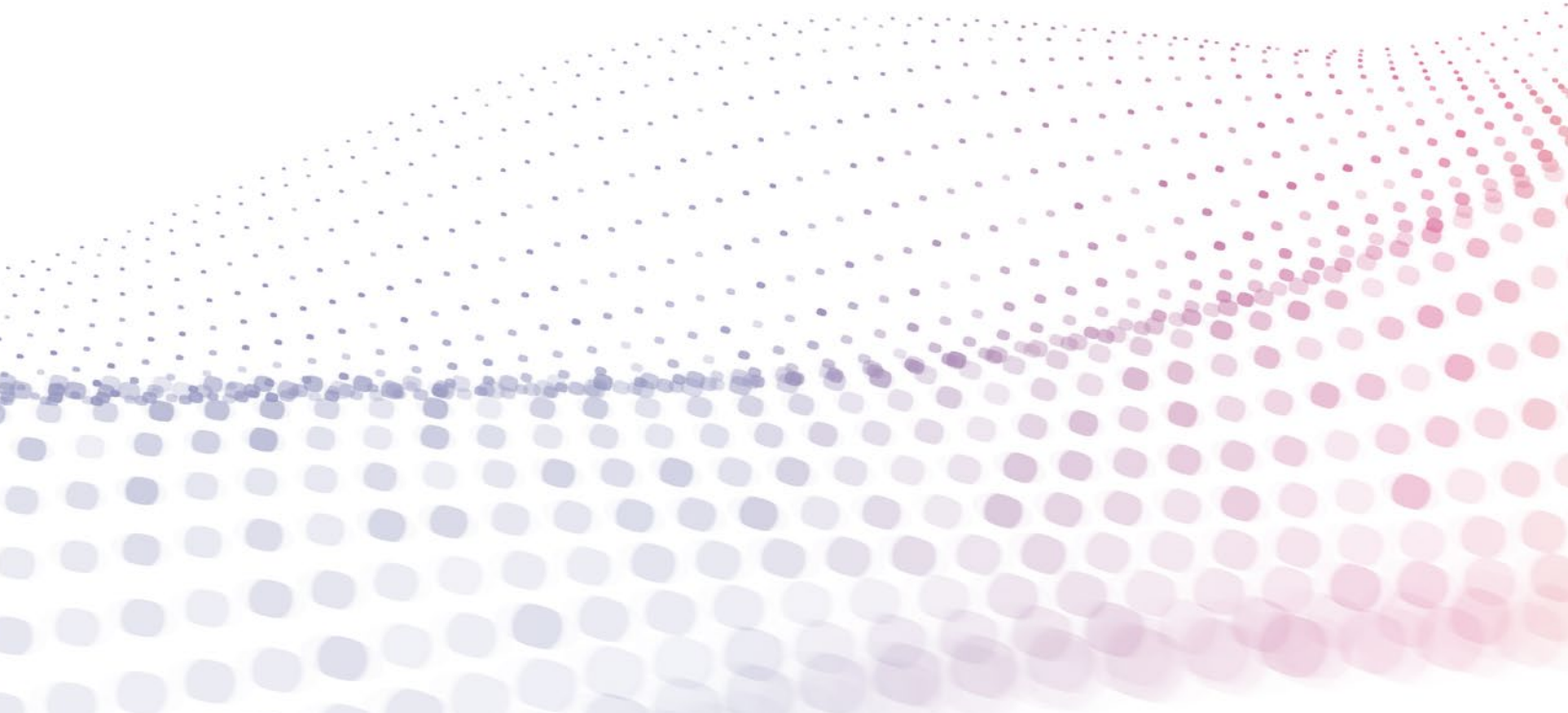
MarkLogic®

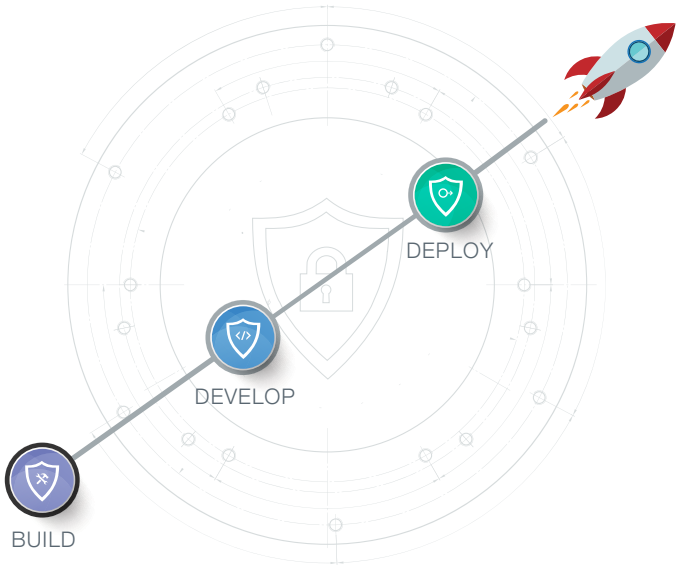
Developing Secure Applications on MarkLogic

MARKLOGIC WHITE PAPER · AUGUST 2018

The burden of security falls heavily on application developers. MarkLogic® lightens the load by providing significant security functionality in the database, with the data. In this technical white paper, we take a close look at that security functionality, and discuss the MarkLogic Security Model that supports developing secure applications on MarkLogic.



Contents



Introduction

Developing secure applications is no easy task. Many organizations consider security a trade-off: build it *fast*, or make it *secure*. It is no surprise then, that according to the Department of Homeland Security, 90 percent of exploits are due to defective software.[1]

MarkLogic's overall approach to security involves looking at an integrated security ecosystem so that organizations using the MarkLogic database can develop the most secure applications possible—without sacrificing agility or data shareability. The MarkLogic security ecosystem is framed by three main components:

- **How We Build a Secure Product**
  This area focuses on how our company's engineering team applies best practices, tools, and techniques to build the most secure product possible.
- **How to Develop Secure Applications on MarkLogic**
  This area focuses on the use of integrated security services and capabilities built into the MarkLogic platform that are available for use by application developers during the development lifecycle.
- **How to Deploy MarkLogic Securely**
  This area focuses on ensuring that MarkLogic is deployed into a secure environment. It includes the ability to work with industry-standard security technologies (e.g., LDAP, Kerberos, SSL/TLS, and KMIP) and also organizational support such as education and consulting.

In this white paper, we focus on that second aspect—how to **develop** secure applications on MarkLogic. We provide an in-depth look at the ***MarkLogic Security Model***, a multi-layered view of how MarkLogic implements security from the data layer controls up through the authorization, authentication, and auditing controls. We also provide specific tips for developers to take into consideration when getting started with MarkLogic.

1 Department of Homeland Security Infosheet, reporting on research done by the Security Engineering Insitute at Carnegie Mellon. <https://www.us-cert.gov/sites/default/files/publications/infosheet_SoftwareAssurance.pdf>

Figure 1: The CIA triad – Confidentiality, Integrity, and Availability – provides the guiding principles for the *MarkLogic Security Model*.

Overarching Principles: Confidentiality, Integrity, Availability

Cybersecurity practitioners often talk about a triad of basic security principles: Confidentiality, Integrity, and Availability (also known as "CIA", and not to be confused with the Central Intelligence Agency). The principles of CIA expand upon the narrow view of cybersecurity as "keeping the bad guys out." It is just as important to ensure that data is protected from unauthorized users, and that the good guys can rely on their data.

Broadly speaking, we can define the *CIA triad* as follows:

- **Confidentiality** – Data is secured for authorized users and not exposed to unauthorized parties or systems. In other words, private data is kept private
- **Integrity** – Data is trusted by authorized users and has not been altered by unauthorized parties or systems. This has to do with data governance—it ensures data is consistent, accurate, and trustworthy over its lifecycle
- **Availability** – Data is accessible and available only to authorized users, and cannot be restricted or made unavailable by unauthorized parties or systems. This includes high availability and disaster recovery (HA/DR)

The CIA triad is how security experts approach the basic security problem. The *MarkLogic Security Model,* discussed next, is how MarkLogic addresses CIA by integrating a data security model into the MarkLogic application development platform. The *MarkLogic Security Model* provides controls to prevent, detect, and mitigate violations of any of the CIA security principles.

