the modification of TSF configuration data.
- FMT_MTD.1(2): The TOE restricts the ability to include and exclude auditable events to authorized administrators (with the `admin` role or the appropriate privileges).
- FMT_REV.1(1): Only TOE authorized administrators (with the `admin` role or the appropriate privileges) can revoke user security attributes according to enforceable rules.
- FMT_REV.1(2): Only TOE authorized administrators (with the `admin` role or the appropriate privileges) can revoke object security attributes according to enforceable rules.
- FMT_SMF.1: The TOE provides management functions identified in the text above to support an administrator's ability to securely install, configure and operate the system as described in the above section.
- FMT_SMR.1: The TOE maintains the security role of authorized administrator.

6.6 Protection of the TSF

The TOE provides security mechanisms for its security functions to ensure that it can protect itself from tampering and bypass by untrusted entities. One of the protection mechanisms is that users must authenticate before any administrative operations can be performed on the system. The TSF requires that all users be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that user. The TOE also enforces an access control policy which restricts user access to DBMS-controlled objects. Authorized users only have access to functions as specified by their assigned role membership and capabilities. The Operating System in the environment of the TOE provides an execution environment that ensures thread separation and safeguards the results of one query from interfering with the results of another query. The TOE also relies on the IT environment for process isolation.

The TOE also has the ability to replicate data by propagating updated configuration and security files throughout a cluster. Configuration information includes the Cluster, Host, Cluster Management Group, Forest, and Database information as described in Section 2.2 above. The TOE ensures the consistency of TSF data between parts of the TOE for both configuration information and security information as follows:

- The TOE's configuration information is stored in a set of files in a special file system directory structure on each host in a cluster. For example, on Linux, the default location for configuration files is /var/opt/MarkLogic. Each configuration file contains a configuration file system timestamp which is a monotonically-increasing number that increases with every configuration or content change cluster-wide. The *configuration file system timestamp* is the latest timestamp at the time the file was last updated. Each heartbeat, or cluster synchronization message, that occurs once per second, contains the *heartbeat configuration system timestamp* which is the most recent timestamp of the configuration files of the host from which it was issued. Within one second of receipt of a heartbeat, the receiving host examines the heartbeat configuration system timestamp and if it is more recent than its own, the newer configuration files from the host that issued the heartbeat are copied and the local configuration files are replaced with the newer versions.
- The TOE's security data is stored in the security database. There is one security database per cluster and other hosts in the cluster cache some of the documents in this database to the security database cache. There is a timestamp for both the security database and the security database cache which indicates the time of the

most recent change to the database or database cache. Each heartbeat also contains a copy of the security timestamp. Upon receipt of a heartbeat, if that heartbeat contains a security timestamp more recent than the security timestamp on the receiving host's security database cache, then the receiving host's database cache is invalidated. Consequently, all sessions initiated on that host subsequent to the security database cache flush will be forced to retrieve the latest copies of documents from the security database.

The TOE utilizes an HTTPS connection for administrators to authenticate to the TOE from a browser that is part of the environment. This initial authentication action occurs over a SSL/TLS in FIPS mode connection negotiated using the ciphers defined as valid for a SSL/TLS in FIPS mode session as described in section 6.2. The SSL/TLS in FIPS mode connection protects communication between the TOE and the administrator's browser session from disclosure. SSL/TLS in FIPS mode also support the detection of modification of the communicated data between the TOE and the administrator's browser session.

The Protection of the TSF function is designed to satisfy the following security functional and assurance requirements:

- FPT_APW_EXT.1: The TOE does not offer an interface to query existing passwords and stores passwords internally in a hashed form[4].
- FPT_SKP_EXT.1: The TOE does not offer an interface to query symmetric or private keys.
- FPT_ITT.1: The TOE utilizes TLS to protect data transmitted between distributed parts of the TOE during the propagation of TSF data.
- FPT_TRC_EXT.1: The TOE ensures that TSF data is consistent between parts of the TOE by providing the mechanism described above to bring inconsistent TSF data into a consistent state in a timely manner.
- FTP_TRP.1(1): Administrators connect to the TOE using HTTPS/TLS to use the administrative GUI for management of the TOE. The initial administrator authentication operation as well as all subsequent remote administration actions occur through this HTTPS channel over SSL/TLS in FIPS mode.

