MarkLogic Essential Enterprise 9
Security Target

Version 1.0
8 November 2017

Prepared for:

MarkLogic Corporation

999 Skyway Road
Suite 200
San Carlos, CA 94070

Prepared By:

Leidos

Accredited Testing & Evaluation Labs

6841 Benjamin Franklin Drive
Columbia, MD 21046
USA

TABLE OF CONTENTS

LIST OF TABLES

1 Security Target Introduction

This section introduces the Target of Evaluation (TOE) and provides the Security Target (ST) and TOE identification, ST and TOE conformance claims, ST conventions, glossary and list of abbreviations.

The TOE is MarkLogic Server 9, available under the following licensing and delivery models:

- MarkLogic Essential Enterprise 9—a full-featured enterprise database that includes search engine, replication, backup, high availability, recovery, fine-grained security, location services, and alerting designed for use with large, globally distributed applications.

The TOE is an enterprise-class database that provides a set of services used to build both content and search applications which query, manipulate and render XML content. Additionally, the TOE provides "Encryption at Rest" functionality to cryptographically protect the data on media.

The Security Target contains the following additional sections:

- TOE Description (Section 2)—provides an overview of the TOE and describes the physical and logical boundaries of the TOE
- Security Problem Definition (Section 3)—describes the threats and assumptions that define the security problem to be addressed by the TOE and its environment
- Security Objectives (Section 4)—describes the security objectives for the TOE and its operational environment necessary to counter the threats and satisfy the assumptions that define the security problem
- IT Security Requirements (Section 5)—specifies the security functional requirements (SFRs) and security assurance requirements (SARs) to be met by the TOE
- TOE Summary Specification (Section 6)—describes the security functions of the TOE and how they satisfy the SFRs
- Rationale (Section 7)—provides mappings and rationale for the security problem definition, security objectives, security requirements, and security functions to justify their completeness, consistency, and suitability.

1.1 Security Target, TOE and CC Identification

ST Title – MarkLogic Essential Enterprise 9 Security Target

ST Version – Version 1.0

ST Date – 8 November 2017

TOE Identification – MarkLogic Server 9

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 ST and the TOE it describes are conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, 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

This ST and the TOE it describes are conformant to the following package:

- EAL2 Augmented (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 Abbreviations and Acronyms

The following abbreviations and acronyms are used in this document. A brief definition is provided for abbreviations that are potentially unfamiliar, are specific to the TOE, or not obviously self-explanatory.

AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining
CC	Common Criteria
DAC	Discretionary Access Control
DBMS	Database Management System
EAL	Evaluation Assurance Level
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
KEK	Key Encryption Key
KMS	Key Management System
LAN	Local Area Network
NTP	Network Time Protocol
ODBC	Open Database Connectivity
OS	Operating System
PKI	Public Key Infrastructure
PP	Protection Profile
REST	Representational State Transfer

SNMP	Simple Network Management Protocol
SSH	Secure Shell
ST	Security Target
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Function
URI	Uniform resource Identifier
W3C	World Wide Web Consortium
XCC	XML Contentbase Connector
XDBC	XML Database Connector
XML	Extensible Markup Language

1.5 Glossary

The terms listed below are described in order to clarify their usage in the ST as well as in the TOE product documentation.

Amps — *Amps* (short for "amplifications") 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.

Application Server — An application is executed on an *Application Server* (*App Server* for short), which is configured by an administrator with a specific database and port number. The TOE supports the following App Server types: HTTP; XDBC; and ODBC.

Application Server Privileges — *Application Server Privileges* are Execute Privileges that can be configured to control access to each application server (i.e., HTTP or XDBC server). If such a privilege is specified, any users that access the server must possess the specified privilege.

Capabilities — *Capabilities* are operations on documents: Read, Update, Insert or Execute.

CCKEK — Cluster Configuration Key Encryption Key, used to encrypt (wrap) the object key encryption keys (OKEY) for configuration files.

CDKEK — Cluster Data Key Encryption Key, used to directly encrypt FRKEYs for stands, forest journals, and large files.

CKEK — Cluster Key Encryption Key, resides in the keystore and is used to encrypt the data (CDKEK), configuration(CCKEK), and log CLKEK) encryption keys.

CLKEK — Cluster Log Key Encryption Key, used to encrypt (wrap) the object key encryption keys (OKEY) for log files.

Cluster — A *cluster* is a set of hosts that work together.

Collection — A *collection* groups a set of documents that are related. A document may belong to any number of collections. A collection exists in the system when a document in the system states that it is part of that collection. However, an associated collection object is not created and stored in the Security database unless it is protected. Permissions created at the collection level apply to the collection but not to documents within the collection. A user needs to have permissions at both the collection and document level to be able to add documents to or remove documents from a collection.

Compartment Security — *Compartment Security* is an extension to the TOE's DAC access control policy. A *compartment* is a name associated with a role. An administrator specifies that

	a role is part of a compartment by adding the compartment name to each role in the compartment. When a role is *compartmented*, the compartment name is used as an additional check when determining a user's authority to access or create documents in a database. Compartments have no effect on execute privileges.
Document	*Documents* are the basic objects protected by the TOE. A document can comprise XML, text or binary content.
Encryption at Rest	Encryption of data that is stored on digital media.
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.
Forest	A *forest* is a collection of XML, text, or binary documents. Forests are created on hosts and attached to databases to appear as a contiguous set of content for query purposes. A forest can only be attached to one database at a time. Data cannot be loaded into a forest that is not attached to a database.
FRKEY	Fast Rotation Key Encryption Key, used to encrypt (wrap) the object key encryption keys (OKEY) for stands, forest journals, and large files.
Group	A *group* is a set of similarly configured hosts within a cluster.
Host	A *host* is an instance of MarkLogic Server. A host is not configured individually but as a member of a group. A host is added to the `Default` group if it is not joined to another group during the installation process. For example, in cases where MarkLogic is running in a single host environment, the host is added to the `Default` group.
Keystore	Repository for cryptographic keys in the PKCS #11 secured wallet or any external KMS that is KMIP-server conformant.
Key rotation	The process of aging out and replacing encryption keys over time.
KMIP	Key Management Interoperability Protocol (KMIP specification) - governed by OASIS standards body. There are two different currently available versions: 1.1.x and 1.2. MarkLogic Encryption supports 1.2.
HSM	Hardware Security Module or other hardware device is a physical computing device that safeguards and manages digital key materials.
MKEK	Master Key Encryption Key, resides in the keystore, and is used to generate the CKEK, which is enveloped (encrypted) with the MKEK.
OKEY	Object Encryption Key, otherwise known as the data object encryption key, a symmetric key used to directly encrypt objects like stands, forest journals, large files, configuration files, or log files.
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.
PKCS #11	One of the Public-Key Cryptography Standards, and also the programming interface to create and manipulate cryptographic tokens.
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 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.
Wallet	The PKCS #11 secured wallet provided and managed by MarkLogic that functions as the default standalone KMS
XQuery	A query and functional programming language that provides the means to extract and manipulate data from XML documents.

2 TOE Description

The Target of Evaluation (TOE) is MarkLogic Server 9, hereinafter referred to as MarkLogic Server or the TOE. The TOE 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), JSON, text and binary content. Additionally, the TOE provides "Encryption at Rest" functionality to cryptographically protect the data on media.

