
MarkLogic Server

Understanding and Using Security Guide

MarkLogic 6
September, 2012

Last Revised: 6.0-1, September, 2012

Table of Contents

Understanding and Using Security Guide

1.0	Introduction to Security	5
1.1	Security Overview	5
1.1.1	Authentication and Access Control	5
1.1.2	Authorization	6
1.1.3	Administration	6
1.2	MarkLogic Security Model	6
1.2.1	Role-Based Security Model (Authorization)	6
1.2.2	Access Control With the Security Database	8
1.2.3	Security Administration	9
1.3	Terminology	9
1.3.1	User	9
1.3.2	Role	9
1.3.3	Execute Privilege	9
1.3.4	URI Privilege	10
1.3.5	Permission	10
1.3.6	Amp	10
2.0	Role-Based Security Model	11
2.1	Understanding Roles	11
2.1.1	Assigning Privileges to Roles	11
2.1.1.1	Execute Privileges	11
2.1.1.2	URI Privileges	12
2.1.2	Associating Permissions With Roles	12
2.1.3	Default Permissions in Roles	12
2.1.4	Assigning Roles to Users	12
2.1.5	Roles, Privileges, Document Permissions, and Users	13
2.2	The admin and security Roles	14
2.3	Example—Introducing Roles, Users and Execute Privileges	14
3.0	Protecting Documents	16
3.1	Creating Documents	16
3.1.1	URI Privileges	16
3.1.2	Built-In URI Execute Privileges	17
3.2	Document Permissions	17
3.2.1	Capabilities Associated Through Permissions	18
3.2.1.1	Read	18
3.2.1.2	Update	18
3.2.1.3	Insert	18

3.2.1.4	Execute	19
3.2.2	Setting Document Permissions	19
3.3	Securing Collection Membership	19
3.4	Default Permissions	20
3.5	Example—Using Permissions	20
3.5.1	Setting Permissions Explicitly	21
3.5.2	Default Permission Settings	21
4.0	Compartment Security	24
4.1	Understanding Compartment Security	24
4.2	Configuring Compartment Security	24
4.3	Example—Compartment Security	25
4.3.1	Create Roles	25
4.3.2	Create Users	26
4.3.3	Create the Documents and Add Permissions	26
4.3.4	Test It Out	27
5.0	Protecting XQuery Functions With Privileges	28
5.1	Built-In MarkLogic Execute Privileges	28
5.2	Protecting Your XQuery Code with Execute Privileges	28
5.2.1	Using Execute Privileges	29
5.2.2	Execute Privileges and App Servers	29
5.2.3	Creating and Updating Collections	30
5.3	Temporarily Increasing Privileges with Amps	30
6.0	Authenticating Users	31
6.1	Users	31
6.2	Types of Authentication	31
6.2.1	Basic	31
6.2.2	Digest	32
6.2.3	Digest-Basic	32
6.2.4	Limitations of Digest and Basic Authentication	32
6.2.5	Application Level	32
7.0	Administering Security	33
7.1	Overview of the Security Database	33
7.2	Associating a Security Database With a Documents Database	34
7.3	Managing and Using Objects in the Security Database	35
7.3.1	Using the Admin Interface	35
7.3.2	Using the security.xqy Module Functions	35
7.4	Backing Up the Security Database	35
7.5	Example: Using the Security Database in Different Servers	36
8.0	Auditing	39

8.1	Why Is Auditing Used?	39
8.2	MarkLogic Auditing	40
8.3	Configuring Auditing	40
8.4	Best Practices	40
9.0	Designing Security Policies	41
9.1	Research Your Security Requirements	41
9.2	Plan Roles and Privileges	41
10.0	Sample Security Scenarios	43
10.1	Protecting the Execution of XQuery Modules	43
10.2	Choosing the Access Control for an Application	44
10.2.1	Open Access, No Log In	44
10.2.2	Providing Uniform Access to All Authenticated Users	44
10.2.3	Limiting Access to a Subset of Users	45
10.2.4	Using Custom Login Pages	46
10.2.5	Access Control Based on Client IP Address	47
10.3	Implementing Security for a Read-Only User	51
10.3.1	Steps For Example Setup	51
10.3.2	Troubleshooting Tips	52
11.0	Securing Your Production Deployment	53
11.1	Add Password Protections	53
11.2	Adhere to the Principle of Least Privilege	53
11.3	Infrastructure Hardening	54
11.3.1	OS-Level Restrictions	54
11.3.2	Network Security	54
11.3.3	Port Management	54
11.3.4	Physical Access	54
11.4	Implement Auditing	55
11.5	Develop and Enforce Application Security	55
11.6	Use MarkLogic Security Features	55
11.7	Read About Security Issues	55
12.0	Technical Support	56
	Combined Product Notices	57
	COPYRIGHT	57
	TRADEMARK NOTICE	69

1.0 Introduction to Security

When you create systems that store and retrieve data, it is important to protect the data from unauthorized use, disclosure, modification or destruction. Ensuring that users have the proper authority to see the data, load new data, or update existing data is an important aspect of application development. Do all users need the same level of access to the data and to the functions provided by your applications? Are there subsets of users that need access to privileged functions? Are some documents restricted to certain classes of users? The answers to questions like these help provide the basis for the security requirements for your application.

MarkLogic Server includes a powerful and flexible role-based security model to protect your data according to your application security requirements. There is always a trade-off between security and usability. When a system has no security, then it is open to malicious or unmalicious unauthorized access. When a system is too tightly secured, it might become difficult to use successfully. Before implementing your application security model, it is important to understand the core concepts and features in the MarkLogic Server security model. This chapter introduces the MarkLogic Server security model and includes the following sections:

- [Security Overview](#)
- [MarkLogic Security Model](#)
- [Terminology](#)

1.1 Security Overview

This section provides an overview of the three main principles used in MarkLogic Server security:

- [Authentication and Access Control](#)
- [Authorization](#)
- [Administration](#)

1.1.1 Authentication and Access Control

Authentication is the process of verifying user credentials for a named user. Authentication makes sure you are who you say you are. Users are typically authenticated with a username and password. Authentication verifies user credentials and associates an application session with the authenticated user. Every request to MarkLogic Server is issued from an authenticated user. Authentication, by itself, does not grant access or authority to perform specific actions. There are several ways to set up server authentication in MarkLogic Server.

Authentication by username and password is only part of the story. You might grant access to users based on something other than identity, something such as the originating IP address for the requests. Restricting access based on something other than the identity of the user is generally referred to as *access control*.

For details on authentication, see “Authenticating Users” on page 31.

1.1.2 Authorization

Authorization provides the mechanism to control document access, XQuery code execution, and document creation. For an authenticated user, authorization determines what you are allowed to do. For example, authorization is what allows the user named *Melanie* to read and update a document, allows the user named *Roger* to only read the document, and prevents the user named *Hal* from knowing the document exists at all. In MarkLogic Server, authorization is used to protect documents stored in a database and to protect the execution of XQuery code. For details on authorization in MarkLogic Server, see “Protecting Documents” on page 16 and “Protecting XQuery Functions With Privileges” on page 28.

1.1.3 Administration

Administration is the process of defining, configuring, and managing the security objects, such as users, roles, privileges, and permissions that implement your security policies. For details on security administration procedures in MarkLogic Server, see “Security Administration” on page 9 and the *Administrator’s Guide*.