6.7 TOE Access

The TOE restricts the maximum number of concurrent sessions that belong to the same user. This is enforced by the setting of an administrator configurable number of sessions per user. Upon successful session establishment, the TOE will store and retrieve the date and time of the last successful session establishment, the date and time of the last unsuccessful attempt to session establishment and the number of unsuccessful attempts since the last successful session establishment to the user. TOE session establishment history and data is stored in the last-login database of the TOE and is persisted indefinitely. Session establishment data is maintained on a per user basis across the entire cluster. Therefore, within a given cluster, session establishment data for any hosts that have been previously accessed by a user will be reported to the user from any other hosts subsequently accessed within the same cluster.

The TOE provides session establishment control and can deny session establishment based on either user identity or role membership, or time of the day or day of the week or some combination thereof. Authorized administrators can configure the session establishment rules via the Admin Interface. Rules for session denial are configured for each application server (that is, HTTP, XDBC, and ODBC). Session establishment may also be denied if the user does not have the application server privilege required to establish a session on the application server to which the user is attempting to connect.

Application server privileges allow you to specify an execute privilege, which is one of the privileges stored in the Security database, to control access to an App Server (HTTP, XDBC, ODBC). The privilege can be either a MarkLogic predefined privilege or one defined by the authorized administrator. The Admin Interface provides the option of specifying that an application server privilege is required for server access. If such a privilege is set on an App Server, access to that server is granted only to users that possess that privilege. Users have privileges based upon the roles the user is assigned. So, a privilege is granted to a role, users with that role have the privileges of the role.

The TOE stores information about unauthorized login attempts and the number of times the login was attempted every time the user logs into their account. The TOE also stores information about the last successful authorized login. This information includes the date, time, and user id of the attempts. The stored information can be used to create an

[4] The TOE does not utilize the OpenSSL cryptographic hash mechanism for storage of passwords internally.

application that analyzes failed login attempts. As an example of such an application, MarkLogic includes a simple example of this functionality and the Admin Interface can be configured to display the date and time of the last successful login, the last unsuccessful login, and the number of intervening unsuccessful logins for authorized administrators. The TOE stores this information in the Last-Login database, which is accessible only through the TOE.

The TOE access function is designed to satisfy the following security functional requirements:

- FTA_MCS.1: The TOE will restrict the maximum number of concurrent sessions that belong to a user by enforcing an administrator configurable limit on the number of sessions per user.
- FTA_TAH_EXT.1: Upon successful session establishment, the TOE stores and retrieves for the user, the date and time of the last successful session establishment, the last unsuccessful attempt to session establishment and the number of unsuccessful attempts since the last successful session establishment.
- FTA_TSE.1: The TOE denies session establishment based on user identity or role membership, or time of day, or day of week or by application server privilege or a combination thereof.

7 Protection Profile Claims

There are no Protection Profile claims in this Security Target.

8 Rationale

This section provides the rationale for completeness and consistency of the Security Target. The rationale addresses the following areas:

- Security Objectives;
- Security Functional Requirements;
- Security Assurance Requirements;
- Requirement Dependencies;
- TOE Summary Specification.

8.1 Security Objectives Rationale

This section shows that all secure usage assumptions, organizational security policies, and threats are completely covered by security objectives. In addition, each objective counters or addresses at least one assumption, organizational security policy, or threat.

8.1.1 Security Objectives Rationale for the TOE and Environment

This section provides evidence demonstrating the coverage of organizational policies and usage assumptions by the security objectives.

	T.TSF_COMPROMISE	T. UNAUTHORIZED_ACCESS	T. UNDETECTED_ACTIONS	A.AUTH	A.CLIENT	A.NO_EVIL	A.NO_GENERAL_PURPOSE	A.OS_TIME	A.PHYSICAL	A.TRUSTED_OS
O.ACCESS_HISTORY			X							
O.AUDIT_GENERATION		X	X							
O.MANAGE	X									
O.MEDIATE		X								
O.PROTECTED_COMMUNIATIONS		X								
O.RESIDUAL_INFORMATION	X									
O.TOE_ACCESS		X								
OE.AUTH				X						
OE.CLIENT					X	X				
OE.NO_GENERAL_PURPOSE							X			
OE.PHYSICAL									X	
OE.PROCESS										X
OE.STORAGE										X
OE.TIME								X		
OE.TRUSTED_ADMIN						X				

8.1.1.1 T.TSF_COMPROMISE

A user may cause, through an unsophisticated attack, TSF data, or executable code to be inappropriately accessed (viewed, modified, or deleted).