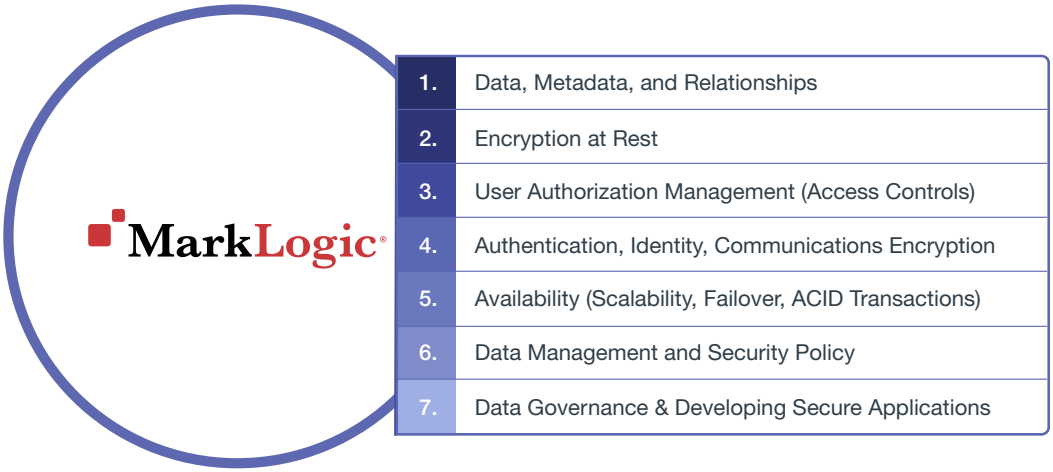


Figure 2: The MarkLogic Security Model shows how we implement security through a set of security controls at different layers.

The MarkLogic Security Model

The *MarkLogic Security Model* shows how MarkLogic implements security through a set of security controls at different layers. The initial layers tie more directly to data, while the latter layers have to do more with policy and governance and are more oriented to the users of the system. Each layer provides security capabilities that help to protect the security controls in the other layers.

The *MarkLogic Security Model* shows the advantage of implementing security policy directly in the data layer. With this approach, it is easy to associate policy metadata with data itself. For example, policy metadata can be used to say that all the data that pertains to topic X, created between these dates, by user ABC, must be handled in such a such a manner. That low-level policy, defined using metadata in the database, then applies to every data consumer.

The *MarkLogic Security Model* includes the following layers:

- **Data, Metadata, and Relationships** – How the underlying data model supports security
- **Encryption at Rest** – How data is secured on disk, including secure key management for separation of duties
- **User Authorization Management** – How roles are used to manage authorization, data access, and privileges
- **Authentication, Identity, Communications Encryption** – Use of industry standards (LDAP, Kerberos, PKI, and certificates) to manage access to the system
- **Availability** – How the system manages scalability, failover, disaster recovery, and ACID transactions
- **Data Management and Security Policy** – How data policy is implemented for regulatory compliance
- **Data Governance & Developing Secure Applications** – What APIs and tools are available to support developers (Note: Data governance utillizes capabilties from all layers)

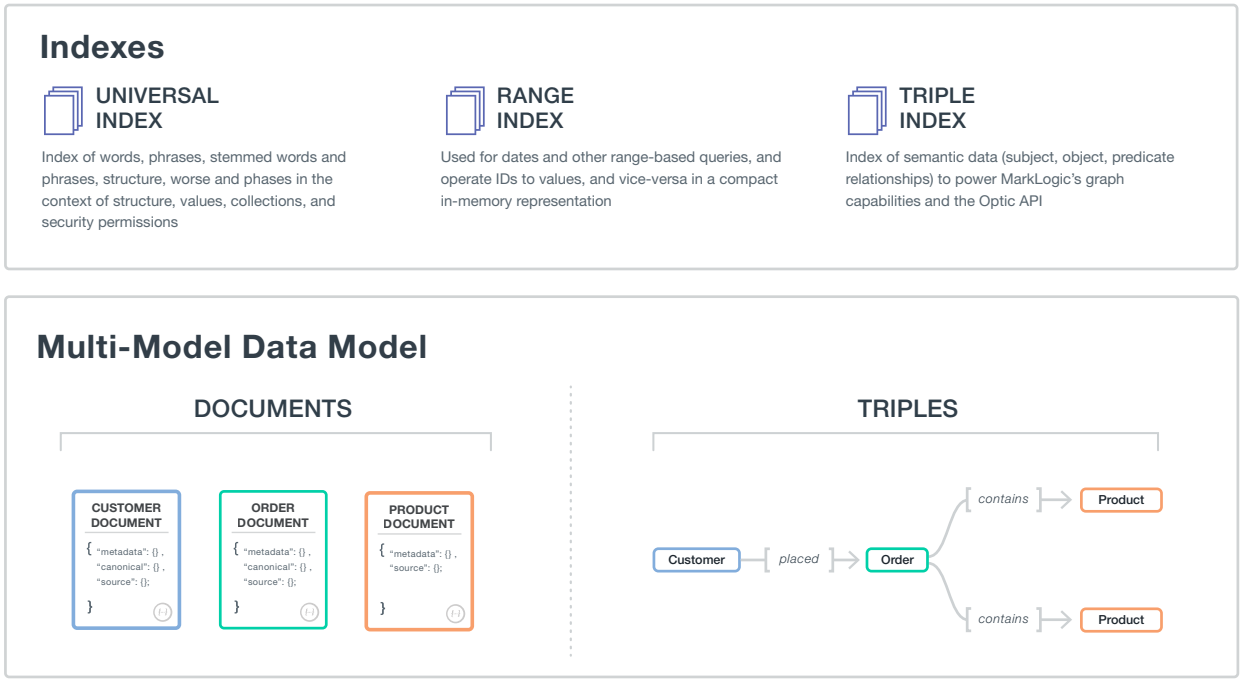


Figure 3: MarkLogic's multi-model approach to storing and managing data, metadata, and relationships is at the center of the security model.

Database Security With a Multi-Model Approach

In the center of the *MarkLogic Security Model* is the data itself—an organization's highest value assets—secured and protected by MarkLogic at the lowest levels.

MarkLogic is a multi-model database that ingests and stores data and metadata as documents and semantic triples. MarkLogic immediately indexes all of the data using a **Universal Index** so that it can be immediately queried. These indexes are secured with the same access controls as the data. A summary of MarkLogic's multi-model approach to managing and indexing data is depicted in Figure 3 above.

With this multi-model approach to data modeling, data is expressed as a cohesive system of entities and relationships. Unlike with a relational database, the document model provides the flexibility to keep data, security, lineage, and provenance information together. Security entitlements are stored in the documents themselves and not in separate systems or across tables within a schema.

For instance, you can use attributes or properties to say who can read the document (also called "markings" in the lingo of database security). The entitlements travel with the document, so even while migrating, transforming, and harmonizing data, the security is always there. This is very hard to achieve with a rows and columns approach to storing data in a relational database. In fact, many relational databases require additional middleware to manage entitlements.

For more information on how MarkLogic manages and indexes data, read the e-book, Inside MarkLogic Server. Or, to gain a broader understanding on why organizations are choosing multi-model databases over traditional relational databases, read the O'Reilly book, Building on Multi-Model Databases.