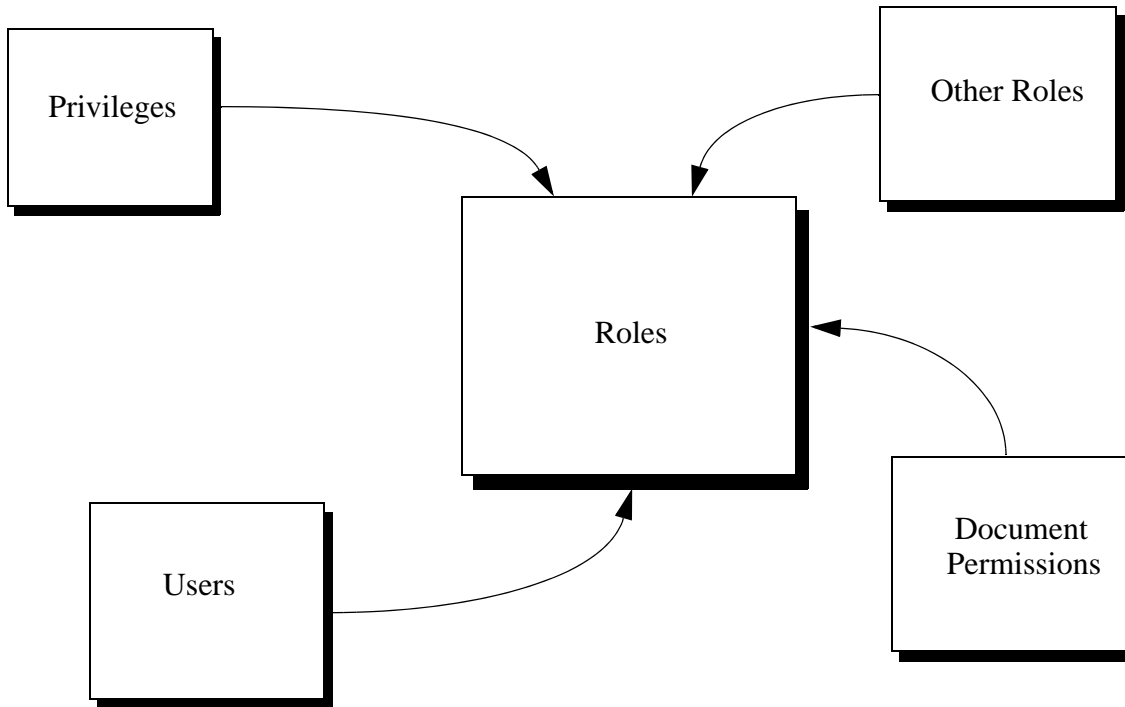
1.2 MarkLogic Security Model

The MarkLogic Server security model is flexible and enables you to set up application security with the level of granularity needed by your security requirements. This section contains the following topics:

- [Role-Based Security Model \(Authorization\)](#)
- [Access Control With the Security Database](#)
- [Security Administration](#)

1.2.1 Role-Based Security Model (Authorization)

Roles are the central point of authorization in the MarkLogic Server security model. Privileges, users, other roles, and document permissions all relate directly to roles. The following conceptual diagram shows how each of these entities points into one or more roles.



There are two types of privileges: 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.

Privileges are assigned to zero or more roles, roles are assigned to zero or more other roles, and users are assigned to zero or more roles. A privilege is like a door and, when the door is locked, you need to have the key to the door in order to open it. If the door is unlocked (no privileges), then you can walk right through. The keys to the doors are distributed to users through roles; that is, if a user inherits a privilege through the set of roles to which she is assigned, then she has the keys to unlock those inherited privileges.

Permissions are used to protect documents. Permissions are assigned to documents, either at load time or as a separate administrative action. Each permission is a combination of a role and a capability (read, insert, update, execute).

Permission

Role	Capability (read, insert, update, or execute)
------	---

Users assigned the role corresponding to the permission have the ability to perform the capability. You can set any number of permissions on a document.

Capabilities represent actions that can be performed. There are four capabilities in MarkLogic Server:

- read
- insert
- update
- execute

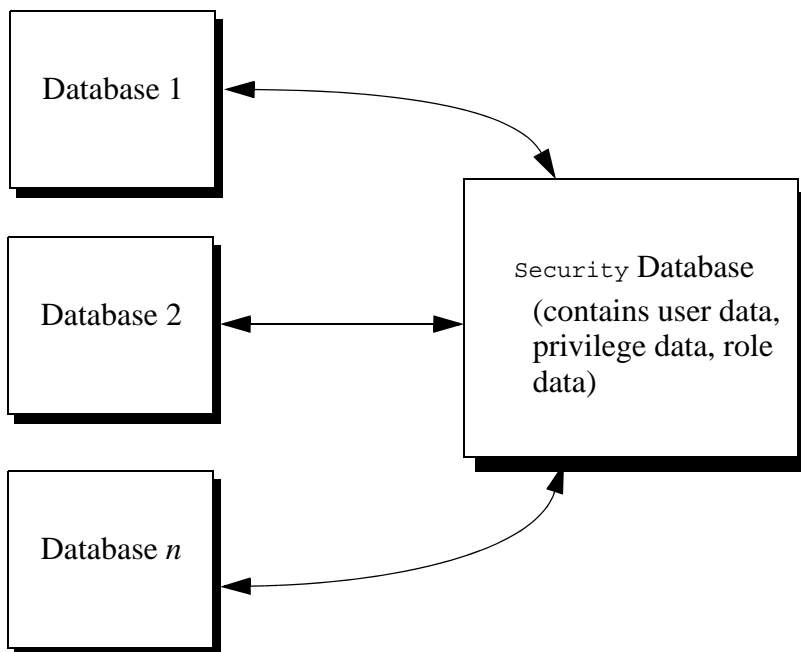
Users inherit the sum of the privileges and permissions from their roles.

For more details on how roles work in MarkLogic Server, see “Role-Based Security Model” on page 11. For more details on privileges and permissions, see “Protecting Documents” on page 16.

1.2.2 Access Control With the Security Database

MarkLogic Server uses a *security database* to store the user data, privilege data, role data, and other security information. Each database in MarkLogic Server references a security database. A database named `security`, which functions as the default security database, is created as part of the installation process.

The following figure shows that many databases can be configured to use the same security database for authentication and authorization.



The security database is accessed to authenticate users and to control access to documents. For details on authentication, the security database, and ways to administer objects in the security database, see “Authenticating Users” on page 31 and “Administering Security” on page 33.

1.2.3 Security Administration

MarkLogic Server administrators are privileged users who have the authority to perform tasks such as creating, deleting, modifying users, roles, privileges, and so on. These tasks change or add data in the security database. Users who perform these tasks must have the `security` role, either explicitly or by inheriting it from another role (for example, from the `admin` role). Typically, users who perform these tasks have the `admin` role, which provides the authority to perform any tasks in the database. Use caution when assigning users to the `security` and/or `admin` roles; users who are assigned the `admin` role can perform any task on the system, including deleting data.

MarkLogic Server provides two ways to administer security:

- Admin Interface
- XQuery security administration functions

For details on administering security, see “Administering Security” on page 33.

1.3 Terminology

This section defines the following terms, which are used throughout the security documentation:

- [User](#)
- [Role](#)
- [Execute Privilege](#)
- [URI Privilege](#)
- [Permission](#)
- [Amp](#)

1.3.1 User

A *user* is a named entity used to authenticate a request to an HTTP, WebDAV, ODBC, or XDBC server. For details on users, see “Authenticating Users” on page 31.

1.3.2 Role

A *role* is a named entity that provides authorization privileges and permissions to other roles or to users. You can assign roles to other roles (which can in turn include assignments to other roles, and so on). Roles are the fundamental building blocks that you use to implement your security policies. For details on roles, see “Role-Based Security Model” on page 11.

1.3.3 Execute Privilege

An *execute privilege* provides the authority to perform a protected action. Examples of protected actions are the ability to execute a specific user-defined function, the ability to execute a built-in function (for example, `xdmp:document-insert`), and so on. For details on execute privileges, see “Protecting XQuery Functions With Privileges” on page 28.

1.3.4 URI Privilege

A *URI privilege* provides the authority to create documents within a base URI. When a URI privilege exists for a base URI, only users assigned to roles that have the URI privilege can create documents with URIs starting with the base string. For details on URI privileges, see “Protecting Documents” on page 16.

1.3.5 Permission

A *permission* provides a role with the capability to perform certain actions (*read*, *insert*, *update*, *execute*) on a document or a collection. Permissions consist of a role and a capability. Permissions are assigned to documents and collections. For details on permissions, see “Protecting Documents” on page 16.

1.3.6 Amp

An *amp* provides a user with the additional authorization to execute a specific function by temporarily giving the user additional roles. For details on amps, see “Temporarily Increasing Privileges with Amps” on page 30.