This Threat is countered by ensuring that:

O.RESIDUAL_INFORMATION: This objective is necessary to mitigate this threat, because even if the security mechanisms do not allow a user to view TSF data, if TSF data were to reside inappropriately in a resource that was made available to a user, that user would be able to view the TSF data without authorization.

O.MANAGE: This objective is necessary because an access control policy is specified to control access to TSF data. This objective is used to dictate who is able to view and modify TSF data, as well as the behavior of TSF functions.

8.1.1.2 T.UNAUTHORIZED_ACCESS

A user may gain unauthorized access to the TSF data and TSF executable code. A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to TSF data or TSF resources. A malicious user, process, or external IT entity may misrepresent itself as the TSF to obtain identification and authentication data.

This threat is satisfied by ensuring that:

- O.PROTECTED_COMMUNICATIONS: To reduce the potential that an attacker might gain unauthorized access to the TOE or its data via data transmitted across a network, the TOE is expected to protect its administrator communication channels from disclosure, modification, and also to ensure the identity of the TSF. The TOE supports protection of non-administrator communication channels at the discretion of the remote user.
- O. AUDIT_GENERATION: To reduce the potential of unauthorized access attempts that might go unnoticed, the TOE is expected to log security relevant events and export those logs to an external log server.
- O.TOE_ACCESS: To reduce the potential of unauthorized access to TOE security functions and data, the TOE is expected to be designed to ensure that only presumably authorized administrators can log in and access security management functions.
- O.MEDIATE: This objective ensures that all accesses to user data are subject to mediation, unless said data has been specifically identifies as public data. The TOE requires successful authentication to the TOE prior to gaining access to any controlled-access content. By implementing strong authentication to gain access to these services, an attacker's opportunity to conduct a man-in-the-middle and/or password guessing attack successfully is greatly reduced. Lastly, the TSF will ensure that all configured enforcement functions (authentication, access control rules, etc.) must be invoked prior to allowing a user to gain access to TOE or TOE mediated services. The TOE restricts the ability to modify the security attributes associated with access control rules, access to authenticated and unauthenticated services, etc to the administrator. This feature ensures that no other user can modify the information flow policy to bypass the intended TOE security policy.

8.1.1.3 T.UNDETECTED_ACTIONS

Malicious remote users or external IT entities may take actions that adversely affect the security of the TOE. These actions may remain undetected and thus their effects cannot be effectively mitigated.

This threat is satisfied by ensuring that:

- O.ACCESS_HISTORY: This objective is important to mitigate this threat because it ensures the TOE will be able to store and retrieve the information that will advise the user of the last successful login attempt and performed actions without their knowledge. Given that this information is provided to legitimate users, those users can support the detection of unauthorized activity.
- O.AUDIT_GENERATION: To reduce the potential of security relevant actions occurring without notice, the TOE is expected to audit security relevant events.

8.1.1.4 A.AUTH

Passwords are encrypted during the authentication process.

This Assumption is satisfied by ensuring that:

- OE.AUTH: Password encryption during the authentication process is provided by the IT environment of the TOE, the Internet Explorer or Chrome web browser.

8.1.1.5 A.CLIENT

The web browsers used to access the Admin Interface perform correctly such that when the browser is closed, the active Admin session is terminated. Client applications used to access the Admin API, Security API, and PKI API will perform correctly and when the application is closed, the active Admin session will be terminated.

This Assumption is satisfied by ensuring that:

- OE.CLIENT: The web browsers used to access the Admin Interface will perform correctly and when the Administrator closes the browser, the active Admin session will be terminated. Client applications used to access the Admin API, Security API, and PKI API will perform correctly and when the application is closed, the active Admin session will be terminated.

8.1.1.6 A.NO_EVIL

TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

This assumption is countered by ensuring that:

OE.TRUSTED_ADMIN: TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

8.1.1.7 A.NO_GENERAL_PURPOSE

It is assumed that there are no general-purpose computing capabilities (e.g., compilers or user applications) available on the TOE, other than those services necessary for the operation, administration and support of the TOE.

This assumption is countered by ensuring that:

OE.NO_GENERAL_PURPOSE: There are no general-purpose computing capabilities (e.g., compilers or user applications) available on the TOE, other than those services necessary for the operation, administration and support of the TOE.