The remainder of this section provides an overview of the TOE and a description of the TOE, including a description of the physical and logical scope of the TOE.

2.1 TOE Overview

The TOE is built with a blend of search engine and database architecture approaches specifically designed to index and retrieve XML and JSON content. The TOE's native data formats are XML and JSON, and the data is accepted in an 'as is' form. Content in other formats can be converted to an XML representation or stored as is (in binary or text formats) when loaded into the TOE. As an XML/JSON database, the TOE 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. It 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. Applications can also be created using Java, JavaScript or other programming languages that access MarkLogic Server.

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 product documentation uses the following terminology:

- Cluster—a set of one or more instances (see "Host" 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 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.
- Cluster Management Group—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 collection of 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 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, the TOE 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.

- App Server—applications are executed on an Application Server (App Server for short), which is configured by an administrator with a specific database and port number. The TOE supports the following App Server types[1]:
    - HTTP—the TOE enables developers to write XQuery-based web applications by connecting sets of XML content to HTTP servers that can access stored XQuery programs. These applications can return XHTML or XML content to a browser or other HTTP-enabled client application. An HTTP server executes the XQuery programs against the database to which it is connected.
    - XDBC—each XDBC server provides access to a specific forest, and to a library (root) of XQuery programs that reside within a specified directory structure. Applications execute by default against the database that is connected to the XDBC server. XDBC servers execute XML Contentbase Connector (XCC) applications. XCC is an API used to communicate with the TOE from Java or .NET middleware applications. XDBC servers also allow XDBC applications to communicate with the TOE. Both XCC and XDBC applications use the same wire protocol. XQuery requests submitted via XCC return results as specified by the XQuery code. These results can include XML and a variety of other data types.
    - ODBC—an ODBC server supports SQL queries to the TOE. The basic purpose of an ODBC server is to return relational-style data resident in the TOE in response to SQL queries. The ODBC server returns data in tuple form and manages server state to support a subset of SQL and ODBC statements from Business Intelligence (BI) tools. An ODBC server connects with a PostgreSQL front end on the client by means of the PostgreSQL message protocol. The ODBC server accepts SQL queries from the PostgreSQL front end and returns the relational-style data needed by the BI applications to build reports.

The TOE provides "Encryption at Rest" functionality to cryptographically protect the data on media. Encryption at rest provides transparent and selective encryption of data residing on disk in MarkLogic clusters. The following types of data can be configured for encryption:

- User data - data ingested into MarkLogic as databases, along with derived data such as indexes, user dictionaries, journals, backups, and so on
- Configuration files - all configuration files generated by MarkLogic (for example, whenever a change is made to the configuration file)
- Log files - all log files generated by MarkLogic, such as error logs, access logs, service dumps, server error logs, logs for each application server, and the task server logs

Encryption at rest can be configured to encrypt data, log files, and configuration files separately. Encryption at Rest can be applied to individual databases, or applied at the cluster level. Encryption at the cluster level encrypts all data on all the hosts in that cluster. For existing data, a merge or re-index will trigger encryption of data, a configuration change will trigger encryption of configuration files, and log rotation will initiate log encryption.

Encryption is only applied to newly created files once encryption at rest is enabled, and does not apply to existing files without further action by the user. For existing data, a merge or re-index will trigger encryption of data, a configuration change will trigger encryption of configuration files, and log rotation will initiate log encryption.

The keystore for encryption at rest is a key management system (KMS). This keystore can be either the MarkLogic embedded PKCS #11 secured wallet, or an external third party KMS that conforms to the KMIP-standard interface. The embedded keystore is installed by default. The MarkLogic PKCS #11 wallet (the embedded KMS) uses SoftHSM as its default hardware security module (HSM).

SoftHSM can be configured for use with an PKCS #11 compliant hardware security module (HSM) like a plug-in card. The external third party KMS and HSM provide even higher security. The key IDs are provided by the KMS and returned through an SSL/TLS tunnel after the MarkLogic-generated keys have been sent to the KMS and wrapped (encrypted). The actual encryption keys never leave the KMS.

The general encryption key hierarchy for stored keys is depicted in the following figure.

[1] MarkLogic Server 9 also supports WebDAV servers, but these are excluded from use in the evaluated configuration.

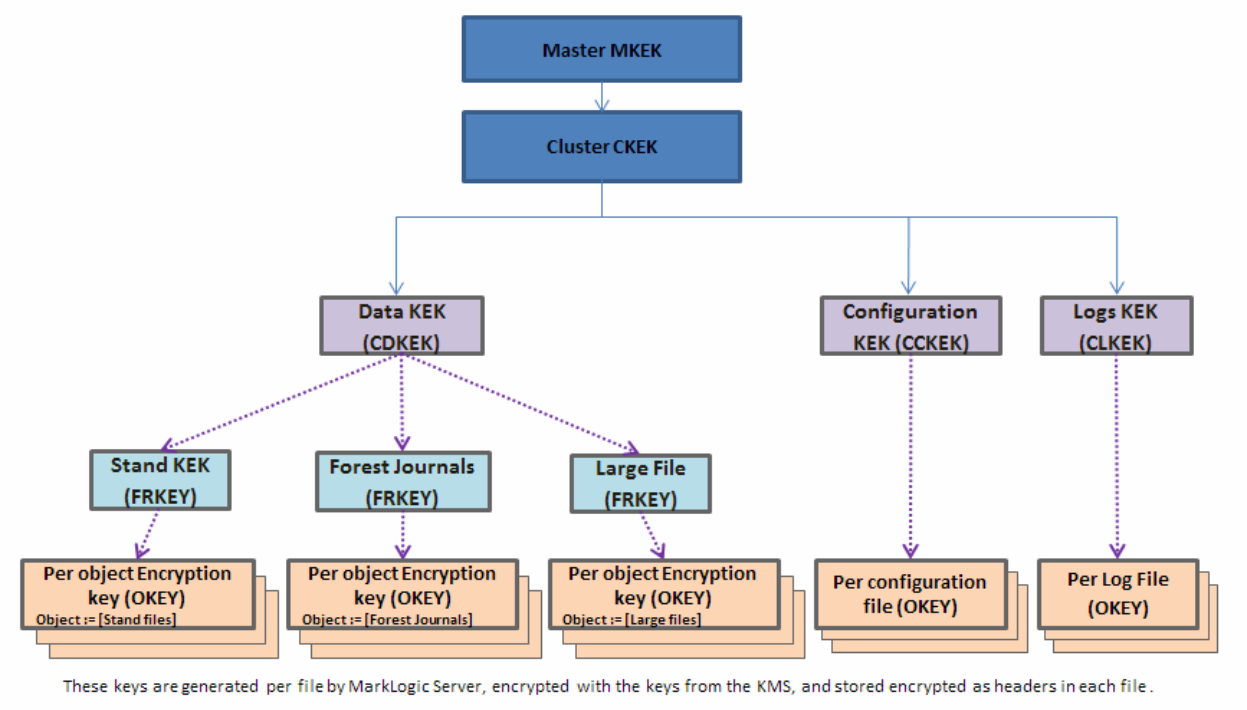

Figure 2-1 Encryption Key Hierarchy

As depicted in the figure above, there are multiple levels to the key hierarchy, each level wrapping (encrypting) the level below it.