2.0 Role-Based Security Model

MarkLogic Server uses a role-based security model. Each security entity is associated with a role. This chapter describes the role-based security model and includes the following sections:

- [Understanding Roles](#)
- [The admin and security Roles](#)
- [Example—Introducing Roles, Users and Execute Privileges](#)

2.1 Understanding Roles

As described in “Role-Based Security Model (Authorization)” on page 6, roles are the central point of authorization in MarkLogic Server. This section describes how the other security entities relate to roles, and includes the following sections:

- [Assigning Privileges to Roles](#)
- [Associating Permissions With Roles](#)
- [Default Permissions in Roles](#)
- [Assigning Roles to Users](#)
- [Roles, Privileges, Document Permissions, and Users](#)

2.1.1 Assigning Privileges to Roles

Execute privileges control access to XQuery code. URI privileges control access to creating documents in a given URI range. You associate roles with privileges by assigning the privileges to the roles.

2.1.1.1 Execute Privileges

Execute privileges allow developers to control authorization for the execution of an XQuery function. If an XQuery function is protected by an execute privilege, the function must include logic to check if the user executing the code has the necessary execute privilege. That privilege is assigned to a user through a role that includes the specific execute privilege. There are many execute privileges pre-defined in the security database to control execution of built-in XQuery functions.

For more details on execute privileges, see “Protecting XQuery Functions With Privileges” on page 28.

2.1.1.2 URI Privileges

URI privileges control authorization for creation of a document with a given URI prefix. To create a document with a prefix that has a URI privilege associated with it, a user must be part of a role that has the needed URI privilege.

For more details on how URI privileges interact with document creation, see “Protecting Documents” on page 16.

2.1.2 Associating Permissions With Roles

Permissions are security characteristics of documents that associate a role with a capability. The capabilities are the following:

- `read`
- `insert`
- `update`
- `execute`

Users gain the authority to perform these capabilities on a document if they are assigned a role to which a permission is associated.

For more details on how permissions interact with documents, see “Document Permissions” on page 17.

2.1.3 Default Permissions in Roles

Roles are one of the places where you can specify *default permissions*. If permissions are not explicitly specified when a document is created, the default permissions of the user creating the document are applied. The system determines the default permissions for a user based on the user’s roles. The total set of default permissions is derived from the user’s roles and all inherited roles.

For more details on how default permissions interact with document creation, see “Default Permissions” on page 20.

2.1.4 Assigning Roles to Users

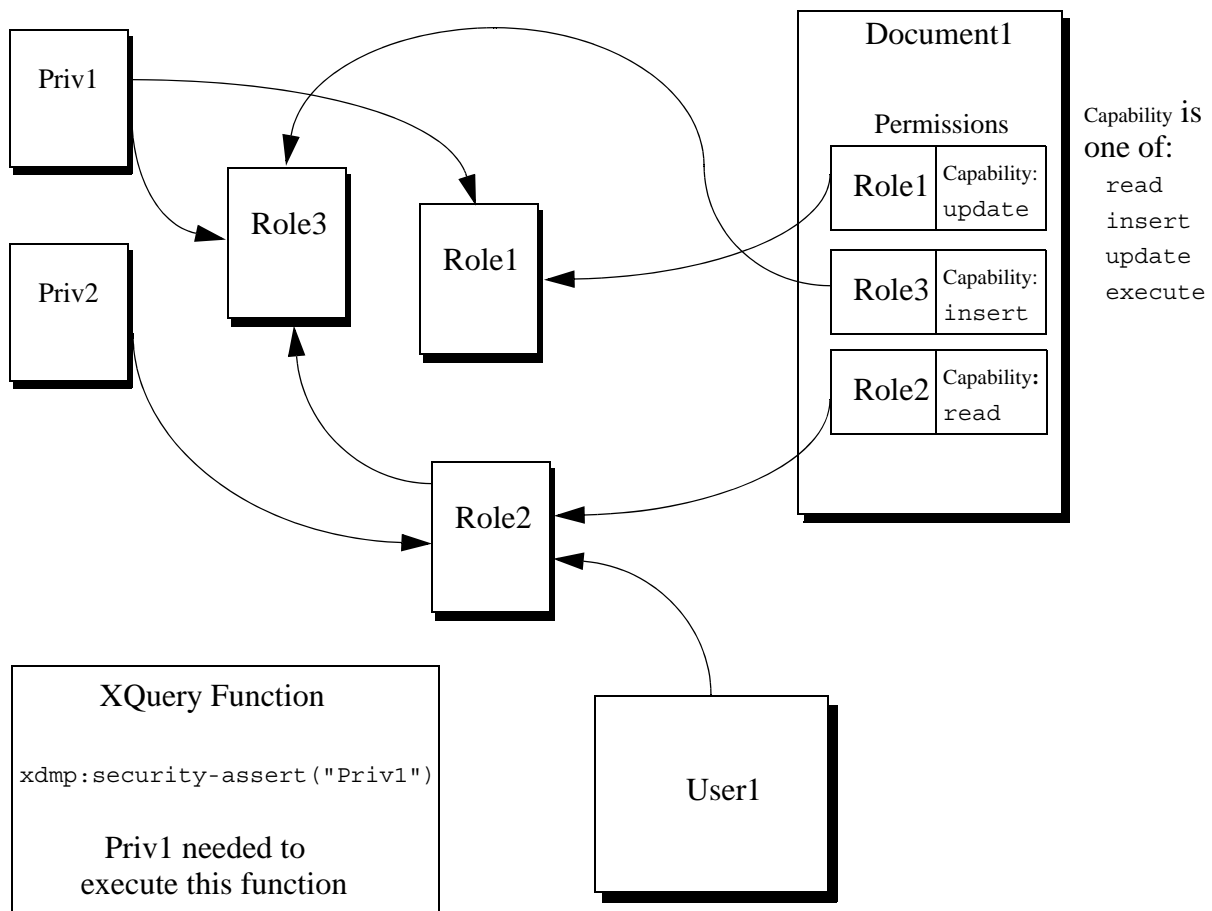
Users are authenticated against the security database configured for the database being accessed. Roles are the mechanism by which authorization information is derived. You assign roles to a user. The roles provide the user with a set of privileges and permissions that grant the authority to perform actions against code and documents. At any given time, a user *possesses* a set of privileges and default permissions that is the sum of the privileges and default permissions inherited from all of the roles currently assigned to that user.

Use the Admin Interface to display the set of privileges and default permissions for a given user; do not try and calculate it yourself as it can easily get fairly complex when a system has many roles. To display a user’s security settings, use Admin Interface > Security > User > Describe. You need to select a specific user to see the Describe tab.

For more details on users, see “Authenticating Users” on page 31.

2.1.5 Roles, Privileges, Document Permissions, and Users

Privileges, document permissions, and users all interact with roles to define your security policies. The following diagram shows an example of how these entities interact.



Notice how all of the arrows point into the roles; that is because the roles are the center of all security administration in MarkLogic Server. In this diagram, User1 is part of Role2, and Role2 inherits Role3. Therefore, even though User1 has only been assigned Role2, User1 possesses all of the privileges and permissions from both Role2 and Role3. Following the arrows pointing into Role2 and Role3, you can see that the user possesses Priv1 and Priv2 based on the privileges assigned to these roles and insert and read capabilities based on the permissions applied to Document1.

Because `User1` possesses `Priv1` (based on role inheritance), `User1` is able to execute code protected with a `xdmp:security-assert("Priv1", "execute")` call; users who do not have the `Priv1` privilege can not execute such code.

2.2 The admin and security Roles

MarkLogic Server has a special role named `admin`. The `admin` role has full authority to do everything in MarkLogic Server, regardless of the permissions or privileges set. In general, the `admin` role is only for administrative activities and should not be used to load data and run applications. Use extreme caution when assigning users the `admin` role, because it gives them the authority to perform any activity in MarkLogic, included adding or deleting users, adding or deleting documents, changing passwords, and so on.