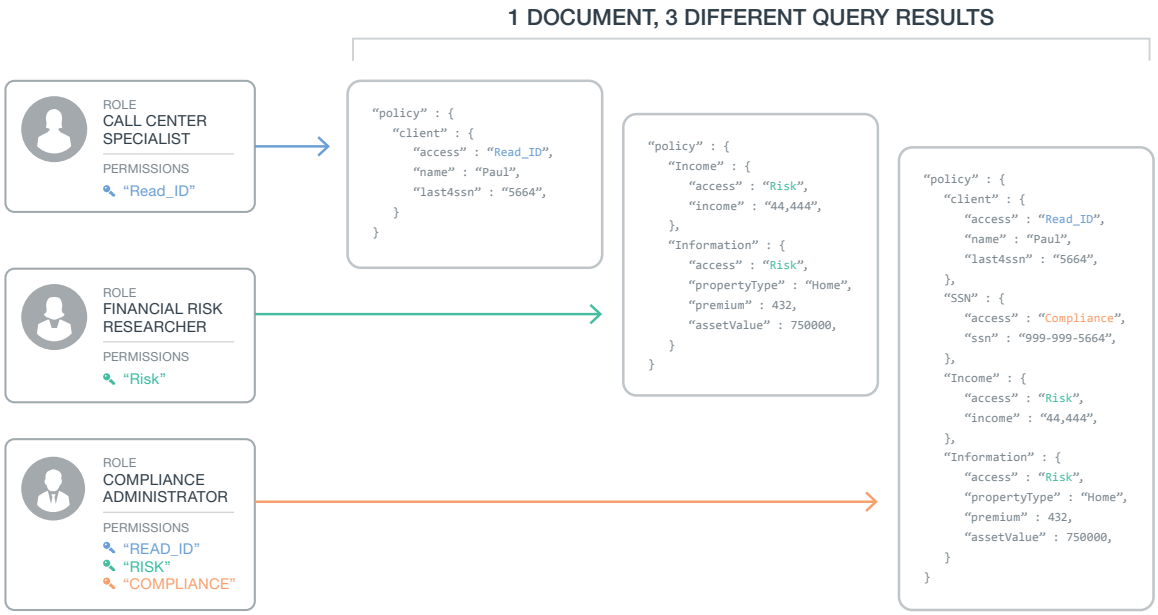


Figure 4: Element Level Security provides security on elements within XML documents or properties within JSON documents, which is the case with this example. The security controls travel with the data and specify which roles have access to which parts of the document.

Document Level Security: Fine-Grained Access Control

By default, MarkLogic provides **Document Level Security**—security at the individual document level. Users must have the right role or roles (e.g., backup administrator, security administrator, database configuration, monitoring, audit administrator, etc.) to access a specific universe of documents. We discuss more of the details about how **Role Based Access Control (RBAC)** works in a later section.

Security works all the way down to individual nodes within documents. Documents, or portions thereof, have permissions which associate a role to its read, update, insert capabilities. And, the document is invisible without the right role to grant the necessary privileges.

MarkLogic also makes it possible to temporarily provide an additional role for a user while they are performing a particular task. This temporary extension of a role is called an "amp." Amps allow some administrative duties to be delegated without needing to give users full permission to become database administrators.

For example, an amp may be helpful when a database adminstrator (DBA) needs to delegate administrative duties for managing user passwords for a defined set of users in a Platform as a Service (PAAS) cloud environment. The DBA would amp a "change-password" function to run as "admin," allowing users to call that function without actually being a database admin.

One other very important component of role based security is that MarkLogic administrative tasks can be scripted and executed by an admin in an operational environment—without ever sharing credentials with admins. Again, this ensures that administering MarkLogic is as secure as possible, all using the same central concepts of role based security.

Element Level Security: Even Finer-Grained Access Control

Element Level Security provides even finer grained security by allowing security administrators to apply additional controls to individual parts of a document at the level of *JSON properties* or *XML elements* within documents.

Element Level Security is a step above "cell-level" security in relational databases, as it is not restricted to protecting a certain set of cells in a relational database schema. With MarkLogic, the elements and properties can appear anywhere in a document and still be secured. Specific information inside a document may be hidden from a particular user based on the user's role, while still providing access to other information in the document.

The key features and benefits of Element Level Security include the following:

- **Based on user roles** – MarkLogic will not allow access to any data element, no matter the path (i.e., through documents, indexes or through any other mechanism) unless a user has the proper privileges and permissions
- **Real-time protection** – MarkLogic protects data during real-time operations, including queries and updates
- **Secure data regardless of schema** – MarkLogic protects sensitive information wherever it happens to appear within the structure of a document using rich, industry-standard, path expressions
- **Secure using attributes and values** – MarkLogic secures data using attributes of XML elements or values of JSON properties. For example, consider the XML element `<person classification="secret">John</person>` that has the classification attribute `"secret."` A more restrictive security rule can be applied to any elements that have that attribute
- **Leak-proof via advanced indexing method** – In MarkLogic, content that matches protected paths are hashed, combined with hashed roles, then added to the indexes. This ensures no confidential data is ever plainly stored in the indexes

Practical Considerations

- Remember: MarkLogic uses a multi-model approach to storing, managing, and searching data. This model allows you to store data governance metadata (security, lineage, and provenance information) right alongside the data itself
- By default, MarkLogic uses document level security, though you can also go a step further by implementing Element Level Security
- When implementing Element Level Security, developers should consider the following:
  - Protected paths are required to secure data within documents and must be inserted into the Security database
  - Ensure that rolesets are established created using the built-in helper functions
  - Be aware that the more rolesets you have, the more performance may be impacted (See the chapter on Element Level Security in the Security Guide for more information)

For certain customers with particularly sensitive information, Element Level Security supports capabilities such as tear-lines, where a specific part of the document is at a lower, more shareable security classification. This balances the need to share data with the requirement to safeguard sources and methods even if the sources have to be controlled at a higher security level or contained in a special compartment.

Another feature, **Redaction**, addresses privacy concerns by making it possible to remove, mask or transform information when importing, exporting, or copying data to and from MarkLogic.

Redaction helps prevents leakage of sensitive information to unauthorized users. For example, Redaction is often required when providing data for analysis by data scientists, or when a developer needs production data but should not have access to real credit card data or personally identifiable information (PII). Overall, Redaction helps avoid privacy violations and reduces risk while enabling secure data sharing.

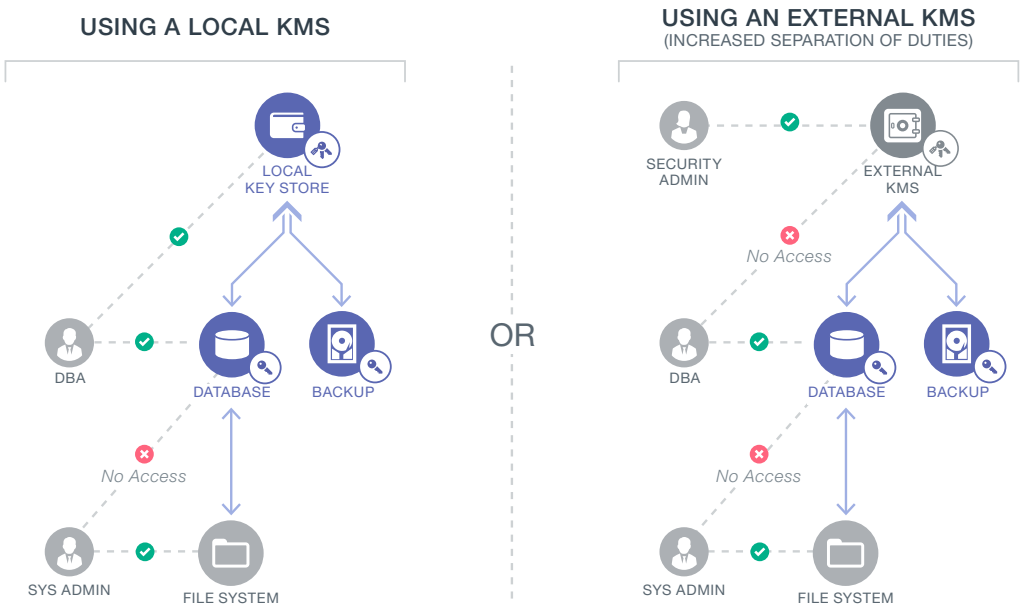


Figure 5: Encryption at Rest provides transparent encryption of databases, logs, configuration files, and backup.