When external third party KMS is used, the Master Key Encryption Key (MKEK) and Cluster CKEK optionally exist and reside in the external KMS only. The keystore generates the Cluster Key Encryption Key (CKEK), which is enveloped (encrypted) with or derived from the Master Key Encryption Key. Both the Master Key Encryption Key and the Cluster Key Encryption Key reside in the keystore (key management system or KMS). These keys never leave the keystore and MarkLogic Server has no knowledge or control over these keys. The keys are referenced from the keystore by their key IDs.

The KMS generates and stores the Cluster Level Encryption Keys for data (CDKEK), configuration files (CCKEK), and log files (CLKEK). The Fast Rotation Key Encryption Key (FRKEY) is used to encrypt (wrap) all the Object Encryption Keys (OKEY) generated by MarkLogic Server for each file, so that a unique key protects each file, no matter what category (data, configuration files, logs).

There are three types of OKEY encryption keys at the forest level, for stands, forest journal, and large files. The individual, Object Encryption Keys (OKEY) are randomly generated per file (for stands, journals, config files, and log files, etc.) wrapped (encrypted) with the Fast Rotation Key Encryption Key (FRKEY). So a unique key protects each file within a category (data, configuration files, logs).

Database backups are encrypted using the CDKEK. This key is then encrypted with the cluster key (CKEK).

Key Rotation is the process of aging out and replacing encryption keys over time. Each time data is written to disk, it is re-encrypted with a new key. Data encryption keys are automatically rotated during merges or reindexing. The TOE generates a one-time-use object key for every new file that it writes. Data that hasn't been touched in a while can be re-encrypted by forcing a full merge or re-index of the data. This changes/updates the encryption keys.

When the following keys are rotated: Data KEK (CDKEK), Configuration KEK (CCKEK), and Logs KEK (CLKEK); AES 256 symmetric encryption is used to envelope the object file encryption keys, and the object file encryption keys are re-encrypted (also using AES 256 symmetric encryption).

For the internal PKCS #11 secured wallet (KMS), key encryption keys (KEK) can be manually rotated. Keys can be manually rotated at regular intervals or if an encryption key has been compromised. This type of key rotation can be triggered on individual encryption categories (configuration, data, logs) using MarkLogic built-in functions. When File level keys are rotated by forcing a merge, log rotation and configuration file updates use new keys. Old logs, backups, and configuration files are not re-encrypted. Only the fast rotation keys are re-encrypted with the new data encryption keys (CDKEK, CCKEK, CLKEK).

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. The programmatic 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 require the Admin Interface, including TOE security management tasks such as management of TOE sessions and configuration of auditing. For example, a program can be written to use 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 provides functions for managing objects stored in the security database. For example, it can be used 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 TLS.

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 TLS).

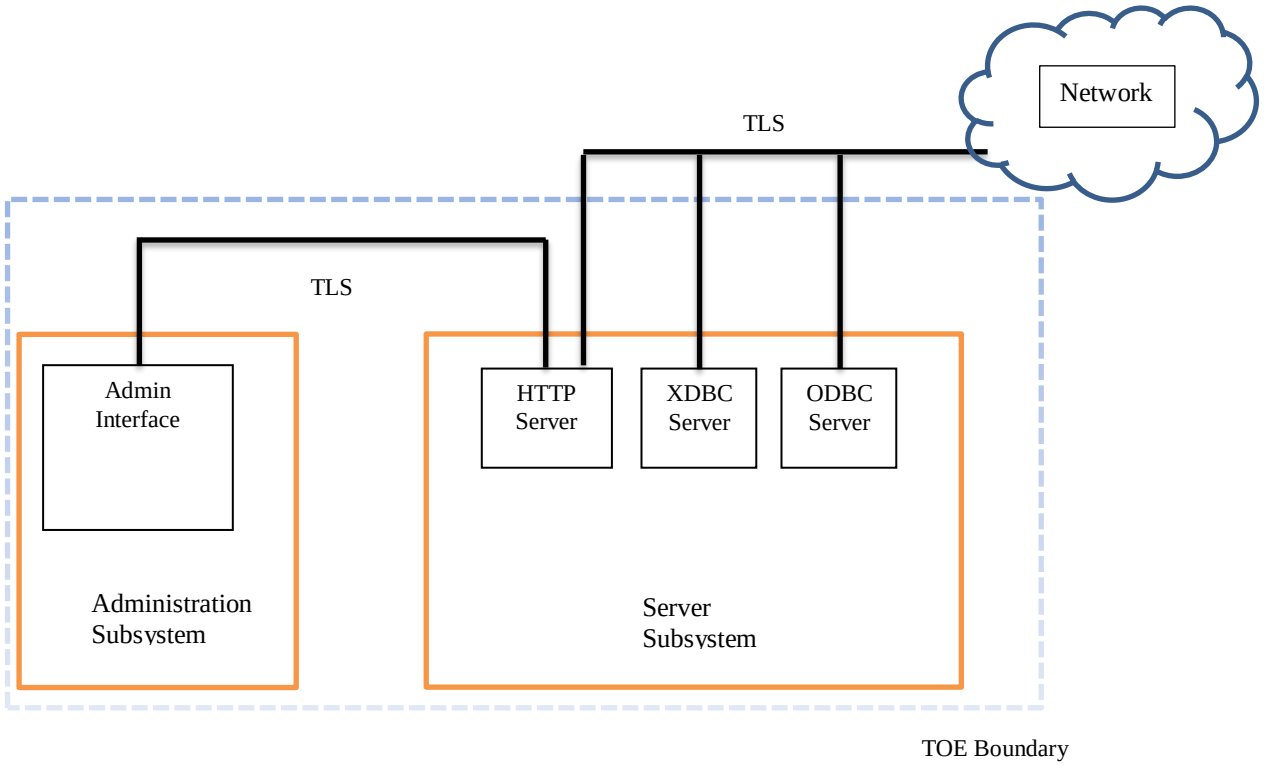

Figure 2-2 TOE Architecture

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 2-2by the lines labeled as "TLS". Two additional programmatic interfaces are provided by XDBC and ODBC protocols that can also use TLS 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). The ODBC server provides read-only access to SQL views that are defined in the context database for that App Server, and is authenticated and authorized based on DAC policy.

The TOE 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 an HTTP server. The Java Client API is implemented in Java, and calls the REST APIs, which in turn run on an HTTP server. The HTTP server is an interface to the TOE that honors DAC policy. The XCC libraries run against an XDBC server, which is also an interface to the TOE that honors DAC policy.

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 on a Linux operating system. The TOE requires the following hardware and OS platforms in its operational environment:

Memory, Disk Space, and Swap Space Requirements

The host system must meet the following minimum requirements:

- 512 MB of system memory, minimum. 2 GB or more recommended, depending on database size.
- 1.5 times the disk space of the total forest size. More specifically, each forest on a filesystem requires its filesystem to have at least 1.5 times the forest size in disk space (or, for each forest less than 32GB, 3 times the forest size).
- Swap space equal to the amount of physical memory on the machine.

Supported Platforms – Server Subsystem

The evaluated configuration of the TOE is supported on Red Hat Enterprise Linux 7 (x64). Note, the deadline I/O scheduler is required on Red Hat Linux platforms. The deadline scheduler is optimized to ensure efficient disk I/O for multi-threaded processes, and the TOE can have many simultaneous threads. In addition, the `redhat-lsb`, `glibc` (both the 32-bit and the 64-bit packages) and `gdb` packages are required.