MarkLogic Server also has a built-in role named `security`. Users who are part of the `security` role have execute privileges to perform security-related tasks on the system using the functions in the `security.xqy` Library Module.

The `security` role does not have access to the Admin Interface. To access the Admin Interface, a user must have the `admin` role. The `security` role provides the privileges to execute functions in the `security.xqy` module, which has functions to perform actions such as creating users, creating roles, and so on. For details on managing security objects programmatically, see [Creating and Configuring Roles and Users](#) and [User Maintenance Operations](#) in the *Scripting Administrative Tasks Guide*.

2.3 Example—Introducing Roles, Users and Execute Privileges

Consider a simple scenario with two roles – `engineering` and `sales`. The `engineering` role is responsible for making widgets and has privileges needed to perform activities related to making widgets. The `sales` role is responsible for selling widgets and has privileges to perform activities related to selling widgets.

To begin, create two roles in MarkLogic Server named `engineering` and `sales` respectively.

The `engineering` role needs to be able to make widgets. You can create an execute privilege with the name `make-widget`, and action URI `http://widget.com/make-widget` to represent that privilege. The `sales` role needs to sell widgets, so you create an execute privilege with the name `sell-widget` and action URI `http://widget.com/sell-widget` to represent that privilege.

Note: Names for execute privileges are used only as display identifiers in the Admin Interface. The action URIs are used within XQuery code to identify the privilege.

Ron is an engineer in your company so you create a user for Ron and assign the `engineering` role to the newly created user. Emily is an account representative so you create a user for Emily and assign her the `sales` role.

In your XQuery code, use the `xdmp:security-assert` function to ensure that only engineers make widgets and only account representatives sell widgets. For example:

```
xquery version "1.0-m1"
define function make-widget(...) as ...
{
  xdm:security-assert("http://widget.com/make-widget",
    "execute"), make widget...
}
```

If Ron is logged into the application and executes the `make-widget()` function, `xdmp:security-assert("http://widget.com/make-widget", "execute")` succeeds since Ron is of the engineering role which has the execute privilege to make widgets.

If Emily attempts to execute the `make-widget` function, the `xdmp:security-assert` function call throws an exception. You can catch the exception and handle it with a `try/catch` in the code. If the exception is not caught, the transaction that called this function is rolled back.

Some functions are common to several protected actions. You can protect such a function with a single `xdmp:security-assert` call by providing the appropriate action URIs in a list. For example, if a user needs to execute the `count-widgets` function when making or selling widgets, you might protect the function as follows:

```
xquery version "1.0-m1"
define function count-widgets(...) as ...
{
  xdm:security-assert( ("http://widget.com/make-widget",
    "http://widget.com/sell-widget"), "execute"),
  count-widget...
}
```

If there is a function that requires more than one privilege before it can be performed, place the `xdmp:security-assert` calls sequentially. For example, if you need to be a manager in the sales department to give discounts when selling the widgets, you can protect the function as follows:

```
xquery version "1.0-m1"
define function discount-widget(...) as ...
{
  xdm:security-assert( "http://widget.com/sell-widget",
    "execute"),
  xdm:security-assert( "http://widget.com/change-price",
    "execute"),
  discount widget...
}
```

where `http://widget.com/change-price` is an action URI for a `change-price` execute privilege assigned to the `manager` role. A user needs to have the `sales` role and the `manager` role, which provides the user with the `sell-widget` and `change-price` execute privileges, to be able to execute this function.

3.0 Protecting Documents

The MarkLogic Server security model has a set of tools you can use to control access to documents. These authorization tools control creating, inserting into, updating, and reading documents in a database. This chapter describes those tools and includes the following sections:

- [Creating Documents](#)
- [Document Permissions](#)
- [Securing Collection Membership](#)
- [Default Permissions](#)
- [Example—Using Permissions](#)

3.1 Creating Documents

To create a document in a MarkLogic Server database, a user must possess the needed privileges to create a document with a given URI. The ability to create documents based on the URI is controlled with URI privileges and with two built-in execute privileges (`any-uri` and `unprotected-uri`). To possess a privilege, the user must be part of a role (either directly or indirectly, through role inheritance) to which the privilege is assigned. This section describes these different privileges.

3.1.1 URI Privileges

URI privileges control the ability to create a new document with a given URI prefix. Using a URI privilege for a given URI protects that URI from new document creation; only users possessing the URI privilege can create a new document with the prefix.

For example, the screenshot below shows a URI privilege with `/widget.com/sales/` as the protected URI. Any URI with `/widget.com/sales/` as the prefix is protected. Users must be part of the `sales` role to create documents with URIs beginning with this prefix. In this example, you need this URI privilege (or a privilege with at least as much authority) to create a document with the URI `/widget.com/sales/my_process.xml`.

New URI Privilege ok cancel

uri privilege -- *Privilege representation.*

privilege name
 Privilege name (unique)
Required. You must supply a value for privilege-name.

uri
 A URI to protect.
Required. You must supply a value for action.

roles -- *The roles assigned.*

- admin
- admin-builtins
- domain-management
- filesystem-access
- merge
- pipeline-execution
- pipeline-management
- read
- sales

3.1.2 Built-In URI Execute Privileges

The following built-in execute privileges control the creation of URIs:

- any-uri
- unprotected-uri

The `any-uri` privilege provides the authority to create a document with any URI in the database, even if the URI is protected with a URI privilege. The `unprotected-uri` privilege provides the authority to create a document at any URI in the database except for URIs that are protected with a URI privilege.

3.2 Document Permissions

Permissions set on a document define access to capabilities (`read`, `insert`, `update`, and `execute`) for that document. Each permission consists of a capability and a role. This section describes how to set permissions on a document. It includes the following subsections:

- [Capabilities Associated Through Permissions](#)
- [Setting Document Permissions](#)

3.2.1 Capabilities Associated Through Permissions

Document permissions pair a role with a capability to perform some action on a document. You can add multiple permissions to a document. If a user is part of a role (either directly or through inheriting the role) specified as part of a document permission, then the user has that capability for the given document. Each permission associates a role with one of the following capabilities:

- [Read](#)
- [Update](#)
- [Insert](#)
- [Execute](#)

3.2.1.1 Read

The `read` capability provides the authority to see the content in the document. Being able to see the content does not allow you to modify the document.

3.2.1.2 Update

The `update` capability provides the authority to modify content in the document or delete the document. However, `update` does not provide the authority to read the document. Reading the document requires the `read` capability. Users with `update` capability, but not `read` capability, can call the `xdmp:document-delete` and `xdmp:document-insert` functions successfully. However, node update functions, such as `xdmp:node-replace`, `xdmp:node-delete`, and `xdmp:node-insert-after`, cannot be called successfully. Node update functions require a node from the document as a parameter. If a user cannot read the document, he cannot access the node in the document and supply it as a parameter.

There is a way to get around the issue with node update functions. The `update` capability provides the authority to change the permissions on a document. Therefore, you can use the `xdmp:document-add-permissions` function to add a new permission to the document with `read` capability for a given role. A user with both `read` and `update` capabilities can call node update functions successfully.

3.2.1.3 Insert

The `insert` capability provides a subset of the `update` capability. The `insert` capability provides the authority to add new content to the document. The `insert` capability by itself does not allow a user to change existing content or remove an existing document (for example, calls to `xdmp:document-insert` and `xdmp:document-delete` on an existing document fail). Furthermore, you need `read` capability on the document to perform actions that use any of the node insert functions (`xdmp:node-insert-before`, `xdmp:node-insert-after`, `xdmp:node-insert-child`), as explained above in the description for `update`. Therefore, a permission with an `insert` capability must be paired with a permission with a `read` capability to be useful.

3.2.1.4 Execute

The `execute` capability provides the authority to execute application code contained in that document, if the document is stored in a database which is configured as a modules database. Users without permissions for the `execute` capability on a stored module, are not able to execute that module.

3.2.2 Setting Document Permissions

When you create documents in a database, you must think about setting permissions on the document. If a document has no permission set on it, no one, other than users with the `admin` role, can read, update, insert, or delete it. Additionally, non-admin users must add update permissions on documents when creating them; attempts to create a document without at least one update permission result in an `XDMP-MUSTHAVEUPDATE` exception.