Preventing Database Bypass With Encryption at Rest

MarkLogic provides **Encryption at Rest**, which enables transparent and selective encryption of data residing on disk to ensure confidentiality and prevent database bypass threats. MarkLogic encrypts databases, logs, configuration files and backup. And, it does not matter where the system is deployed. Encryption works locally and in the cloud.

Encryption at Rest significantly enhances data security controls by enforcing separation of duties and preventing information tampering of data residing on disk. With separation of duties, the system administrator has access to the host, but a security administrator controls the encryption keys. This reduces potential threats, particularly those posed by insiders.

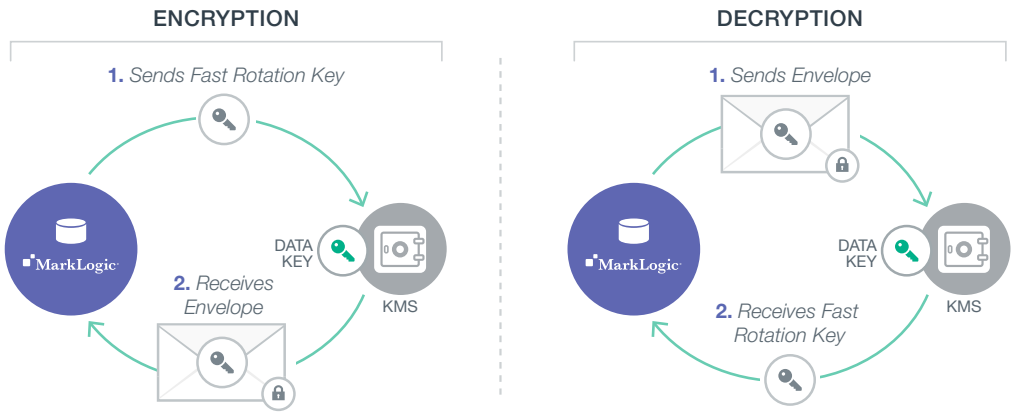


Figure 6: With Encryption at Rest, to access low level keys and read files, MarkLogic sends an envelope to the KMS, which then sends back the unencrypted key. If using an external KMS, MarkLogic has no access to enveloped keys, which means no access to files.

MarkLogic uses well-known encryption technology. It is NIST-approved, AES-256, hardware-optimized, key-based encryption technology. Like any encryption technology, there are cryptographic keys that users have to securely manage. MarkLogic manages keys in the most secure way possible to reduce the likelihood of any keys being compromised and minimizing the damage if individual keys are ever compromised.

First, MarkLogic provides fast key rotation. Data keys are continually updated to provide an additional level of security just like updating your password on a frequent basis. And, even if a key is compromised before it is rotated, it would only enable access to a very small subset of data. Each new file in a stand, log, configuration file, or journal is encrypted with a new key so dictionary, known plaintext, or brute force attacks could compromise just one file.

Second, in addition to fast key rotation, MarkLogic stores and manages the encryption keys separate from the encrypted data. Many other databases do not do this—they store keys and data in the database together. MarkLogic's approach is more secure and is an industry recommended best practice per OWASP security recommendations.

MarkLogic uses enveloped encryption keys with keys managed by separate entities other than the database administrator, ensuring separation of duties. MarkLogic sends the envelope to the KMS, which then sends back the unencrypted key. If using an external KMS, MarkLogic has no access to envelope keys, which means no access to files, no ingestion, and no compromises.

MarkLogic provides powerful key management either through a local KMS or external Key Management System (KMS).[2] A KMS, or "keystore," is a secure location where the enveloped encryption keys used to encrypt data are stored and managed. Both options for key management, local and external, provide high performance and are transparent to users:

1. **Local KMS** (enabled by default) – Default rapid key rotation is provided with the product and is enabled if the local KMS option is selected and encryption is enabled
2. **External KMS** (paid option) – If you have purchased the Advanced Security Option, then you can enable *Advanced Encryption*, which makes it possible to use an third-party, external KMS that is KMIP 1.2 compliant

2 To use an external Key Management System (KMS), you must purchase the Advanced Security Option separately. The Advanced Security Option includes the ability to use an external KMS, Redaction, and Compartment Security.

Practical Considerations

- No special coding is required to enable Encryption at Rest
- Encryption at Rest is totally transparent to application developers
- Enabling Advanced Encryption and using an external, third-party KMS is an option that provides further separation of duties and significantly raises the MarkLogic security profile database

When Encryption at Rest has a Key Management System (KMS) deployed and managed externally, separately from the application servers in the MarkLogic cluster, it provides additional separation of duties between the security administrator, application administrators, and other system administrators. It also allows you to integrate with the key management tools and processes that you may already be using.

MarkLogic interoperates with third-party external KMS systems that are KMIP 1.2 compliant. Key Management Interoperability Protocol (KMIP) is a communication protocol standard that defines message formats for the manipulation of cryptographic keys on a key management server.

Preventing Unauthorized Access to The Database

Many breaches occur because users have unauthorized access, giving users privileges and permissions they should never have had, or unintentionally giving users access to far more than they should have had in the first place.

With MarkLogic, roles are central to all security and authorization, data access, and granting (or denying) privileges. In the *MarkLogic Security Model*, user authorization management is more fundamental than in traditional databases where data and access controls are not coupled tightly together. With MarkLogic, access controls are tightly integrated with data management. MarkLogic can manage data from disparate data sources, process transactions at scale, maintain high availability—all while maintaining granular, fine-grained access controls.

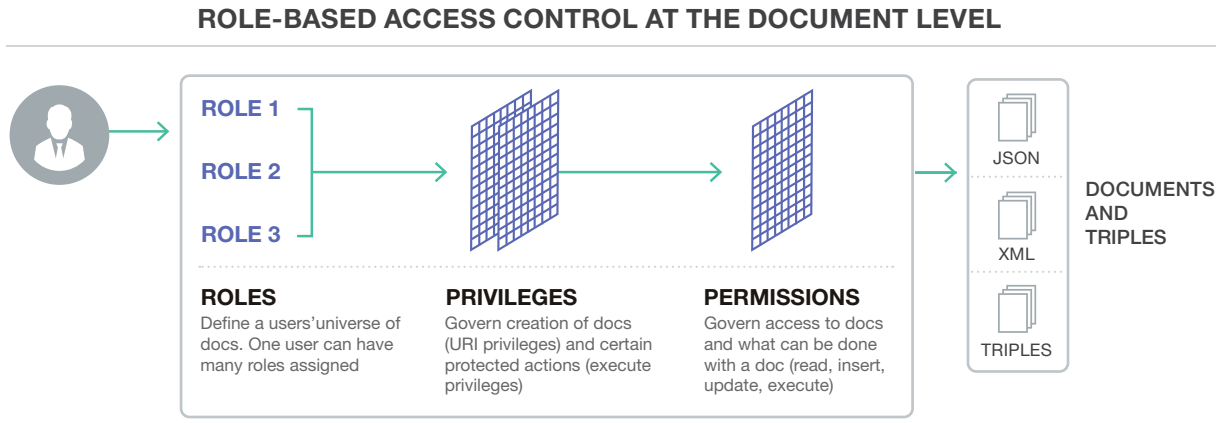


Figure 7: Role Based Access Control (RBAC) is the primary, default approach MarkLogic uses to manage user authorization.

Role Based Access Control (RBAC): Database Security by Default

By default, MarkLogic employs Role Based Access Control (RBAC), where each user is assigned any number of roles, and these roles have associated permissions and privileges. A user's privileges and permissions are based on the roles assigned to the user. Privileges are like doors. When the door is locked, you need to have the key to the door in order to open it. Permissions are used to protect documents and are assigned either at load time or as a separate administrative action. Each permission is a combination of a role and a capability (read, insert, update, node-update, execute).

Security Database: A Safe, Simple Approach to Storing Security Objects

A specific security database is created when MarkLogic is installed in order to store security objects such as privileges, roles, users, associated data, and other security information. Because the security database is managed in MarkLogic, it inherits all of the security, scalability, and HA/DR capabilities that the database provides.