Supported Platforms – Administration Subsystem

The Administration subsystem is supported on the following browsers in the evaluated configuration:

- Firefox on Windows and Mac OS
- Internet Explorer on Windows
- Chrome on Windows and Mac OS.

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

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 TLS versions 1.0, 1.1 and 1.2. 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.

An optional external third party KMS is permitted in the evaluated configuration and is assumed to be a trusted external IT. The functionality of external third party KMS is not covered by this evaluation.

The TOE can be configured to use external authentication entities (Kerberos or LDAP) to authenticate users. The TOE can also be configured to authenticate the MarkLogic Server specifically as a client to external authentication systems: Kerberos, LDAP and AWS. Kerberos, LDAP and AWS are provided by the operational environment.

2.2.2 TOE Logical Boundaries

This section summarizes the security functions provided by the TOE:

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

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 TOE relies on the operational environment for secure storage of the audit records and for system clock information that is used by the TOE to timestamp each audit record.

Cryptographic Support

The Transport Layer Security (TLS) protocol is used to provide protection of the communications surrounding the remote administrative sessions from disclosure and from undetected modification. The TOE supports TLS v1.0, v1.1, and v1.2. 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 TLS session to protect these communications. Finally, the TOE uses a TLS protected channel to distribute TSF data when it is transmitted between distributed parts of the TOE (that is, hosts within a cluster); and to transmit MarkLogic-generated keys to trusted external IT entities.

The Advanced Encryption Standard (AES) with 256-bit keys is used for Encryption at Rest data encryption (databases, logs, and config files).

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.

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. Access to all TOE-controlled objects is denied unless access, based on role, 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.

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 TOE-controlled objects based on role membership. The TOE maintains the following security attributes belonging to individual users: identity, 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).

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, Data at Rest, user accounts, access control and TOE sessions.

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 TLS, which protects against the disclosure and undetected modification of data exchanged between the TOE and the administrator. Communication with remote customer applications can also utilize TLS 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 TLS is necessary for that specific customer application's data. TLS protects all MarkLogic-generated keys transmitted from the TSF to trusted external third-party KMS from unauthorised disclosure during transmission.

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.

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

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

- *MarkLogic Server Administrator's Guide,* May 2017, Last Revised: 9.0-3, September 2017
- *MarkLogic Server Security Guide MarkLogic 9,* May, 2017, Last Revised: 9.0-3, September, 2017
- *MarkLogic Common Criteria Evaluated Configuration Guide,* May 2017, Last Revised: 9.0-1, May, 2017
- *MarkLogic Server Installation Guide for All Platforms,* May 2017, Last Revised: 9.0-3, September, 2017

The product documentation can be found at http://docs.marklogic.com/.

3 Security Problem Definition

This section defines the security problem to be addressed by the TOE, in terms of threats to be countered by the TOE or its operational environment, and assumptions about the intended operational environment of the TOE.

3.1 Assumptions

This section contains assumptions regarding the operational environment and the intended usage of the TOE.

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

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 section identifies the security objectives for the TOE and its operational environment. The security objectives identify the responsibilities of the TOE and its environment in addressing the security problem defined in Section 3.

4.1 Security Objectives for the TOE

The following are the TOE security objectives:

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.DATA_AT_REST_PROTECTION	The TOE must protect selected/configured data that is stored on digital media using encryption.
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 document resource within its Scope of Control is not released when the document 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 trusted external IT entities; in addition to supporting protected communication channels for non-administrative users.

4.2 Security Objectives for the Environment

The following are the security objectives for the operational environment of the TOE:

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 (SFRs) 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 7.3, Requirement Dependency Rationale).

5.1.1 Extended Family Definitions

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.1 Security audit data generation is included:

Basic Level:

- Restoring consistency

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.

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.

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

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.

5.2 TOE Security Functional Requirements

This section specifies the security functional requirements (SFRs) for the TOE.

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

Requirement Class	Requirement Component
	FCS_CKM.4(1): Cryptographic key destruction (for TLS/OpenSSL)
	FCS_CKM.4(2): Cryptographic key destruction (for Encryption at Rest)
	FCS_COP.1(1): Cryptographic operation (for data encryption/decryption)
	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 signature services using rDSA)
FDP: User data protection	FDP_ACC.1: Subset access control
	FDP_ACF.1: Security attribute based access control
	FDP_RIP.1: Subset residual information protection
FIA: Identification and authentication	FIA_ATD.1: User attribute definition
	FIA_UAU.2: Timing of authentication
	FIA_UAU.5: Multiple authentication mechanisms
	FIA_UID.2: User identification before any action
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_ITC.1: Inter-TSF confidentiality during transmission
	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: Trusted path

Table 5-1: TOE Security Functional Components

5.2.1 Security Audit (FAU)

FAU_GEN.1 – Audit data generation

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

a) Start-up and shutdown of the audit functions;
b) All auditable events for the [***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 (if applicable), and the outcome (success or failure) of the event; and
b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [**information specified in column three of Table 5-2: Auditable** Events].

Table 5-2: Auditable Events

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 object and the subject performing the operation.
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.
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.

FAU_GEN.2 – User identity association

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.

FAU_SEL.1 – Selective audit

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

a) [***object identity, user identity, event type***]
b) [**role, success of auditable security events, failure of auditable security events**].

5.2.2 Cryptographic Support (FCS)

FCS_CKM.1 – Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [**DRBG (AES in CBC mode) for AES keys; RSA key generation]** and specified cryptographic key sizes [**128, 256 bits for AES; 2048, 3072 and 4096 bits for RSA**] that meet the following [**SP 800-90A for AES keys; ANSI X9.31 for RSA keys**].

FCS_CKM.4(1) – Cryptographic key destruction (for TLS/OpenSSL)

FCS_CKM.4.1(1) The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [**overwriting once with zeroes**] that meets the following: [**FIPS 140-2**].

FCS_CKM.4(2) – Cryptographic key destruction (for Encryption at Rest)

FCS_CKM.4.1(2) The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [**overwrite the key encryption keys: Data KEK (CDKEK), Configuration KEK (CCKEK), and Logs KEK (CLKEK) with a new value of a key of the same size**] that meets the following: [**no standard**].

FCS_COP.1(1) – Cryptographic operation (for data encryption/decryption)

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 and 256 bits**] that meets the following: [

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

FCS_COP.1(3) – Cryptographic operation (for cryptographic hashing)

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'**].

FCS_COP.1(4) – Cryptographic operation (for keyed-hash message authentication)

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'**].

FCS_COP.1(5) – Cryptographic operation (for cryptographic signature services using rDSA)

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'**].

5.2.3 User Data Protection (FDP)

FDP_ACC.1 – Subset access control

FDP_ACC.1.1 The TSF shall enforce the [**Discretionary Access Control policy**] on [

- **Subjects: Users**
- **Objects: Documents; Document Elements; Credentials**
- **Operations: create; read; update; insert; execute (read, update, insert, execute are collectively called capabilities)**].