8.1.1.8 A.OS_TIME

The OS in the environment shall be able to provide reliable time stamps for use by the TOE.

This assumption is countered by ensuring that:

OE.TIME: The environment must provide a time source for use by the TOE.

8.1.1.9 A.PHYSICAL

Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the environment.

This assumption is countered by ensuring that:

OE.PHYSICAL: Physical security, commensurate with the value of the TOE and the data it contains, is provided by the environment.

8.1.1.10 A.TRUSTED_OS

The underlying OS is trusted to provide protection of the DBMS processes and stored data from other processes running on the underlying OS

This assumption is countered by ensuring that:

OE.PROCESS: The environment is required to provide an execution environment that isolates the TOE from non-TOE processes, such that real and virtual resources offered by the environment can be exclusively used by the TOE.

OE.STORAGE: The environment is required to provide storage mechanisms for use by the TOE. These storage mechanisms must be available only to the TOE.

8.2 Security Requirements Rationale

This section provides evidence supporting the internal consistency and completeness of the components (requirements) in the Security Target. Note Table 8-1 indicates the requirements that effectively satisfy the individual objectives.

8.2.1 Security Functional Requirements Rationale

All of the Security Functional Requirements (SFR) identified in this Security Target are fully addressed in this section and each SFR is mapped to the objective for which it is intended to satisfy.

	O.ACCESS_HISTORY	O.AUDIT_GENERATION	O.MANAGE	O.MEDIATE	O.PROTECTED_COMMUNICATIONS	RESIDUAL_INFORMATION	O.TOE_ACCESS
FAU_GEN.1		X					
FAU_GEN.2		X					
FAU_SEL.1		X					
FCS_CKM.1					X		
FCS_CKM_EXT.4					X		
FCS_COP.1(1)					X		
FCS_COP.1(2)					X		
FCS_COP.1(3)					X		
FCS_COP.1(4)					X		
FCS_COP.1(5)					X		
FCS_HTTPS_EXT.1					X		
FCS_TLS_EXT.1					X		
FDP_ACC.1				X			
FDP_ACF.1				X			
FDP_RIP.2						X	
FIA_ATD.1							X
FIA_UAU.2							X
FIA_UID.2							X
FIA_PMG_EXT.1							X
FMT_MSA.1			X				
FMT_MSA.3			X				
FMT_MTD.1(1)			X				
FMT_MTD.1(2)			X				
FMT_REV.1(1)			X				
FMT_REV.1(2)			X				
FMT_SMF.1			X				

	O.ACCESS_HISTORY	O.AUDIT_GENERATION	O.MANAGE	O.MEDIATE	O.PROTECTED_COMMUNICATIONS	RESIDUAL_INFORMATION	O.TOE_ACCESS
FMT_SMR.1			X				
FPT_APW_EXT.1							X
FPT_SKP_EXT.1					X		
FPT_ITT.1					X		
FPT_TRC_EXT.1				X			
FTA_MCS.1							X
FTA_TAH_EXT.1	X						
FTA_TSE.1							X
FTP_TRP.1(1)					X		
FTP_TRP.1(2)					X		

Table 8-1 Objective to Requirement Correspondence

8.2.1.1 O.ACCESS_HISTORY

The TOE will store and retrieve information (to authorized users) related to previous attempts to establish a session

The TOE Security Objective is satisfied by ensuring that

- FTA_TAH_EXT.1: The TOE must be able to store and retrieve information about previous unauthorized login attempts and the number times the login was attempted every time the user logs into their account. The TOE must also store the last successful authorized login. This information will include the date, time, and location of the attempts. When appropriately displayed, this will allow the user to detect if another user is attempting to access their account. This information should not be deleted until after the user has been notified of their access history.

8.2.1.2 O.AUDIT_GENERATION

The TOE will provide the capability to detect and create records of security relevant events associated with users.

This TOE Security Objective is satisfied by ensuring that:

- FAU_GEN.1: The TOE is required to provide a set of events that it is capable of recording. Among these events the TOE is able to audit must be security relevant events occurring within the TOE. This requirement also defines the information that must be recorded for each auditable event.
- FAU_GEN.2: The TOE is required to associate a user identity with the auditable events being recorded.
- FAU_SEL.1: The TOE is required to allow administrators to configure which auditable events are actually recorded in the audit trail. This provides the administrator with the flexibility in recording only those events that are deemed necessary by site policy, thus reducing the amount of resources consumed by the audit mechanism.