Authentication in MarkLogic occurs via the security database in combination with external authentication (e.g., LDAP/Kerberos/PKI/Certificates). Typically, a single security database services the entire MarkLogic cluster and is used to authorize actions in MarkLogic.

The security database stores both configurable items such as users, roles, customized privileges, and also stores pre-defined system roles and privileges. Pre-defined privileges control access to privileged activities in MarkLogic such as loading data, accessing URIs, or creating users.

The security database is initialized during the installation process and is locally replicated using MarkLogic's built-in capabilities to achieve high availability of the overall system. It is possible to have multiple security databases if necessary for disaster recovery.

Security Indexes: Enable High Performance Security

To manage data access, MarkLogic leverages security indexes. Like MarkLogic's other indexes, these indexes are sophisticated indexes built for fast, limitless querying. MarkLogic gathers the term lists for each role, and unions those together to create that user's universe of documents. It intersects this list with any ad hoc query the user runs to make sure the results only display documents in that user's universe. Term lists are created when data is ingested into MarkLogic, enabling fine-grained access controls at the entity attribute level and/or element level.

For more information, read the Concept Guide on Indexing in MarkLogic in the documentation.

Additional Authorization Models: Flexibility for Every Use Case

Compartment Security

Compartment Security is available with the Advanced Security paid option and provides a higher level of access control. Typically, when roles are not compartmented, satisfying any privilege authorization condition is sufficient (e.g., user is a "U.S. citizen" *OR* has a "Top Secret" clearance). When a role is compartmented, all privileges associated with a resource must be valid at the same time (e.g., user is a "U.S. citizen" *AND* has a "Top Secret" clearance).

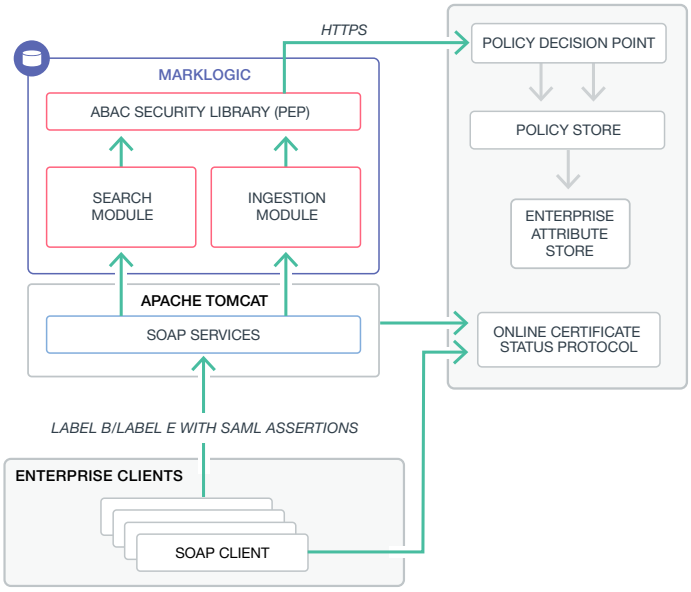


Figure 8: This example depicts how one government organization implements Attribute Based Access Control (ABAC).

ABAC, LBAC, & PBAC

Beyond RBAC, MarkLogic also supports **Attribute Based Access Control (ABAC)**, and **Policy Based Access Control (PBAC)**. These models further restrict access based on attributes (i.e., metadata about the data such as provenance, geo-location, time of day, etc.), policy information stored in document metadata, or simple labels representing "high" or "low" levels of trust.

Attribute-based access control (ABAC) can be implemented many ways in MarkLogic. One approach, illustrated in Figure 6, was used by a U.S. government customer in order to meet stringent requirements for information assurance.

In this organization's SOA environment, end users perform two-way SSL authentication with applications using a physical token called a Common Access Card (CAC). Applications then make "Label B" or "Label E" SOAP service calls to MarkLogic with Security Assertion Markup Language (SAML) assertions to pass on the subject ID based upon their ID from the CAC certificate.

With trust configured between the client and MarkLogic, this ID is used to ask an external PDP (Policy Decision Point) whether that person has access to information domains relevant to the search. Requests to the PDP are through XACML queries over a SOAP interface. The PDP is responsible for fetching the subjects' attributes from attribute server, policy from the policy store, and returning a "Permit," "Deny," "NotApplicable," or "Unknown."

Authorization Management: Apply the Principle of Least Privilege

In MarkLogic, users have roles that are given certain privileges and permissions. *Permissions* provide a role with the capability to perform certain actions (read, insert, update, execute) on a document or a protected collection (a collection is an organized group of documents in MarkLogic). In addition to permissions, *URI Privileges* control access to creating documents in a given URI range and *Execute Privileges* allow developers to control authorization for the execution of an XQuery or JavaScript function.

Each user has an associated user name and password, and a set of default collections they have access to. When a user creates a document but does not explicitly associate the document with a set of collections, the document is automatically added to the user's default collections (assuming that the user's default permissions have been established by the MarkLogic admin). Default permissions are created for a user such that when a user creates a document but does not explicitly set the permissions for the document, the document will be given the user's default permissions. If no permissions are set, then the default permission is no access, or least privilege.

Practical Considerations

- By default, MarkLogic employs Role Based Access Control (RBAC). All you need to do is setup and assign the roles for each user
- You can also setup other security models such as Attribute Based Access Control (ABAC) and Policy Based Access Control (PBAC) to support certain use cases
- A security database is automatically created when MarkLogic is installed in order to store security objects such as privileges, roles, users, associated data, and other security information

---

Using Authentication, Identification, & Communications Encryption

Authentication is critical to identify those gaining accessing to a system. MarkLogic follows industry standards for authentication (e.g., TLS) and can plug into and leverage standard enterprise authentication systems such as LDAP, Kerberos, and PKI.

Authentication & Identification: Manage Database User Access

MarkLogic supports mutual authentication, whereby the client also authenticates itself to MarkLogic by sending its digital certificate to the server. MarkLogic also supports external authentication using **Lightweight Directory Access Protocol (LDAP)**, **Kerberos**, or certificate-based authentication. MarkLogic can:

- Authenticate users using an internal security database or external authentication mechanism, or both
- Use external authentication and authorization with LDAP authorization and Kerberos tickets
- Utilize external authentication which can be established at system initialization time
- Track all authentication configuration changes as audit events in the server logs

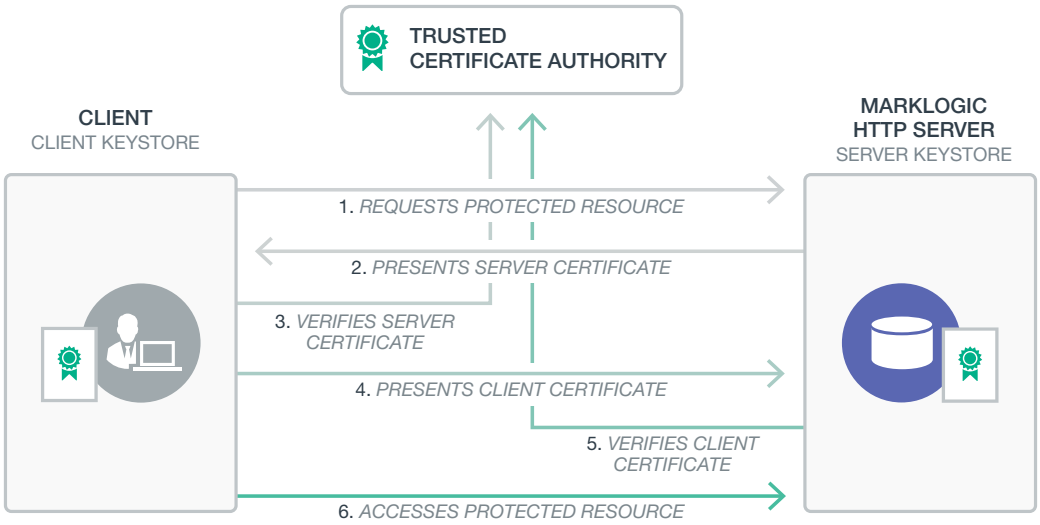


Figure 9: This graphic depicts MarkLogic's mutual authentication between the database and client applications using FIPS enabled TLS/SSL. A similar process is used to secure communication between database nodes, cluster to cluster, and Admin GUI to database.