FDP_ACF.1 – Security attribute based access control

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

- **Subject security attributes:**
  - **User Identity**
  - **Role**
- **Object security attributes (Documents):**
  - **Object Identity (URI)**
  - **Permissions, consisting of (capability, role) pairs where roles may have associated compartments**
  - **Protected Collections, where a protected collection has associated permissions**].
- **Object security attributes (Document Elements):**
  - **Protected Path**
  - **Permissions, consisting of (capability, role) pairs where roles may have associated compartments**
  - **Query roleset**
- **Object security attributes (Credentials):**
  - **Object Identity (URI)**
  - **Permissions, consisting of (capability, role) pairs.**

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 User 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.

b) The TSF shall allow a User 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 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 User must have that Role in order to perform the capability.**

iv) If the Document Element belongs to a Protected Path, then

- **for each protected element permission, the User must have that Role in order to perform the capability and**
- **the role must have been added to a query roleset for the protected element.**

v) The TSF shall allow a User a capability to a Credential URI if the User Identity is assigned to or inherits a Role that is specified in the credential.

Otherwise, the TSF shall deny the capability.

].

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 User with the Admin role all modes of access to any Document regardless of the Document's permissions.**
- **Amps[2] can be used to temporarily grant 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 following additional rules: [**no additional explicit denial rules**].

FDP_RIP.1 – Subset residual information protection

FDP_RIP.1.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the [***allocation of the resource to***] the following objects: [**documents**].

5.2.4 Identification and Authentication (FIA)

FIA_ATD.1 – User attribute definition

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**].

FIA_UAU.2 – User authentication before any action

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.

FIA_UAU.5 – Multiple authentication mechanisms

FIA_UAU.5.1 The TSF shall provide [**Passwords, Kerberos, LDAP, Certificate-based authentication**] to support user authentication.

FIA_UAU.5.2 The TSF shall authenticate any user's claimed identity according to the [**following rules:**

- **The user is authenticated using the mechanism for which the App Server the user is attempting to access is configured.**
- **When a user attempts to access an App Server that is configured for local password authentication, the App Server compares the submitted password with the password stored in the security database for the submitted user identity**
- **When an internal user attempts to access an App Server that is configured for certificate-based authentication, the user is authenticated either using the common name in a certificate; or via the distinguished name in a certificate, by matching the distinguished name to an external name configured for an internal user.**
- **When an external LDAP user attempts to access an App Server that is configured for certificate-based authentication, the user is authenticated via a certificate subject name, with internal authorization.**
- **When a user attempts to access an App Server that is configured for both local password authentication and certificate-based authentication, the App Server compares the submitted password with the password stored in the security database for the submitted user identity; and uses the certificate authentication rules specified above to authenticate the user based on whether the user is an internal or external user. The client certificate must match the specified user.**

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

- **When a user attempts to access an App Server that is configured for external authentication, the requested App Server sends the username and password to the LDAP server or Kerberos for authentication. Once authenticated, the LDAP or Kerberos protocol is used to identify the user on the TOE.**
- **When an external user attempts to access an App Server that is configured for certificate-based authentication, the user is authenticated via a certificate subject name, with external authorization. Note: This type of user is entirely defined external to MarkLogic.**]

FIA_UID.2 – User identification before any action	
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.

5.2.5 Security Management (FMT)

FMT_MSA.1 – Management of security attributes	
FMT_MSA.1.1	The TSF shall enforce the [**Discretionary Access Control policy**] to restrict the ability to [***[manage]***] the security attributes [**User Identity, Role, Object Identity (URI), Permissions, Protected Collections**] to [**authorized administrators with the required privileges and database users as allowed by Discretionary Access Control policy**].
FMT_MSA.3 – Static attribute initialization	
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.
FMT_MTD.1(1) – Management of TSF data (TSF data)	
FMT_MTD.1.1(1)	The TSF shall restrict the ability to [***[manage]***] the [**TSF data**] to [**authorized administrators**].
FMT_MTD.1(2) – Management of TSF data (audit selection)	
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**].
FMT_REV.1(1) – Revocation (users)	
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**].
FMT_REV.1(2) – Revocation (objects)	
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**].

FMT_SMF.1 – Specification of Management Functions	
FMT_SMF.1.1	The TSF shall be capable of performing the following management functions: [• **Configure the cryptographic functionality** • **Configure and Manage the Encryption at Rest functionality** • **Configure the auditing functionality** • **Manage user accounts** • **Manage TLS configuration** • **Manage access controls** • **Manage TOE sessions**].

FMT_SMR.1 – Security roles	
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)

FPT_ITC.1 Inter-TSF confidentiality during transmission	
FPT_ITC.1.1	The TSF shall protect TSF data transmitted from the TSF to another trusted IT product from unauthorised disclosure during transmission.

FPT_ITT.1 – Basic internal TSF data transfer protection	
FPT_ITT.1.1	The TSF shall protect TSF data from **[*disclosure and modification*]** when it is transmitted between separate parts of the TOE.

FPT_TRC_EXT.1 – Internal TSF consistency	
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)

FTA_MCS.1 – Basic limitation on multiple concurrent sessions	
FTA_MCS.1.1	The TSF shall restrict the maximum number of concurrent sessions **or http requests** 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 **or http requests** per user.

FTA_TAH_EXT.1 – TOE access history	
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.

FTA_TSE.1 – TOE session establishment	
FTA_TSE.1.1	The TSF shall be able to deny session establishment based on [**User Identity, Role, time of day, day of the week, application server privilege**].

5.2.8 Trusted Path/channels (FTP)

FTP_TRP.1 – Trusted Path

FTP_TRP.1.1 The TSF shall provide a communication path between itself and [***local, remote***] 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, [undetected modification]***].

FTP_TRP.1.2 The TSF shall permit [***local users, remote users***] to initiate communication via the trusted path.

FTP_TRP.1.3 The TSF shall require the use of the trusted path for [***initial user authentication, [all interactions with the TSF]***].

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
ASE: Security Target evaluation	ASE_CCL.1: Conformance claims
	ASE_ECD.1: Extended components definition
	ASE_INT.1: ST introduction
	ASE_OBJ.2: Security objectives
	ASE_REQ.2: Derived security requirements
	ASE_SPD.1: Security problem definition
	ASE_TSS.1: TOE summary specification
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)

ADV_ARC.1 – Security architecture description

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.

ADV_FSP.2 – Security-enforcing functional specification

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.

ADV_TDS.1 – Basic design

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)

AGD_OPE.1 – Operational user guidance	
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.
AGD_PRE.1 – Preparative procedures	
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)

ALC_CMC.2 – Use of a CM system	
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.

ALC_CMS.2 – Parts of the TOE CM coverage

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.

ALC_DEL.1 – Delivery procedures

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.

ALC_FLR.3 – Systematic flaw remediation

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 Security Target Evaluation (ASE)

ASE_CCL.1 – Conformance claims

ASE_CCL.1.1D The developer shall provide a conformance claim.

ASE_CCL.1.2D The developer shall provide a conformance claim rationale.

ASE_CCL.1.1C The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the ST and the TOE claim conformance.

ASE_CCL.1.2C The CC conformance claim shall describe the conformance of the ST to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.

ASE_CCL.1.3C The CC conformance claim shall describe the conformance of the ST to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.

ASE_CCL.1.4C The CC conformance claim shall be consistent with the extended components definition.

ASE_CCL.1.5C The conformance claim shall identify all PPs and security requirement packages to which the ST claims conformance.

ASE_CCL.1.6C The conformance claim shall describe any conformance of the ST to a package as either package-conformant or package-augmented.

ASE_CCL.1.7C The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.

ASE_CCL.1.8C The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.