8.2.1.3 O.MANAGE

The TOE will provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.

This TOE Security Objective is satisfied by ensuring that:

- FMT_MSA.1: The TOE is required to restrict the ability to perform operations on security attributes to authorized administrators.
- FMT_MSA.3: The TOE is required to have restrictive default values for security attributes.
- FMT_MTD.1(1): The TOE is required to restrict to authorized administrators (with the admin role or the necessary privileges) the ability to manipulate TOE data used to enforce the TOE security function.
- FMT_MTD.1(2): The TOE is required to restrict to authorized administrators (with the admin role or the necessary privileges) the ability to configure which auditable events are actually recorded in the audit trail.
- FMT_REV.1(1): TOE is required to restrict the ability to revoke user attributes to authorized administrators (with the admin role or the necessary privileges).
- FMT_REV.1(2): The TOE is required to restrict the ability to revoke object attributes to authorized administrators (with the admin role or the necessary privileges).
- FMT_SMF.1: The TOE is required to provide at least the identified management functions for use by the authorized administrators (with the admin role or the necessary privileges).
- FMT_SMR.1: The TOE is required to establish an authorized administrator role.

8.2.1.4 O.MEDIATE

The TOE must protect user data in accordance with its security policy.

This TOE Security Objective is satisfied by ensuring that:

- FDP_ACC.1: This requirement defines the Access Control policy that will be enforced by the TOE on a list of subjects acting on the behalf of users attempting to gain access to a list of named objects. All the operation between subject and object covered are defined by the TOE's policy.
- FDP_ACF.1: This requirement defines the security attribute used to provide access control to objects based on the TOE's access control policy.
- FPT_TRC_EXT.1: The TOE must maintain consistency of replicated TSF data, specifically, the replicated TSF data that specifies attributes for access control must be consistent across distributed components of the TOE.

8.2.1.5 O.PROTECTED_COMMUNICATIONS

The TOE will provide protected communication channels for administrators and support protected communication channels for non-administrative users.

This TOE Security Objective is satisfied by ensuring that:

- FCS_CKM.1: The TOE is required to be able to generate encryption keys to support other cryptographic operations.
- FCS_CKM_EXT.4: The TOE is required to zeroize keys when no longer need to prevent subsequent disclosure.
- FCS_COP.1(1): The TOE is required to implement FIPS-conformant AES in support of cryptographic protocols.
- FCS_COP.1(2): The TOE is required to implement FIPS-conformant cryptographic signatures (ECDSA) using specific algorithms in support of cryptographic protocols.
- FCS_COP.1(3): The TOE is required to implement FIPS-conformant cryptographic hashing using specific algorithms in support of cryptographic protocols.
- FCS_COP.1(4): The TOE is required to implement FIPS-conformant keyed-hash message authentication using specific algorithms in support of cryptographic protocols.

- FCS_COP.1(5): The TOE is required to implement FIPS-conformant cryptographic signatures (rDSA) using specific algorithms in support of cryptographic protocols.
- FCS_HTTPS_EXT.1: The TOE is required to implement HTTP over TLS to protect remote administrative communications.
- FCS_TLS_EXT.1: The TOE is required to implement TLS properly to protect applicable network communication channels.
- FPT_SKP_EXT.1: The TOE is required to prevent even administrators from readily accessing sensitive user and TSF data such as cryptographic keys, thus ensuring the protection of data communication with the TOE.
- FPT_ITT.1: The TOE is required to protect communications from disclosure and detect the modification of those communications when it is transmitted between distributed parts of the TOE.
- FTP_TRP.1(1): The TOE is required to protect communication between itself and its remote administrative users from disclosure and detect the modification of those communications. The TOE is required to use HTTP over TLS to provide these protections.
- FTP_TRP.1(2): The TOE supports protect communication between itself and its remote non-administrative users from disclosure and detect the modification of those communications.

8.2.1.6 **O.RESIDUAL_INFORMATION**

The TOE will ensure that any information contained in a protected resource within its Scope of Control is not released when the resource is reallocated

This TOE Security Objective is satisfied by ensuring that:

- FDP_RIP.2: The TOE is required to ensure the contents of resources are not available to subjects other than those explicitly granted access to the data.

8.2.1.7 **O.TOE_ACCESS**

The TOE will provide mechanisms that control a user's logical access to the TOE.