Communications Encryption: Prevent Network Tampering

Transport Layer Security (TLS) is a communications security standard which has replaced Secure Sockets Layer (SSL) for providing encrypted communication for data-in-motion (usually via HTTPS). Typically, a handshake procedure authenticates the server so that the client can trust the server but the client remains unauthenticated to the server. MarkLogic supports mutual authentication where the client also holds a digital certificate, which it sends to the server so that both the client and server are authenticated.

MarkLogic uses OpenSSL to implement SSL/TLS and maintains the current version of OpenSSL so any fixes available will always be incorporated into MarkLogic. MarkLogic also supports a **FIPS 140-enabled version of OpenSSL** and supports all the operations necessary to enable full SSL/TLS encryption, such as automating certificate setup.

Practical Considerations

- MarkLogic has a flexible and extensible authentication/authorization model that is based on industry standards
- MarkLogic supports strong communications security using self-signed certificates and third-party certificate authorities

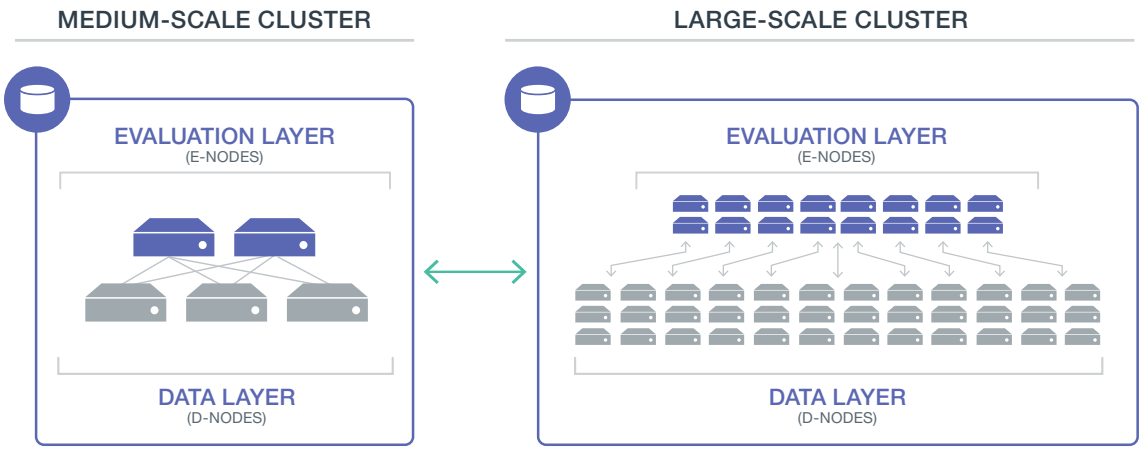


Figure 10: MarkLogic's cluster architecture works as a distributed system, having a design that enables scalability, and HA/DR.

Ensuring Your Data Is Always Available

Distributed computing and clustering is a key part of availability (referring to the "Availability" part of the "CIA" triad). MarkLogic is designed with a distributed cluster architecture that provides the foundation for many powerful features that work together to make sure your data is always available—both in the face of system failures and at massive scale.

Scalability & Elasticity: Designed for Large-Scale Systems

Distributed computing and clustering is usually understood in the context of performance, but it is also key to providing availability. MarkLogic's cluster architecture of distributed computing is separated into E-Nodes for code execution and query planning, and D-Nodes to serve up and manage data.

This architecture enables highly available and consistent transactions with high scalability and throughput. Integrated into the MarkLogic cluster architecture is a set of powerful features including **High Availability (HA)**, **Disaster Recovery (DR)**, **Tiered Storage**, and **ACID Transactions**.

For more information, read the Scalability, Availability, and Failover Guide in the documentation.

High Availability & Failover: No Single Point of Failure

MarkLogic provides a variety of disaster recovery capabilities including full and customizable backup, incremental backup, journal archiving. Together, these capabilities can be used to form a complete enterprise-grade disaster recovery strategy.

Regarding MarkLogic's disaster recovery replicas, they support the same encryption, security roles, auditing, and other protections configured in the main cluster. This includes encrypting data in transit.

For more information, read the Scalability, Availability, and Failover Guide in the documentation.

ACID Transactions: No Data Is Ever Lost or Corrupted

A key security feature of MarkLogic is ACID transactions—a feature that is lacking in most other NoSQL databases. ACID stands for atomicity, consistency, isolation, and durability. If a database is ACID compliant, it means that reads and writes are durably logged to disk, and strongly isolated from other transactions. Without this feature, you run the risk of encountering data corruption, stale reads, and inconsistent data—all of which are unacceptable for enterprise-grade systems.

ACID Transactions is a security feature. While not often viewed this way, compliance with all of the ACID properties ensures data and transactional integrity (the "I" in "CIA"). ACID is invaluable to ensuring that data is not corrupted in distributed environments, even when there are network partitions or system failures.

MarkLogic satisfies all of the ACID properties by using MVCC (multi-version concurrency control). In an MVCC system, changes are tracked with a timestamp number on each document. The database uses these timestamps to ensure that all users see consistent data.

Here is a summary of the key ways in which MarkLogic achieves all of the ACID properties:

- **Document locks** – MarkLogic protects data during updates and keep transactions from conflicting with one another without impacting reads
- **Timestamps on documents** – MarkLogic ensures a query only sees copies of documents that are valid at the time the query is run (also known as Multi-Version Concurrency Control, or MVCC)
- **Journaling of updates** – Before updates are committed in MarkLogic, they are journaled to ensure transactions can be replayed in the face of system failures
- **Commit process** – MarkLogic ensures data changes happen all at once or not at all, even across multiple hosts

For more information on ACID Transactions, read the section in the Applications Developer's Guide on Understanding Transactions.

Practical Considerations

- MarkLogic is designed to provide high availability and disaster recovery at scale. Follow the steps in the read the Scalability, Availability, and Failover Guide for setting up your system
- Disaster recovery replicas support the same encryption, security roles, auditing and other protections configured in the main cluster
- No special coding is required to enable support for ACID transactions. MarkLogic maintains all ACID properties at the level of the database, a security feature that ensures data is not corrupted

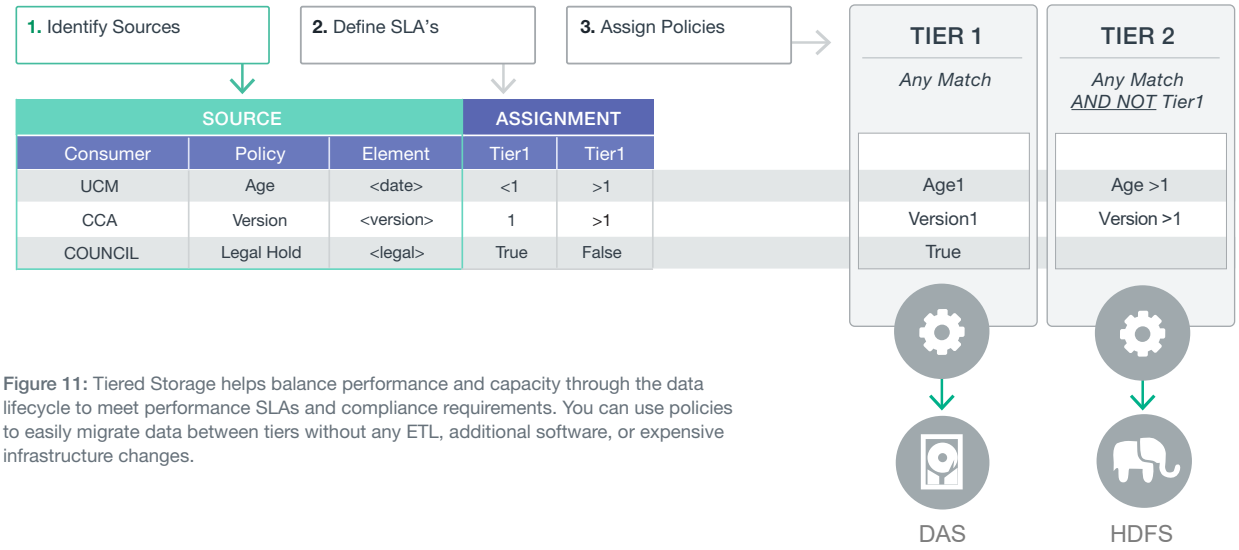


Figure 11: Tiered Storage helps balance performance and capacity through the data lifecycle to meet performance SLAs and compliance requirements. You can use policies to easily migrate data between tiers without any ETL, additional software, or expensive infrastructure changes.