You set document permissions in the following ways:

- Explicitly set permissions on a document at load time (as a parameter to `xdmp:document-load` OR `xdmp:document-insert`, for example).
- Explicitly set and remove permissions on a document using the following functions:
 - `xdmp:document-add-permissions`
 - `xdmp:document-set-permissions`
 - `xdmp:document-remove-permissions`
- Implicitly set permissions when the document is created based on the default permissions of the user who creates the documents. Permissions are applied to a document at document creation time based on the default permissions of the user who creates the document.

For examples of setting permissions on documents, see “Example—Using Permissions” on page 20.

3.3 Securing Collection Membership

You can also secure membership in collections by assigning permissions to collections. To assign permissions to collections, you must use the Admin Interface or the `security.xqy` Library Module functions. You cannot assign permissions to collections implicitly with default permissions.

For more information about permissions on collections, see [Collections and Security](#) in the *Search Developer's Guide*.

3.4 Default Permissions

When a document is created, it is initialized with a set of permissions. If permissions are not explicitly set (by using `xdmp:document-load` or `xdmp:document-insert`, for example), then the permissions are set to the *default permissions*. The default permissions are determined based on the roles assigned (both 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 are creating documents in a database, it is important to configure default permissions for the roles assigned to that user. Without default permissions, it is easy to create documents that no users (except those with the `admin` role) can read, update, or delete.

3.5 Example—Using Permissions

It is important to consider document permissions when you load content into a database, whether you load data using the built-in functions (for example, `xdmp:document-load` or `xdmp:document-insert`), WebDAV (for example, dragging and dropping files into a WebDAV folder), the REST API, the Java API, or a custom program. In each case, setting permissions is necessary, whether explicitly or by taking advantage of default permissions. This example shows several ways of setting permissions on documents.

Suppose that Ron, of the `engineering` role, is given the task to create a document to describe new features that will be added to the next version of the widget. Once the document is created, other users with the `engineering` role contribute to the document and add the features they are working on. Ian, of the `engineering-manager` role, decides that users of the `engineering` role should only be allowed to read and add to the document. This enables Ian to control the process of removing or changing features in the document. To implement this security model, the document should be created with `read` and `insert` permissions for the `engineering` role, and `read` and `update` permissions for the `engineering-manager` role.

There are two ways to apply permissions to documents at creation time:

- [Setting Permissions Explicitly](#)
- [Default Permission Settings](#)

3.5.1 Setting Permissions Explicitly

Assume that the following code snippet is executed as user `Ron` of the `engineering` role. The code inserts a document with the following permissions:

- `read` and `insert` permissions for the `engineering` role
- `update` and `read` permissions for the `engineering-manager` role

```
...
xdmp:document-insert("/widget.com/engineering/features/2004-q1.xml",
  <new-features>
    <feature>
      <name>blue whistle</name>
      <assigned to>Ron</assigned to>
      ...
    </feature>
    ...
  </new-features>,
  (xdmp:permission("engineering", "read"),
   xdmp:permission("engineering", "insert"),
   xdmp:permission("engineering-manager", "read"),
   xdmp:permission("engineering-manager", "update")))
...
```

If you specify permissions to the function call explicitly, as shown above, those permissions override any default permission settings associated with the user (through user settings and role inheritance).

3.5.2 Default Permission Settings

If there is a set of permission requirements that meets the needs of most application scenarios, MarkLogic recommends creating the appropriate default permission settings at the role or user level. This avoids having to explicitly create and set document permissions each time you call `xdmp:document-load` or `xdmp:document-insert`.

Default permission settings that apply to a user, either through a role or through the user definition, are important if you are loading documents using a WebDAV client. When you drag and drop files into a WebDAV folder, the permissions are automatically set based on the default permissions of the user logged into the WebDAV client. For more information about WebDAV servers, see [WebDAV Servers](#) in the *Administrator's Guide*.

The following screenshot shows a portion of the Admin Interface for the `engineering` role. It shows `insert` and `update` capabilities being added to the `engineering` role's default permissions.

default permissions -- *The default set of permissions used in document creation.*

role name (capability)

No Current Permissions

[add] engineering read

[add] engineering insert

[add] read

more permissions

A user's set of default permissions is additive; it is the aggregate of the default permissions for all of the user's role(s) as well as for the user himself. Below is another screenshot of a portion of a User configuration screen for Ron. It shows read and update capabilities being added to the `engineering-manager` role as Ron's default permissions at the user level.

default permissions -- *The default set of permissions used in document creation.*

role name + capability

engineering-manager update

engineering-manager read

read

more permissions

Note: Ron has the `engineering` role and does not have the `engineering-manager` role. A user does not need to have a certain role in order to specify that role in its default permission set.

You can also use a hybrid of the two methods described above. Assume that `read` and `insert` capabilities for the `engineering` role are specified as default permissions for the `engineering` role as shown in the first screenshot. However, `update` and `read` capabilities are not specified for the `engineering-manager` at the user or `engineering` role level.

Further assume that the following code snippet is executed by Ron. It achieves the desired objective of giving the `engineering-manager` role `read` and `update` capabilities on the document, and the `engineering` role `read` and `insert` capabilities.

```

...
xdmp:document-insert ("/widget.com/engineering/features/2004-q1.xml",
  <new-features>
    <feature>
      <name>blue whistle</name>
      <assigned to>Ron</assigned to>
      ...
    </feature>
    ...
  </new-features>,
  (xdmp:default-permissions(),
   xdmp:permission("engineering-manager", "read")
   xdmp:permission("engineering-manager", "update")))
...

```

The `xdmp:default-permissions` function returns Ron's default permissions (from the role level in this example) of `read` and `insert` capabilities for the `engineering` role. The `read` and `update` capabilities for the `engineering-manager` role are then added explicitly as function parameters.

Note: The `xdmp:document-insert` function performs an update (rather than a create) function if a document with the specified document URI already exists. Consequently, if Ron calls the `xdmp:document-insert` function the second time with the same document URI, the call fails since Ron does not have update capability on the document.

Suppose that Ian, of the `engineering-manager` role, decides to give users of the `sales` role `read` permission on the document. (He wisely withholds `update` or `insert` capability or there will surely be an explosion of features!) The code snippet below shows how to add permissions to a document after it has been created.

```

...
xdmp:document-add-permissions (
  "/widget.com/engineering/features/2004-q1.xml",
  xdmp:permission("sales", "read"))
...

```

The `update` capability is needed to add permissions to a document. Therefore, the code snippet only succeed if it is executed by Ian, or another user of the `engineering-manager` role. This prevents Ron from giving Emily, his buddy in sales, `insert` capability on the document.

Note: Changing default permissions for a role or a user does not affect the permissions associated with existing documents. To change permissions on existing documents, you need to use the permission update functions. See the documentation for the MarkLogic Built-In Functions in *MarkLogic XQuery and XSLT Function Reference* for more details.

4.0 Compartment Security

The MarkLogic Server includes an extension to the security model called compartment security. Compartment security allows you to specify more complex security rules on documents.

To enable compartment security, a license key that includes compartment security is required. For details on purchasing a compartment security license, contact your MarkLogic sales representative.

This chapter describes compartment security and includes the following sections:

- [Understanding Compartment Security](#)
- [Configuring Compartment Security](#)
- [Example—Compartment Security](#)

4.1 Understanding Compartment Security

A *compartment* is a name associated with a role. You specify 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. 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, `compartment1` and `compartment2`, respectively), then the permissions are checked using AND semantics for each compartment, as well as OR semantics for each non-compartmented role. To access the document if `role1` and `role2` are in different compartments, a user must possess both `role1` and `role2` to access the document, as well as a non-compartmented role that has a corresponding permission on the document.

If any permission on a document has a compartment, then the user must have that compartment in order to access any of the capabilities, even if the capability is not the one with the compartment.