This TOE Security Objective is satisfied by ensuring that:

- FIA_ATD.1: This requirement defines the attributes of users, including a user ID that is used by the TOE to determine a user's identity and/or role memberships and enforce what type of access the user has to the TOE.
- FIA_UAU.2: The TOE is required to ensure that users must be authenticated in order to access functions, other than those specifically intended to be accessed without authentication (i.e., user data resources available to client hosts).
- FIA_UID.2: The TOE is required to ensure that users must be identified in order to access functions of the TOE.
- FIA_PMG_EXT.1: The TOE is required to provide password management capabilities for administrative passwords.
- FPT_APW_EXT.1: The TOE is required to prevent even administrators from readily accessing sensitive user and TSF data such as passwords.
- FTA_MCS.1: The TOE must ensure that users may only have a maximum of a specified number of active sessions open at any given time.
- FTA_TSE.1: The TOE must restrict access to itself based on certain criteria.

8.2.2 Security Assurance Requirements Rationale

The security assurance requirements for the TOE are the EAL 2 augmented with ALC_FLR.3 components as specified in Part 3 of the Common Criteria. No operations are applied to the assurance components.

EAL 2 augmented with ALC_FLR.3 was selected as the assurance level because the TOE is a commercial product whose users require a low to moderate degree of independently assured security. ALC_FLR.3 was selected to exceed EAL2 assurance objectives in order to ensure that identified flaws are addressed. The TOE is targeted at a relatively benign environment with good physical access security and competent administrators. Within such environments it is assumed that attackers will have little attack potential. As such, EAL 2 augmented with ALC_FLR.3 is appropriate to provide the assurance necessary to counter the limited potential for attack.

8.3 Requirement Dependency Rationale

The following table demonstrates the dependencies among the claimed security requirements. It shows that all dependencies are satisfied. Therefore the requirements work together to accomplish the overall objectives defined for the TOE.

ST Requirement	CC Dependencies	ST Dependencies
FAU_GEN.1	FPT_STM.1	See TimeStamp Note Below.
FAU_GEN.2	FAU_GEN.1 and FIA_UID.1	FAU_GEN.1 and FIA_UID.2
FAU_SEL.1	FAU_GEN.1 and FMT_MTD.1	FAU_GEN.1 and FMT_MTD.1(2)
FCS_CKM.1	(FCS_CKM.2 or FCS_COP.1) and FCS_CKM.4	FCS_COP.1(1)and FCS_CKM_EXT.4
FCS_CKM_EXT.4	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	FCS_CKM.1
FCS_COP.1(1)	(FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1) and FCS_CKM.4	FCS_CKM.1 and FCS_CKM_EXT.4
FCS_COP.1(2)	(FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1) and FCS_CKM.4	FCS_CKM.1 and FCS_CKM_EXT.4
FCS_COP.1(3)	(FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1) and FCS_CKM.4	FCS_CKM.1 and FCS_CKM_EXT.4
FCS_COP.1(4)	(FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1) and FCS_CKM.4	FCS_CKM.1 and FCS_CKM_EXT.4
FCS_COP.1(5)	(FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1) and FCS_CKM.4	FCS_CKM.1 and FCS_CKM_EXT.4
FCS_HTTPS_EXT.1	FCS_TLS_EXT.1	FCS_TLS_EXT.1
FCS_TLS_EXT.1	FCS_COP.1	FCS_COP.1(1)
FDP_ACC.1	FDP_ACF.1	FDP_ACF.1
FDP_ACF.1	FDP_ACC.1 and FMT_MSA.3	FDP_ACC.1 and FMT_MSA.3
FDP_RIP.2	None	None
FIA_ATD.1	None	None
FIA_UAU.2	FIA_UID.1	FIA_UID.2
FIA_UID.2	None	None
FIA_PMG_EXT.1	None	None
FMT_MSA.1	FMT_SMR.1 and FMT_SMF.1 and (FDP_ACC.1 or FDP_IFC.1)	FMT_SMR.1 and FMT_SMF.1 and FDP_ACC.1
FMT_MSA.3	FMT_MSA.1 and FMT_SMR.1	FMT_MSA.1 and FMT_SMR.1
FMT_MTD.1(1)	FMT_SMR.1 and FMT_SMF.1	FMT_SMR.1 and FMT_SMF.1
FMT_MTD.1(2)	FMT_SMR.1 and FMT_SMF.1	FMT_SMR.1 and FMT_SMF.1
FMT_REV.1(1)	FMT_SMR.1	FMT_SMR.1
FMT_REV.1(2)	FMT_SMR.1	FMT_SMR.1
FMT_SMF.1	None	None
FMT_SMR.1	FIA_UID.1	FIA_UID.2
FPT_APW_EXT.1	None	None
FPT_SKP_EXT.1	None	None
FPT_ITT.1	None	None
FPT_TRC_EXT.1	FPT_ITT.1	FPT_ITT.1
FTA_MCS.1	FIA_UID.1	FIA_UID.2