Data Management & Security Policy

MarkLogic enables flexible and comprehensive data governance to meet new and changing guidelines around data retention and management. To reduce the burden on developers and make data governance less complex, MarkLogic provides controls at the level of data and metadata in the database. Data governance-based polices can be implemented in the database, and expensive and time-consuming code changes are not required to change or update policies.

When data-driven governance is established at the data level, not the code or metric level, MarkLogic can manage the specific data constraints. This centralizes data governance and reduces the burden on developers and makes policy execution easier. Examples of basic polices include decisions about how data is stored, archived, backed up, and protected, as well as processes around how data will be accessed and controlled by authorized individuals and organizations.

Tiered Storage, Retention, & Backup: Address Regulatory Compliance

New and changing regulation around data retention and management are challenging to keep up with. Depending on the industry, regulation may require saving data anywhere from 5 to 30 years.

For example, MiFID II requires that companies make records available to customers for five years and up to seven years for regulators. Under the US Law Enforcement 28 CFR 23, "Criminal Intelligence Systems Operating Policies," certain data is only allowed to be stored for 5 years unless it has been updated. On the other hand, the EU GDPR mandates that data may be stored if consent has been granted (consent can also be revoked).

Other regulations concerning data retention include Sarbanes-Oxley (SOX), PIPEDA (Personal Information Protection & Electronic), PHIA (Personal Health Information Act), HIPPA (Health Information Protection Act), and PCI DSS (Payment Card Industry Data Security). The list goes on.

To help address the challenges of managing historical data, MarkLogic has **Tiered Storage** so you can manage data across the lifecycle at different tiers of storage and computation environments. A top-most

tier provides the fastest access to your most critical data, and the lowest tier provides the slowest access to your least critical data. Storage tiers can include any mix of SSD, local disk, SAN, NAS, or even HDFS or Amazon S3 – MarkLogic supports them all.

To route data to different tiers, you can use any query to define which documents reside in each tier and MarkLogic can automatically route data to different tiers based on its age. In this way, Tiered Storage can be used to implement data retention and backup policies.

Tiered Storage carries all of the same security benefits as with any other data stored with MarkLogic. All tiers of storage and backups are secured and encrypted in the same way as the main database. Database backup and restore operations in MarkLogic are distributed over all of the data nodes in a cluster, and provide consistent database-level backups and restores to ensure that data is protected and secured.

Auditing: Monitor Database Activity

Auditing is the monitoring and recording of selected operational actions of application users and administrative users. Auditing is an essential part of almost every regulatory compliance standard and provides many security benefits. It deters users or potential intruders from inappropriate actions and provides the ability to investigate suspicious activity and notify an auditor of inappropriate actions from an unauthorized user.

Auditing is used to verify other security controls as well. For example, consider an audit policy says that no audit events of a certain type should be detected during normal operations. If the system then detects an audit event of that type, it indicates an unexpected condition that could be the result of an inadequate security control. As a result, the security control can then be evaluated and improved.

MarkLogic supports a comprehensive auditing system which logs information about data accesses, security reconfigurations and administrative changes. These logs can be exposed to log file management tools to make monitoring and escalation easy and automated. MarkLogic can also encrypt these logs to reduce leakage.

Bitemporal: Prevent Tampering With Historical Data

Bitemporal support is critical for an organization working in a regulated industry due to its importance in maintaining regulatory compliance.

Most databases are *uni*temporal, and only track one dimension of time (valid time), which makes it difficult to track and audit data changes over time. MarkLogic's *Bitemporal* feature makes it possible to provide a full audit history of data by tracking how data changes along two dimensions of time. In other words, you can answer the question, "What did you know and when did you know it?" MarkLogic is the only NoSQL database with this capability.

Bitemporal support makes it possible to rewind information "as it actually was" (valid time) in combination with "as it was recorded" (system time) at some point-in-time. This capability makes it possible to detect data tampering and to ensure data integrity by greatly enhancing the ability to audit changes to data. With Bitemporal, you can easily and quickly recreate the state of the data at a historical point in time to prove what the data looked like when a certain decision may have been made.

Temporal collections are secure from tampering in MarkLogic. Information cannot be removed and the ability to compare the "latest" information to earlier information supports non-repudiation and lineage.

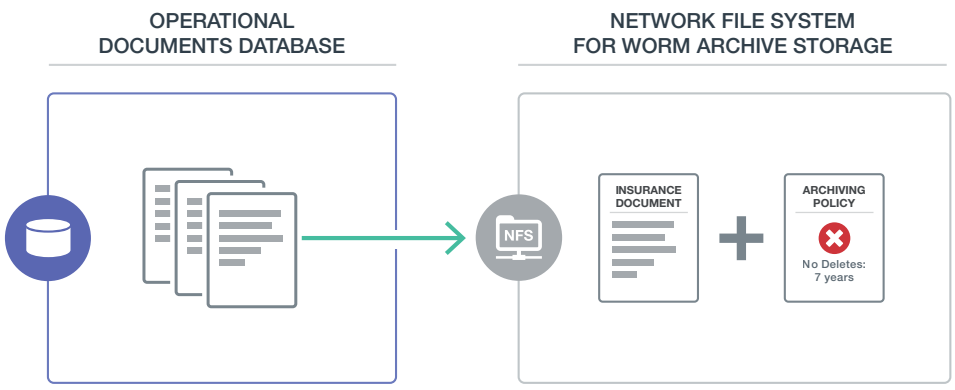


Figure 12: Compliance Archives provides an out-of-the-box mechanism to protect temporal documents against deletion, updates, and wipes using time-based or event-based policies, and save those documents to WORM storage.

Compliance Archive: Address Stringent Data Retention Regulations

Another feature of MarkLogic that helps address regulatory compliance is **Compliance Archive**. The Compliance Archive feature supports e-discovery and legal compliance and meets the requirements for document retention, accuracy, and availability outlined in data privacy regulation (e.g., HIPAA, SEC17a-4, FINRA, etc.).

Compliance Archive achieves this by providing an out-of-the-box mechanism to protect temporal documents against deletion, updates, and wipes using time-based or event-based policies, and save those documents to WORM (Write Once, Read Many) storage with a single operation.