ASE_CCL.1.9C The conformance claim rationale shall demonstrate that the statement of security objectives is consistent with the statement of security objectives in the PPs for which conformance is being claimed.

ASE_CCL.1.10C The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.

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

ASE_ECD.1 – Extended components definition

ASE_ECD.1.1D The developer shall provide a statement of security requirements.

ASE_ECD.1.2D The developer shall provide an extended components definition.

ASE_ECD.1.1C The statement of security requirements shall identify all extended security requirements.

ASE_ECD.1.2C The extended components definition shall define an extended component for each extended security requirement.

ASE_ECD.1.3C The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.

ASE_ECD.1.4C The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.

ASE_ECD.1.5C The extended components shall consist of measurable and objective elements such that conformance or nonconformance to these elements can be demonstrated.

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

ASE_ECD.1.2E The evaluator shall confirm that no extended component can be clearly expressed using existing components.

ASE_INT.1 – ST introduction	
ASE_INT.1.1D	The developer shall provide an ST introduction.
ASE_INT.1.1C	The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.
ASE_INT.1.2C	The ST reference shall uniquely identify the ST.
ASE_INT.1.3C	The TOE reference shall identify the TOE.
ASE_INT.1.4C	The TOE overview shall summarise the usage and major security features of the TOE.
ASE_INT.1.5C	The TOE overview shall identify the TOE type.
ASE_INT.1.6C	The TOE overview shall identify any non-TOE hardware/software/firmware required by the TOE.
ASE_INT.1.7C	The TOE description shall describe the physical scope of the TOE.
ASE_INT.1.8C	The TOE description shall describe the logical scope of the TOE.
ASE_INT.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ASE_INT.1.2E	The evaluator shall confirm that the TOE reference, the TOE overview, and the TOE description are consistent with each other.

ASE_OBJ.2 – Security objectives	
ASE_OBJ.2.1D	The developer shall provide a statement of security objectives.
ASE_OBJ.2.2D	The developer shall provide a security objectives rationale.
ASE_OBJ.2.1C	The statement of security objectives shall describe the security objectives for the TOE and the security objectives for the operational environment.
ASE_OBJ.2.2C	The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective.
ASE_OBJ.2.3C	The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.
ASE_OBJ.2.4C	The security objectives rationale shall demonstrate that the security objectives counter all threats.
ASE_OBJ.2.5C	The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.
ASE_OBJ.2.6C	The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.
ASE_OBJ.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_REQ.2 – Derived security requirements	
ASE_REQ.2.1D	The developer shall provide a statement of security requirements.
ASE_REQ.2.2D	The developer shall provide a security requirements rationale.
ASE_REQ.2.1C	The statement of security requirements shall describe the SFRs and the SARs.
ASE_REQ.2.2C	All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.
ASE_REQ.2.3C	The statement of security requirements shall identify all operations on the security requirements.
ASE_REQ.2.4C	All operations shall be performed correctly.
ASE_REQ.2.5C	Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.

ASE_REQ.2.6C	The security requirements rationale shall trace each SFR back to the security objectives for the TOE.
ASE_REQ.2.7C	The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.
ASE_REQ.2.8C	The security requirements rationale shall explain why the SARs were chosen.
ASE_REQ.2.9C	The statement of security requirements shall be internally consistent.
ASE_REQ.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_SPD.1 – Security problem definition	
ASE_SPD.1.1D	The developer shall provide a security problem definition.
ASE_SPD.1.1C	The security problem definition shall describe the threats.
ASE_SPD.1.2C	All threats shall be described in terms of a threat agent, an asset, and an adverse action.
ASE_SPD.1.3C	The security problem definition shall describe the OSPs.
ASE_SPD.1.4C	The security problem definition shall describe the assumptions about the operational environment of the TOE.
ASE_SPD.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_TSS.1 – TOE summary specification	
ASE_TSS.1.1D	The developer shall provide a TOE summary specification.
ASE_TSS.1.1C	The TOE summary specification shall describe how the TOE meets each SFR.
ASE_TSS.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ASE_TSS.1.2E	The evaluator shall confirm that the TOE summary specification is consistent with the TOE overview and the TOE description.

5.3.5 Tests (ATE)

ATE_COV.1 – Evidence of coverage	
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.

ATE_FUN.1 – Functional testing	
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.

ATE_IND.2 – Independent testing - sample	
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.

5.3.6 Vulnerability Assessment (AVA)

AVA_VAN.2 – Vulnerability analysis	
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 TOE
- All administrative actions
- Successful use of an amp
- All auditable events as specified in the following table.

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.

Table 6-1: Auditable Events

Each audit record includes 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 (OS) in the TOE's operational environment provides protected storage of audit records generated by the TOE and capabilities to view the stored audit records. The OS is responsible for providing sufficient storage space to hold TOE audit data. The OS administrator can configure conditions under which the TOE starts 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. The OS 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.

The TOE offers the ability to start and stop the audit function independently from the starting and stopping of the entire DBMS. The TOE is able to generate audit events recording the starting and stopping of the DBMS and the starting and stopping of the audit function.

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
- Customer application to TOE sessions.

An administrative management session is initiated by a login and occurs only over HTTPS using TLS. 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. 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 TLS 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 TLS.

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

TLS protects all MarkLogic-generated and wrapped keys transmitted from the TSF to trusted external third-party KMS from unauthorised disclosure during transmission.

The TOE uses the same implementation of TLS for each of these communication pathways. TLS provides protection of the communications pathways from disclosure and from undetected modification.

The TOE uses Advanced Encryption Standard (AES) with 256-bit keys is for Encryption at Rest data encryption.

The TOE uses a FIPS capable OpenSSL object module consisting of OpenSSL version 1.0.2j with OpenSSL FIPS Object Module v2.0, which has undergone a FIPS 140-2 validation. 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. 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 and for data at rest encryption/decryption functions. The TOE also uses RSA from the FIPS object module as part of the verification and use of certificates. Refer to Table 6-2: 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
rDSA	960
SHS	1655
HMAC	1126

Table 6-2: OpenSSL FIPS Object Module Certificates

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 256 bits for AES; and 2048, 3072 and 4096 bits for RSA.

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 during Data at Rest functions and in support of the TLS protocol. The TOE uses AES in CBC mode and 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 keyed-hash message authentication algorithms are used by the TOE implementation of TLS.

For TLS operations, 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. For Encryption at Rest, the Key Rotation functions provide the ability to overwrite existing keys with a new value of a key of the same size. Specifically the following key encryption keys can be rotated (e.g. overwritten): Data KEK (CDKEK), Configuration KEK (CCKEK), and Logs KEK (CLKEK).

The TOE implements HTTPS as specified by RFC 2818. The TOE does not support HTTP connections for administration. The TOE implements TLS versions 1.0, 1.1 and 1.2 and supports the following ciphersuites.

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA

- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_ SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_ 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 800-90A for AES keys and ANSI X9.31 for RSA-based key establishment schemes. The TOE implements AES using key sizes of 128, 256 bits and RSA using key sizes of 2048 3072 and 4096 bits.
- FCS_CKM.4(1): The TOE clears, by overwriting with zeros, plaintext secret and private keys (associated with TLS/OpenSSL) when no longer needed.
- FCS_CKM.4(2): For Encryption at Rest, the TOE destroys key encryption keys (CDKEK, CCKEK, CLKEK), by overwriting them with a new value of a key of the same size.
- 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(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.
- FPT_ITT.1: The TSF shall protect TSF data from disclosure and modification when it is transmitted between separate parts of the TOE.
- FTP_TRP.1: The TOE provides TLS support for secure communication between users and the TOE.