FTA_TAH_EXT.1	None	None
FTA_TSE.1	None	None
FTP_TRP.1(1)	None	None
FTP_TRP.1(2)	None	None
ADV_FSP.1	None	None
AGD_OPE.1	ADV_FSP.1	ADV_FSP.1
AGD_PRE.1	None	None
ALC_CMC.1	ALC_CMS.1	ALC_CMS.1
ALC_CMS.1	None	None
ATE_COV.1	ADV_FSP.2 and ATE_FUN.1	ATE_FUN.1 **and ATE_FUN.1**
ATE_FUN.1	ATE_COV.1	ATE_COV.1
ATE_IND.2	ADV_FSP.2 and AGD_OPE.1 and AGD_PRE.1 and ATE_COV.1 and ATE_FUN.1	ADV_FSP.2 and AGD_OPE.1 and AGD_PRE.1 and ATE_COV.1 and ATE_FUN.1
AVA_VAN.2	ADV_ARC.1 and ADV_FSP.2 and ADV_TDS.1 and AGD_OPE.1 and AGD_PRE.1	ADV_ARC.1 and ADV_FSP.2 and ADV_TDS.1 and AGD_OPE.1 and AGD_PRE.1

Timestamp Note: The TOE is not a physical device and operates as an application within a process provided by the environment. Thus, the environment is providing resources for the TOE. The environmental objective OE.TIME requires that the TOE's environment provide a reliable timestamp which the TOE can use as needed (e.g., within audit records). Thus, the functionality reflected in the dependency of FAU_GEN.1 upon FPT_STM.1 is available to the TOE from the environment.

8.4 TOE Summary Specification Rationale

Each subsection in Section 5.3, the TOE Security Assurance Requirements, describes a security function of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding security function. The set of security functions work together to satisfy all of the security functions and assurance requirements. Furthermore, all of the security functions are necessary in order for the TSF to provide the required security functionality.

This Section in conjunction with Section 6, the TOE Summary Specification, provides evidence that the security functions are suitable to meet the TOE security requirements. The collection of security functions work together to provide all of the security requirements. The security functions described in the TOE summary specification are all necessary for the required security functionality in the TSF. **Table 8-2 Security Functions vs. Requirements Mapping** demonstrates the relationship between security requirements and security functions.

	Security audit	**Cryptographic support**	**User data protection**	**Identification and authentication**	**Security management**	**Protection of the TSF**	**TOE Access**
FAU_GEN.1	X						
FAU_GEN.2	X						
FAU_SEL.1	X						
FCS_CKM.1		X					
FCS_CKM_EXT.4		X					
FCS_COP.1(1)		X					
FCS_COP.1(2)		X					
FCS_COP.1(3)		X					
FCS_COP.1(4)		X					
FCS_COP.1(5)		X					
FCS_HTTPS_EXT.1		X					

FCS_TLS_EXT.1		X					
FDP_ACC.1			X				
FDP_ACF.1			X				
FDP_RIP.2			X				
FIA_ATD.1				X			
FIA_UAU.2				X			
FIA_UID.2				X			
FIA_PMG_EXT.1				X			
FMT_MSA.1					X		
FMT_MSA.3					X		
FMT_MTD.1(1)					X		
FMT_MTD.1(2)					X		
FMT_REV.1(1)					X		
FMT_REV.1(2)					X		
FMT_SMF.1					X		
FMT_SMR.1					X		
FPT_APW_EXT.1						X	
FPT_SKP_EXT.1						X	
FPT_ITT.1						X	
FPT_TRC_EXT.1						X	
FTA_MCS.1							X
FTA_TAH_EXT.1							X
FTA_TSE.1							X
FTP_TRP.1(1)						X	
FTP_TRP.1		X					

Table 8-2 Security Functions vs. Requirements Mapping