With MarkLogic's Compliance Archive solution, you can:

- Choose the appropriate temporal versioning (uni-temporal or bitemporal) according to your business needs
- Protect and copy to WORM storage using a single operation, based on document metadata
- Ensure that queries are tamper-proof and reflect the information as it was inserted by the application
- Ensure that documents are not deleted or tampered with (even by DBAs and System Administrators)
- Ensure that temporal versions of documents have immutable URIs (document keys)
- Provide traceability by using protection combined with encrypted audit logs
- Ensure that documents are permanently archived and can be recovered
- Continue to have lightning fast performance for search and query across your data

Practical Considerations

- MarkLogic moves a lot of work down to the level of the database, reducing the burden on business application developers. Developers do not need to worry about the details required to manage data storage tiers, security policies, auditing, and temporal data
- The Compliance Archives feature can be used to protect temporal documents against deletion, updates, and wipes. You just need to define the time-based or event-based policies

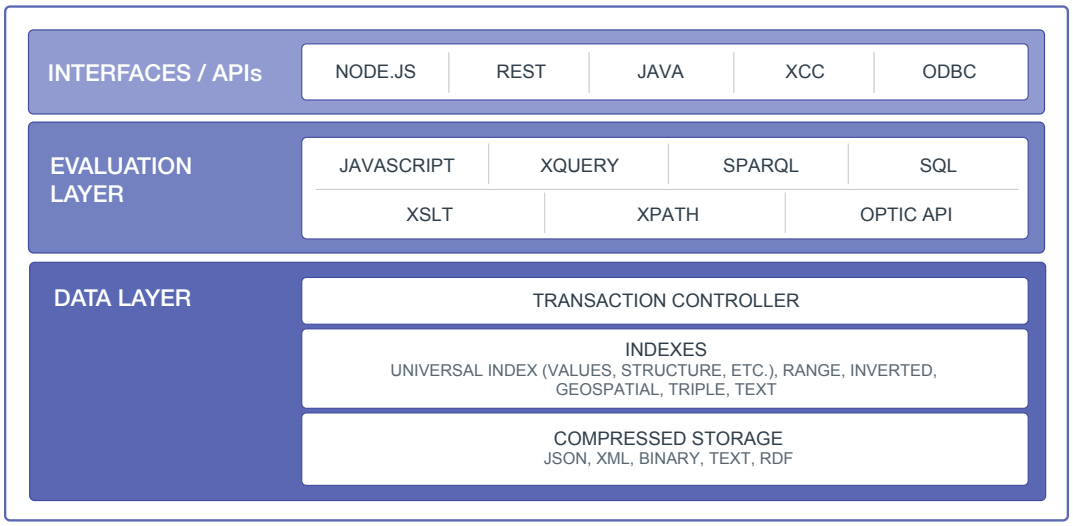


Figure 13: MarkLogic's internal architecture is composed of three main layers, with a variety of interfaces and APIs for developer access.

Data Governance & Developing Applications

MarkLogic has programming APIs so developers can create and execute policies quickly and easily. The security controls and checks when running APIs are transparent to developers. In other databases, developers may have to write code to leverage APIs to get the same level of security that comes with MarkLogic out-of-the-box.

With MarkLogic, you can download a free developer edition and in a few minutes ingest data and have enterprise-grade security applied just by setting up a few roles.

Developing with MarkLogic APIs

MarkLogic exposes the *MarkLogic Security Model* to programming APIs so solutions developers can create and execute policies utilizing all of the security and data protection capabilities in MarkLogic. Policies such as backup, retention, data access, data lifecycle (with Tiered Storage) and authentication can be created utilizing existing MarkLogic APIs.

Policies can be associated with data, metadata, and data attributes so that policies such as those for privacy or compliance can be easily executed. All of the *MarkLogic Security Model* building blocks such as Element Level Security, Redaction, RBAC, Bitemporal, Authentication, and Auditing are available to the application developer to use. Developers do not have to security experts, and security experts can review code faster.

MarkLogic supports secure coding with **JavaScript and XQuery APIs** which utilize all the authorization and authentication controls available to any MarkLogic administrator so that security applications can programmatically secure communications, configuration and data access for the database.

Additional Tools & Interfaces

MarkLogic provides a number of additional tools and interfaces that are helpful for developers. These tools are supported by MarkLogic, and they have gone through the appropriate security testing.

Many of the tools are open source and are available on MarkLogic's GitHub page, which has almost 90 different open source project listed. Other tools are available through the MarkLogic Developer Site. Some of the key supported tools and interfaces are listed here:

- **Data Hub Framework** – The MarkLogic *Data Hub Framework* is a data integration framework and tool-set to quickly and efficiently integrate data from many sources into a single MarkLogic database, and expose that data. If your team wants to build a production-ready Operational Data Hub as quickly as possible, this framework is the best option
- **MLCP (MarkLogic Content Pump)** – *MLCP* is a command line tool for getting data into and out of MarkLogic. It is the most popular tool for bulk loading data into MarkLogic
- **Connector for Hadoop** – *The Connector for Hadoop* is a drop-in extension to Hadoop's MapReduce framework that makes it easy and efficient to communicate with a MarkLogic database from within a MapReduce job

This list is just a quick highlight of some of the key tools that developers use to further leverage the power of MarkLogic and get the benefits of the database quickly and efficiently.

Practical Considerations

- MarkLogic and a community of experienced developers are available to support your development efforts along with extensive and rich set of resources.
- Policies such as backup, retention, data access, data lifecycle (with Tiered Storage) and authentication, as well as any specific policies that are unique to your organization, can all be managed through APIs
- Go to marklogic.com/training to sign up for free training through MarkLogic University
- Take a look at open source projects on MarkLogic's GitHub page
- View MarkLogic documentation and additional tools on the MarkLogic Developer Site

Conclusion

MarkLogic's approach to security relies on an integrated security ecosystem that involves securely building a secure product, ensuring developers can build secure applications using security features that are themselves secure, and by enabling secure deployments. This white paper covers that second aspect—how to build secure applications on the MarkLogic platform (check out the resources listed on the next page that address the other aspects of security).

To support building secure applications, the MarkLogic Security Model provides a layered view of how MarkLogic implements security from the data layer controls up through the authorization, authentication, and authorization controls.

Here is a summary of the key aspects of the MarkLogic Security Model:

- **Secure data at the core** – MarkLogic uses a multi-model approach (document database plus semantics) to storing, managing, and searching data that keeps data, metadata, and relationships all together in one unified system.
- **Encryption at rest** – MarkLogic uses strong encryption, which allows data, configuration information, and logs to be fully encrypted at rest. An external Key Management System (KMS) can be used to provide additional separation of controls.
- **Authorization** – MarkLogic uses Role Based Access Control (RBAC) by default. The privileges, roles, and users, associated data, and other security information is stored in a Security Database. And, security indexes are used to control data access.
- **Authentication** – MarkLogic supports mutual authentication whereby the client also authenticates itself to the server by sending its digital certificate to the server. MarkLogic also supports external authentication using Lightweight Directory Access Protocol (LDAP), Kerberos or certificate-based authentication.
- **Availability** – MarkLogic's distributed, clustered architecture enables powerful features like High Availability (HA), Disaster Recovery (DR), Tiered Storage, and ACID transactions using Multi-Version Concurrency Control (MVCC). All together, these features ensure your data is never lost and is always available.
- **Data management and security policy** – MarkLogic provides a lot of built-in support, including native APIs, policy and rule management and enforcement, auditing, and advanced features such as Bitemporal (time-based audit trail of data).
- **Application development** – All the security capabilities in the *MarkLogic Security Model* are available to developers writing to the MarkLogic platform APIs without the need to be a security expert. And, those APIs are available in the language of their choice (Node.js, Java, REST, XCC, ODBC). MarkLogic also has over 90 open source tools available, many of which are open source and fully supported.

Together, all of these security capabilities make it easier and faster to implement stronger security across the full lifecycle of data. It means your data is less vulnerable and your organization is less at risk because the important task of data security is maintained where it should be—in the database.

There are many aspects to securing applications that are not addressed here. We encourage you to take a look at our other resources on security or contact us for more information.

Key Resources

Presentation – Data Security in Practice
http://www.marklogic.com/resources/data-security-practice/

Understanding and Using Security Guide
https://docs.marklogic.com/guide/security

White Paper – Top Data Security Concerns When Integrating Data
http://www.marklogic.com/resources/top-data-security-soncerns-integrating-data

White Paper – Building Security Into MarkLogic
http://www.marklogic.com/resources/building-security-marklogic/

White Paper – How to Deploy MarkLogic Securely
http://www.marklogic.com/resources/deploy-marklogic-securely/

MARKLOGIC CORPORATION
999 Skyway Road, Suite 200 San Carlos, CA 94070
+1 650 655 2300 | +1 877 992 8885 | www.marklogic.com | sales@marklogic.com

999 Skyway Road, Suite 200 San Carlos, CA 94070
+1 650 655 2300 | +1 877 992 8885
www.marklogic.com | sales@marklogic.com