Access to a document requires a permission in each compartment for which there is a permission on the document, regardless of the capability of the permission. So if there is a `read` permission for a role `compartment1`, there must also be an `update` permission for some role in `compartment1` (but not necessarily the same role). If you try to add `read`, `insert`, or `execute` permissions that reference a compartmented role to a document for which there is no `update` permission with the corresponding compartment, the `XDMP-MUSTHAVEUPDATE` exception is thrown.

4.2 Configuring Compartment Security

You can only add a compartment for a new role. To add a compartment, use the Admin Interface > Security > Roles > Create and enter a name for the compartment in the compartment field when you define each role in the compartment.

You cannot modify an existing role to use a compartment. To add a compartment to a role, you must delete the role and re-create it with a compartment. If you do re-create a role, any permissions you have on documents reference the old role (because they use the role ID, not the role name). So if you want those document permissions to use the new role, you need to update those documents with new permissions that reference the new role.

4.3 Example—Compartment Security

This section describes a scenario that uses compartment security. The scenario is not meant to demonstrate *the* correct way to set up compartment security, as your situation is likely to be unique. However, it demonstrates how compartment security works and may give you ideas for how to implement your own security model.

Description: For a MarkLogic application used by a government department, documents are classified with a security classification that dictates who may access the document. The department also restricts access to some documents based on the citizenship of the user. Additionally, some documents can only be accessed by employees with certain job functions.

To set up the compartment security for this scenario, you create the necessary roles, users, and documents with the example permissions. You will need access to both MarkLogic Admin Interface and Query Console.

To run through the example, perform the steps in each of the following sections:

- [Create Roles](#)
- [Create Users](#)
- [Create the Documents and Add Permissions](#)
- [Test It Out](#)

4.3.1 Create Roles

Using the Admin Interface > Security > Roles > Create, create the roles and compartments as follows:

1. Create roles named `US` and `Canada` and assign each of these roles the `country` compartment name. These roles form the `country` compartment.
2. Create roles named `Executive` and `Employee` and assign each of these roles the `job-function` compartment name. These roles form the `job-function` compartment.
3. Create roles named `top-secret` and `unclassified` and assign each of these roles the `classification` compartment name. These roles form the `classification` compartment.
4. Create a role named `can-read` with no compartment.

4.3.2 Create Users

Using the Admin Interface > Security > Users > Create, create users and give them the roles indicated in the following table.

User	Roles
Don	Executive, US, top-secret, can-read
Ellen	Employee, US, unclassified, can-read
Frank	Executive, Canada, top-secret, can-read
Gary	can-read
Hannah	unclassified, can-read

4.3.3 Create the Documents and Add Permissions

Using the MarkLogic Query Console, add a document for each combination of permissions in the following table:

Document	Permissions [Role and Capability]	Users with Access
doc1.xml	(Executive, read) (Executive, update) (US, read) (US, update) (top-secret, read) (top-secret, update) (can-read, read) (can-read, update)	Don
doc2.xml	(US, read) (US, update) (can-read, read) (can-read, update)	Don and Ellen
doc3.xml	(can-read, read) (can-read, update)	All users
doc4.xml	(Canada, read) (US, read) (US, update) (can-read, read) (can-read, update)	Frank, Don, Ellen
doc5.xml	(unclassified, read) (unclassified, update) (can-read, read) (can-read, update)	Ellen, Hannah

1. You can use XQuery code similar to the following example to insert the sample documents into a database of your choice. This code adds a document with a URI of `doc1.xml`, containing one `<a>` element and a set of five permissions.

```
xquery version "1.0-m1";
declare namespace html = "http://www.w3.org/1999/xhtml";
xdmp:document-insert (
  "/doc10.xml", <a>This is document 10.</a>,
  (xdmp:permission("can-read", "read"),
   xdmp:permission("can-read", "update"),
   xdmp:permission("US", "read"),
   xdmp:permission("US", "update"),
   xdmp:permission("Executive", "read"),
   xdmp:permission("Executive", "update"),
   xdmp:permission("top-secret", "read"),
   xdmp:permission("top-secret", "update")))

```

The `doc1.xml` document can only be read by `Don` because the permissions designate all three compartments and `Don` is the only user with a role in all three of the necessary compartmented roles `Executive`, `US`, and `top-secret`, plus the basic `can-read` role.

2. Create the rest of the sample documents changing the sample code as needed. You need to change the document URI and the text to correspond to `doc2.xml`, `doc3.xml`, `doc4.xml`, and `doc5.xml` and modify the permissions for each document as suggested in the table in “Create the Documents and Add Permissions” on page 26.

4.3.4 Test It Out

Using Query Console, you can execute a series of queries to verify that the users can access each document as specified in the table in “Create the Documents and Add Permissions” on page 26.

For simplicity, this sample query uses `xdmp:eval` and `xdmp:user` to execute a query in the context of each different user. Modify the document URI and the user name to verify the permissions until you understand how the compartment security logic works. If you added the roles, users, and documents as described in this scenario, the query results should match the table in “Create the Documents and Add Permissions” on page 26.

```
xquery version "1.0-m1";
declare namespace html = "http://www.w3.org/1999/xhtml";

xdmp:eval('fn:doc("/doc1.xml")', (),
  <options xmlns="xdmp:eval">
    <user-id>{xdmp:user("Don")}</user-id>
  </options>)

```

5.0 Protecting XQuery Functions With Privileges

Execute privileges provide authorization control for executing XQuery functions. MarkLogic provides three ways to protect XQuery functions:

- Built-in execute privileges, created by MarkLogic, control access to protected functions such as `xcmp:document-load`.
- Custom execute privileges, which you create using the Admin Interface or the security function in the `security.xqy` module, control access to functions you write.
- *Amps* temporarily amplify a user's authority by granting the authority to execute a single, specific function. You can only amp a function in a library module that is stored in the MarkLogic modules database.

This chapter describes the following:

- [Built-In MarkLogic Execute Privileges](#)
- [Protecting Your XQuery Code with Execute Privileges](#)
- [Temporarily Increasing Privileges with Amps](#)

5.1 Built-In MarkLogic Execute Privileges

Every installation of MarkLogic Server includes a set of pre-defined execute privileges. You can view this list either in the Admin Interface or in [Appendix B: Pre-defined Execute Privileges](#) of the *Administrator's Guide*.

5.2 Protecting Your XQuery Code with Execute Privileges

To protect the execution of an individual XQuery function that you have written, you can use an *execute privilege*. When a function is protected with an execute privilege, a user must have that specific privilege to run the protected XQuery function.

Note: Execute privileges operate at the function level. To protect an entire XQuery document that is stored in a modules database, you can use execute permissions. For details, see “Document Permissions” on page 17.

This section describes the following:

- [Using Execute Privileges](#)
- [Execute Privileges and App Servers](#)
- [Creating and Updating Collections](#)

5.2.1 Using Execute Privileges

The basic steps for using execute privileges are:

- Create the privilege.
- Assign the privilege to a role.
- Write code to test for the privilege.

You create privileges and assign them to roles using the Admin Interface. You use the `xdmp:security-assert` built-in function in your XQuery code to test for a privilege. This function tests to determine if the user running the code has the specified privilege. If the user possesses the privilege, then the code continues to execute. If the user does not possess the privilege, then the server throws an exception, which the application can catch and handle.

For example, to create an execute privilege to control the access to an XQuery function called `display-salary`, use the following steps:

1. Use the Admin Interface to create an execute privilege named `allow-display-salary`.
2. Assign any URI (for example, `http://my/privs/allow-display-salary`) to the execute privilege.
3. Assign a role to the privilege. You may want to create a specific role for this privilege depending on your security requirements.
4. Finally, in your `display-salary` XQuery function, include an `xdmp:security-assert` call to test for the `allow-display-salary` execute privilege as follows:

```
xquery version "1.0-ml";
declare function display-salary (
    $employee-id as xs:unsignedLong)
as xs:decimal
{
    xdmp:security-assert("allow-display-salary", "execute"),
    ...
} ;
```

5.2.2 Execute Privileges and App Servers

You can also control access to specific HTTP, WebDAV, ODBC, or XDBC servers using an execute privilege. Using the Admin Interface, you can specify that a privilege is required for server access. Any users that access the server must then possess the specified privilege. If a user tries to access an application on the server and does not possess the specified privilege, an exception is thrown. For an example of using this technique to control server access, see “Example: Using the Security Database in Different Servers” on page 36.