6.3 User Data Protection

The TOE enforces a Discretionary Access Control (DAC) Policy on all subjects, all controlled objects, and all operations among them. The subjects are the authorized users of the TOE. The controlled objects are *documents* and *document elements*. The operations on *documents* are create, read, update, insert, and execute. The operations on *document elements* are read, node-update, and insert. The read, update, node-update, insert and execute operations are termed *capabilities*.

The DAC policy controls access to documents based on the document's identity (its Uniform Resource Identifier (URI)) and the user's role membership. Element level security protects elements or JSON properties in a document using a protected path, where the path to an element or property within the document is protected so that only roles belonging to a specific query roleset can view the contents of that element or property.

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.

Roles are the central point of authorization in the TOE. A *role* is a named entity that provides authorization *privileges* and *permissions* to users and to other roles. Administrators assign roles to users and to other roles (which can in turn include assignments to other roles, and so on). A role gives a user privileges to perform certain actions in the TOE and permissions to access protected documents.

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.

A permission associates a role with a capability (i.e., Read, Update, Insert or Execute). Documents and document elements are assigned permissions. Users assigned the role defined in the permission are able to perform the capability on the associated document or element. For access to a document element, the role must also be added to a query roleset for the protected element. The query roleset attribute specifies what roles can view certain elements in a document. If a role is not associated with the query roleset that has permission to view the element, the role cannot access the contents of that element.

Documents can be organized into *collections*, which are groups of related documents that enable queries to target subsets of content within the TOE. A document may belong to any number of collections simultaneously. A collection is either unprotected or protected. 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. Unprotected collections do not have any security attributes associated with them, so 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. A protected collection is explicitly created using the Admin Interface. 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.

Roles can be *compartmented*. A *compartment* is a name associated with a role. An administrator specifies that a role is part of a compartment by adding the compartment name to each role in the compartment. When a role is compartmented, the compartment name is used as an additional check when determining a user's authority to create or access documents.

Without compartment security, permissions are checked using OR semantics. For example, if a document has **read** permission for **role1** and **read** permission for **role2**, a user who possesses either **role1** or **role2** can read that document. If those roles have different compartments associated with them (for example, **comp1** and **comp2**, respectively), then the permissions are checked using AND semantics for each compartment, in addition to OR semantics for any non-compartmented role. To access the document if **role1** and **role2** are in different compartments, a user must belong to both **role1** and **role2**, as well as a non-compartmented role that has a corresponding permission on the document.

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.

The TOE includes a Secure Credentials feature that enables MarkLogic Server to authenticate specifically as a client to external authentication systems such as Kerberos, AWS, and LDAP. Secure credentials consist of a PEM-encoded x509 certificate and private key and/or a username and password. Secure credentials are stored as secure documents in the Security database on MarkLogic Server, with passwords and private keys encrypted.

A user references a credential by name and access is granted to the Credential URI if the permissions (role and capability) stored within the credential document permit the access to the user. There is no way for a user to get access to the unencrypted credentials.

Secure credentials allow you to control which users have access to specific resources. A secure credential controls what URIs it may be used for, the type of authentication (e.g. digest), whether the credential can be used to sign other certificates, and the user role(s) needed to access the resource.

The security on a credential can be configured three different ways:

- Credentials that secure a resource by username and password.

- Credentials that secure a resource by a PEM-encoded X509 certificate and a PEM-encoded private key.
- Credentials that secure a resource by username and password, as well as a PEM-encoded X509 certificate and a PEM-encoded private key.

The private key and x509 certificate used to configure a secure credential may be obtained from a trusted Certificate Authority or an administrator may generate their own private key and certificate.

Authorized administrators with the `admin` role have explicitly authorized access to all documents. A user with admin privileges can access documents with protected elements by using fn:doc to retrieve documents (instead of using a query), but to see protected elements as part of a query, that user must have the appropriate role(s). 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 function. 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 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 documents. 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.1: 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 be identified and authenticated prior to gaining any further access to TSF-mediated actions. The TOE supports the following types of authentication:

- Local password-based
- Local certificate-based
- Both local password and certificate
- Remote password LDAP-based
- Remote certificate LDAP-based
- Both remote password and certificate LDAP-based
- Remote Kerberos-based.

For local password-based authentication, 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 Server subsystem then authenticates the user credentials against the security database.

For local certificate-based authentication, the TOE uses either the common name in a certificate; or uses the distinguished name in a certificate, by matching the distinguished name to an external name configured for an internal user.

When a user attempts to access an App Server that is configured for both local password authentication and certificate-based authentication, the App Server compares the submitted password with the password stored in the security database for the submitted user identity; and uses the certificate authentication rules specified above to authenticate the user based on whether the user is an internal or external user. The client certificate must match the specified user.

The TOE supports external authentication by means of LDAP and Kerberos. When a user attempts to access an App Server that is configured for external authentication via passwords, the requested App Server sends the username and password to the LDAP server or Kerberos for authentication. Once authenticated, the LDAP or Kerberos protocol is used to identify the user on the TOE. When an external LDAP user attempts to access an App Server that is configured for certificate-based authentication, the user is authenticated via a certificate subject name, with internal authorization.

When an external user attempts to access an App Server that is configured for certificate-based authentication, the user is authenticated via a certificate subject name, with external authorization. Note: This type of user is entirely defined external to MarkLogic.

All security attributes (except external users configured for certificate-based authentication) are stored in the security database of the 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.

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_UAU.5: The TOE supports user authentication using a local password mechanism, and/or certificates and can be configured to use remote Kerberos or LDAP authentication.
- FIA_UID.2: The TOE requires all users to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

6.5 Security Management

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.

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

Only authorized administrators can perform TOE security management functions. The built-in `admin` role 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 of 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 TLS configuration
  - Configure the algorithm and key sizes used by TLS
  - Configure the certificates used by TLS for the HTTPS servers
- Configure Cryptographic functionality
  - Configure certificates and private key for a Secure Credential
  - Configure certificates for certificate-based authentication
- Configure and Manage the Encryption at Rest functionality
  - Enable/disable Encryption at Rest
  - Configure the KMS, HSM to be used by the TOE
  - Rotate encryption keys
  - Re-index database, forest
  - Force a merge
- 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 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 TOE provides administrators with the ability to revoke security attributes associated with users and objects. User security attributes are role membership and password. 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 important to set up default permissions for roles to which that user is assigned. Without default permissions, it is easy to create documents 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 administrators 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.
- FMT_REV.1(1): Only TOE authorized administrators can revoke user security attributes according to enforceable rules.
- FMT_REV.1(2): Only TOE authorized administrators 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 operational 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 operational 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 2.1 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 TLS connection negotiated using the ciphers defined as valid for a TLS session as described in Section 6.2. The TLS connection protects communication between the TOE and the administrator's browser session from disclosure. TLS protects all MarkLogic-generated and wrapped keys transmitted from the TSF to trusted external third-party KMS from unauthorised disclosure during transmission. TLS also supports the detection of modification of the communicated data between the TOE and the administrator's browser session.

The TOE requires all communication with an external authentication entity to be over a secure communication channel using Kerberos or LDAPS (LDAP over TLS). Kerberos and LDAPS are provided by the operational environment.

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

- FPT_ITC.1: The TOE utilizes TLS to protect all MarkLogic-generated keys transmitted from the TSF to trusted external third-party KMS from unauthorised disclosure during transmission.
- 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: Administrators connect to the TOE using HTTPS to use the administrative GUI for management of the TOE. The initial administrator authentication operation, as well as all subsequent remote administration actions, occurs through this HTTPS channel.

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 at session establishment, and the number of unsuccessful attempts since the last successful session establishment by 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. Note that for HTTP, the control limits the number of concurrent HTTP requests.

Application server privileges allow the administrator 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 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 or http requests that belong to a user by enforcing an administrator configurable limit on the number of concurrent sessions or http requests 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 at 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 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.

7.1 Security Objectives Rationale

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

7.1.1 Security Objectives Rationale for the TOE and Environment

This section shows that all secure usage assumptions and threats are completely covered by security objectives for the TOE or operational environment. In addition, each objective counters or addresses at least one assumption or threat.

	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.DATA_AT_REST_PROTECTION		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					
OE.NO_GENERAL_PURPOSE							X			
OE.PHYSICAL									X	
OE.PROCESS										X
OE.STORAGE										X
OE.TIME								X		
OE.TRUSTED_ADMIN						X				

Table 7-1: Security Problem Definition to Security Objective Correspondence

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.

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 and channels with trusted external entities 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.DATA_AT_REST_PROTECTION: To reduce the potential of unauthorized access to data stored on the TOE digital media, the TOE is expected to encrypt the data specified by the administrator.
- 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 identified 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.

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 countered 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.

A.AUTH

Passwords are encrypted during the authentication process.

This assumption is addressed 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.

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 addressed 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.

A.NO_EVIL

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

This assumption is addressed by ensuring that:

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

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 addressed 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.

A.OS_TIME

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

This assumption is addressed by ensuring that:

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

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 addressed by ensuring that:

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

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 addressed 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.

7.2 Security Requirements Rationale

All security functional requirements identified in this Security Target are fully addressed in this section and each is mapped to the objective it is intended to satisfy. Table 7-2: Objective to Requirement Correspondence summarizes the correspondence of functional requirements to TOE security objectives.

7.2.1 Security Functional Requirements Rationale

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

	O.ACCESS_HISTORY	O.AUDIT_GENERATION	O.DATA_AT_REST_PROTECTION	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			X		
FCS_CKM.4(1)						X		
FCS_CKM.4(2)			X					
FCS_COP.1(1)			X			X		
FCS_COP.1(3)						X		
FCS_COP.1(4)						X		
FCS_COP.1(5)						X		
FDP_ACC.1					X			
FDP_ACF.1					X			

	O.ACCESS_HISTORY	O.AUDIT_GENERATION	O.DATA_AT_REST_PROTECTION	O.MANAGE	O.MEDIATE	O.PROTECTED_ COMMUNICATIONS	RESIDUAL_INFORMATION	O.TOE_ACCESS
FDP_RIP.1							X	
FIA_ATD.1								X
FIA_UAU.2								X
FIA_UAU.5								X
FIA_UID.2								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_ITC.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						X		

Table 7-2: Objective to Requirement Correspondence

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.

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.

O.DATA_AT_REST_PROTECTION

The TOE must protect selected/configured data that is stored on digital media using encryption.

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.4(2): The TOE is required to zeroize keys when no longer needed to prevent subsequent disclosure.
- FCS_COP.1(1): The TOE is required to implement FIPS-conformant AES in support of cryptographic protocols.

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.

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.

O.PROTECTED_COMMUNICATIONS

The TOE will provide protected communication channels for administrators and trusted external IT entities in addition to supporting 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.4(1): The TOE is required to zeroize keys when no longer needed 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(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.
- FPT_ITC.1: The TOE is required to protect all TSF data transmitted from the TSF to another trusted IT product from unauthorised disclosure during transmission.
- 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: The TOE is required to protect communication between itself and its remote users from disclosure and to detect the modification of those communications. The TOE is required to use HTTP over TLS to provide these protections.

O.RESIDUAL_INFORMATION

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

This TOE Security Objective is satisfied by ensuring that:

- FDP_RIP.1: The TOE is required to ensure the contents of resources are not available to subjects other than those explicitly granted access to the data.

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_UAU.5: The TOE supports a local password-based authentication mechanism, certificate-based mechanisms and remote authentication using LDAP or Kerberos.
- FIA_UID.2: The TOE is required to ensure that users must be identified in order to access functions of the TOE.
- 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.

7.2.2 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.

7.3 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.4(1), FCS_CKM.4(2)
FCS_CKM.4(1)	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	FCS_CKM.1
FCS_CKM.4(2)	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.4(1), FCS_CKM.4(2)
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.4(1)
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.4(1)
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.4(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.1	None	None

ST Requirement	CC Dependencies	ST Dependencies
FIA_ATD.1	None	None
FIA_UAU.2	FIA_UID.1	FIA_UID.2
FIA_UID.2	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_ITC.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	None	None
ADV_ARC.1	ADV_FSP.1, ADV_TDS.1	ADV_FSP.2
ADV_FSP.2	ADV_TDS.1	ADV_TDS.1
ADV_TDS.1	ADV_FSP.2	ADV_FSP.2
AGD_OPE.1	ADV_FSP.1	ADV_FSP.2
AGD_PRE.1	None	None
ALC_CMC.2	ALC_CMS.1	ALC_CMS.2
ALC_CMS.2	None	None
ALC_DEL.1	None	None
ALC_FLR.3	None	None
ASE_CCL.1	ASE_ECD.1, ASE_INT.1, ASE_REQ.1	ASE_ECD.1, ASE_INT.1, ASE_REQ.2
ASE_ECD.1	None	None
ASE_INT.1	None	None
ASE_OBJ.2	ASE_SPD.1	ASE_SPD.1
ASE_REQ.2	ASE_ECD.1, ASE_OBJ.2	ASE_ECD.1, ASE_OBJ.2
ASE_SPD.1	None	None
ASE_TSS.1	ADV_FSP.1, ASE_INT.1, ASE_REQ.1	ADV_FSP.2, ASE_INT.1, ASE_REQ.2
ATE_COV.1	ADV_FSP.2 and ATE_FUN.1	ADV_FSP.2 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

Table 7-3: Requirement Dependencies

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.

7.4 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 7-4: 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.4(1)		X					
FCS_CKM.4(2)		X					
FCS_COP.1(1)		X					
FCS_COP.1(3)		X					
FCS_COP.1(4)		X					
FCS_COP.1(5)		X					
FDP_ACC.1			X				
FDP_ACF.1			X				
FDP_RIP.1			X				
FIA_ATD.1				X			
FIA_UAU.2				X			
FIA_UID.2				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_ITC.1						X	
FPT_ITT.1		X				X	
FPT_TRC_EXT.1						X	
FTA_MCS.1							X

	Security audit	Cryptographic support	User data protection	Identification and authentication	Security management	Protection of the TSF	TOE Access
FTA_TAH_EXT.1							X
FTA_TSE.1							X
FTP_TRP.1		X				X	

Table 7-4: Security Functions vs. Requirements Mapping