5.2.3 Creating and Updating Collections

To create or update a document and add it to a collection, the `unprotected-collections` privilege is required. You also need a role corresponding to an insert or update permission on the document. For a *protected collection* (a protected collection is created using the Admin Interface), you either need permissions to update that collection or the `any-collection execute` privilege. If the collection is an unprotected collection, then you need the `unprotected-collections execute` privilege. For details on adding collections while creating a document, see the documentation for `xdmp:document-load`, `xdmp:document-insert`, and `xdmp:document-add-collections` in the *MarkLogic XQuery and XSLT Function Reference*.

5.3 Temporarily Increasing Privileges with Amps

Amps provide users with additional authorization to execute a specific function. Assigning the user this authorization permanently could compromise the security of the system. When executing an *amped* function, the user is part of an *amped* role, which temporarily grants the user additional privileges and permissions of that role. Amps enable you to limit the effect of the additional roles (privileges and permissions) to a specific function.

For example, a user may need a count of all the documents in the database in order to create a report. If the user does not have read permissions on all the documents in the database, queries run by the user do not “see” all the documents in the database. If you want anyone to be able to know how many documents are in the database, regardless of whether they have permissions to see those documents, you can create a function named `document-count()` and use an amp on the function to elevate the user to a role with read permission for all documents. When the user executes the *amped* function, she temporarily has the necessary read permissions that enable the function to complete accurately. The administrator has in effect decided that, in the context of that `document-count()` function, it is safe to let anyone execute it.

Amps are security objects and you use the Admin Interface to create them. Amps are specific to a single function in a library module, which you specify by URI and local name when creating the amp. You can only amp a function that resides in a library module stored in the `Modules` directory (`<install_dir>/Modules`) or in the modules database configured for the server in which the function is executed. The recommended best practice is to put your code into the modules database. You can not amp functions in XQuery modules stored in other locations. For example, you cannot amp a function in a module installed under the filesystem root of an HTTP server, and you cannot amp functions that reside in a main module. Functions must reside in the `Modules` database or in the `Modules` directory because these locations are trusted. Allowing *amped* functions from under a server root or from functions submitted by a client could compromise security. For details on creating amps, see the “Security Administration” chapter of the *Administrator’s Guide*.

For an example that uses an amp, see “Access Control Based on Client IP Address” on page 47.

6.0 Authenticating Users

MarkLogic Server authenticates users when they access an application. This chapter describes users and the available authentication schemes, and includes the following sections:

- [Users](#)
- [Types of Authentication](#)

6.1 Users

A *user* in MarkLogic Server is the basis for authenticating requests to a MarkLogic application server. Users are assigned to roles. Roles carry security attributes, such as privileges and default permissions. Permissions assigned to documents pair a role with a capability, therefore roles are central to document permissions. Users derive authorization to perform actions from their roles.

You configure users in the Admin Interface, where you assign a user a name, a password, a set of roles, and a set of default permissions. To see the security attributes associated with a given user, click on the `User:username` link in the Admin Interface screen for the given user. For details on configuring users in the Admin Interface, see the “Security Administration” chapter in the *Administrator’s Guide*.

During the initial installation of MarkLogic Server, two users are created. One of the users is an authorized administrator who has the `admin` role. During the installation, you are prompted to specify the username and password for this user. The other user is a user named `nobody`, which is created with no roles assigned and is given a password which is randomly generated. For details about installing MarkLogic Server, see the *Installation Guide*.

6.2 Types of Authentication

You can control the authentication scheme for HTTP, WebDAV, ODBC, and XDBC App Servers. This section describes the authentication schemes and includes the following parts:

- [Basic](#)
- [Digest](#)
- [Digest-Basic](#)
- [Limitations of Digest and Basic Authentication](#)
- [Application Level](#)

6.2.1 Basic

Basic authentication is the typical authentication scheme for web applications. When a user accesses an application page, she is prompted for a username and password. In basic mode, the password is obfuscated but not encrypted.

6.2.2 Digest

Digest authentication works the same way as basic, but offers encryption of passwords sent over the network. When a user accesses an application page, she is prompted for a username and password.

Note: If you change an App Server from basic to digest authentication, it invalidates all passwords in the security database. You must then reenter the passwords in the Admin Interface. Alternatively, you can migrate to digest-basic mode initially, then switch to digest-only mode once all users have accessed the server at least once. The first time the user accesses the server after changing from basic to digest-basic scheme, the server computes the digest password by extracting the relevant information from the credentials supplied in basic mode.

6.2.3 Digest-Basic

The digest-basic authentication scheme uses the more secure digest scheme whenever possible, but reverts to basic authentication when needed. Some older browsers, for example, do not support digest authentication. The digest-basic scheme is also useful if you previously used basic authentication, but want to migrate to digest. The first time a user accesses the server after changing from basic to digest-basic authentication scheme, the server computes the digest password by extracting the relevant information from the credentials supplied in basic mode.

6.2.4 Limitations of Digest and Basic Authentication

Since the browser does not provide a way to clear a user's authentication information in basic or digest mode, the user remains logged in until the browser is shut down. In addition, there is no way to create a custom login page using these schemes. For certain deployments, application-level authentication may be more appropriate.

6.2.5 Application Level

Application-level authentication bypasses all authentication and automatically logs all users in as a specified *default user*. You specify the default user in the Admin Interface, and any users accessing the server automatically inherit the security attributes (roles, privileges, default permissions) of the default user. Application-level authentication is available on HTTP, ODBC, and WebDAV servers.

The default user should have the required privileges to at least read the initial page of the application. In many application scenarios, the user is then given the opportunity to explicitly log in to the rest of the application from that page. How much of the application and what data a user can access before explicitly logging in depends on the application and the roles that the default user is part of. For an example of this type of configuration, see “Using Custom Login Pages” on page 46.

7.0 Administering Security

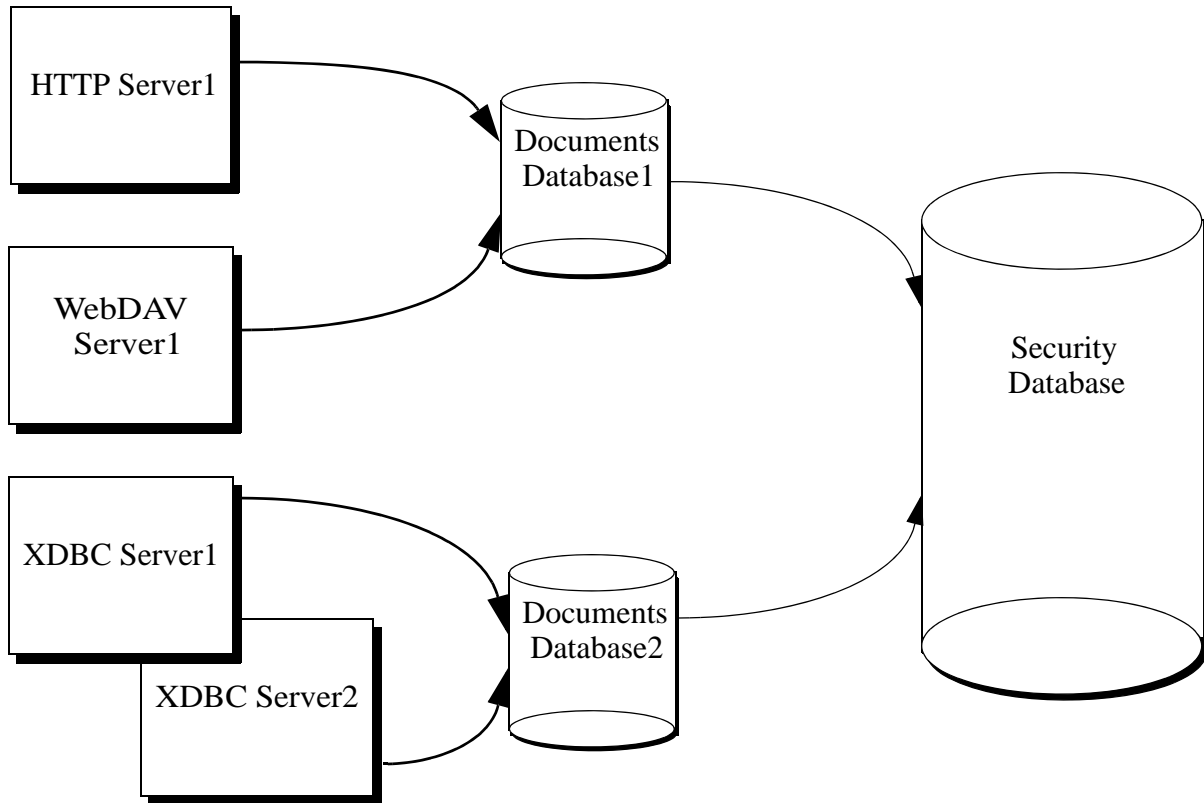
This chapter describes the basic steps to administer security in MarkLogic Server. It does not provide the detailed procedures for creating users, roles, privileges, and so on. For those procedures, see the “Security Administration” chapter of the *Administrator’s Guide*. This chapter includes the following sections:

- [Overview of the Security Database](#)
- [Associating a Security Database With a Documents Database](#)
- [Managing and Using Objects in the Security Database](#)
- [Backing Up the Security Database](#)
- [Example: Using the Security Database in Different Servers](#)

7.1 Overview of the Security Database

Authentication in MarkLogic Server occurs via the *security database*. The security database contains security objects such as privileges, roles, and users. A security database is associated with each HTTP, WebDAV, ODBC, or XDBC server. Typically, a single security database services all of the servers configured in a system. Actions against the server are authorized based on the security database. The security database works the same way for Enterprise Edition clustered systems as it does for Standard Edition single-node systems; there is always a single security database associated with each HTTP, WebDAV, ODBC, or XDBC server.

The configuration that associates the security database with the database and servers is at the database level. HTTP, WebDAV, ODBC, and XDBC servers each access a single documents database, and each database in turn accesses a single security database. Multiple documents databases can access the same security database. The following figure shows many servers accessing some shared and some different documents databases, but all accessing the same security database.

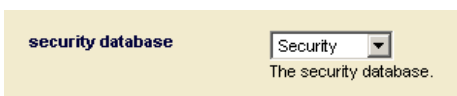


Sharing the security database across multiple servers provides a common security configuration. You can set up different privileges for different databases if that makes sense, but they are all stored in a common security database. For an example of this type of configuration, see “Example: Using the Security Database in Different Servers” on page 36.

In addition to storing users, roles, and privileges that you create, the security database also stores pre-defined privileges and pre-defined roles. These objects control access to privileged activities in MarkLogic Server. Examples of privileged activities include loading data and accessing URIs. The security database is initialized during the installation process. For a list of all of the pre-defined privileges and roles, see the corresponding appendixes in the *Administrator’s Guide*.

7.2 Associating a Security Database With a Documents Database

When you configure a database, you must specify which database is its security database. You can associate the security database to another database in the database configuration screen of the Admin Interface. This configuration specifies which database the server will use to authenticate users and authorize requests. By default, the security database is named *Security*. The following screen shot shows the server configuration screen drop-list that specifies the security database.



7.3 Managing and Using Objects in the Security Database

There are two mechanisms available to add, change, delete, and use objects in the security database: the Admin Interface and the XQuery functions, provided by the `security.xqy` library module. This section describes what you can do with each of these mechanisms and includes the following topics:

- [Using the Admin Interface](#)
- [Using the security.xqy Module Functions](#)

7.3.1 Using the Admin Interface

The Admin Interface is an application installed with MarkLogic Server for administering databases, servers, clusters, and security objects. The Admin Interface is designed to manage the objects in the security database, although it manages other things, such as configuration information, too. You use the Admin Interface to create, change, or delete objects in the security database. Activities such as creating users, creating roles, assigning privileges to roles, and so on, are all done in the Admin Interface. By default, the Admin Interface application runs on port 8001.

For the procedures for creating, deleting, and modifying security objects, see the *Administrator's Guide*.

7.3.2 Using the security.xqy Module Functions

The installation process installs an XQuery library to help you use security objects in your XQuery code. The `security.xqy` library module includes functions to access user and privilege information, as well as functions to create, modify, and delete objects in the security database.

The functions in `security.xqy` must be executed against the security database. You can use these functions to do a wide variety of things. For example, you can write code to test which collections a user has access to, and use that information in your code.

For the signatures and descriptions of the functions in `security.xqy`, see the *MarkLogic XQuery and XSLT Function Reference*.

7.4 Backing Up the Security Database

The security database is the central entry point to all of your MarkLogic Server applications. If the security database becomes unavailable, no users can access any applications. Therefore, it is important to create a backup of the security database. Use the database backup utility in the Admin Interface to back up the security database. For details, see the “Backing Up and Restoring a Database” chapter of the *Administrator's Guide*.

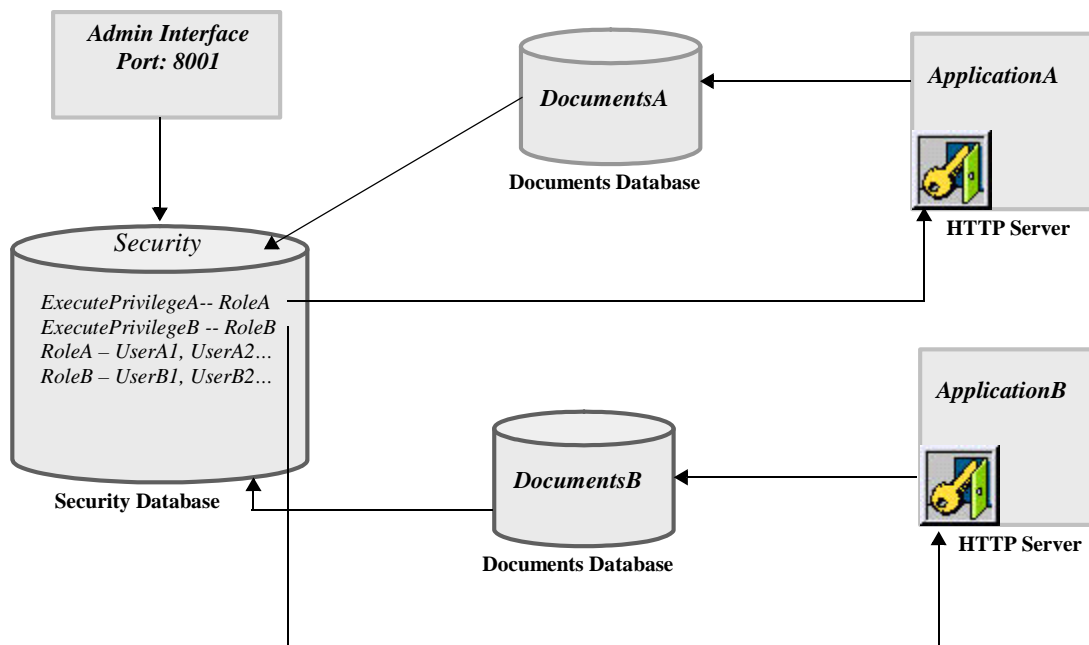
7.5 Example: Using the Security Database in Different Servers

The security database typically is used for the entire system, including all of the HTTP, WebDAV, ODBC, and XDBC servers configured. You can create distinct privileges to control access to each server. If each server accesses a different document database, these privileges can effectively control access to each database (because the database is associated with the server). Users must have the appropriate *login privileges* to log into the server, and therefore they have no way of accessing either the applications or the content stored in the database accessed through that server without possessing the appropriate privilege. This example describes such a scenario.

Consider an example with two databases—`DocumentsA` and `DocumentsB`. `DocumentsA` and `DocumentsB` share a single security database, `Security`. `Security` is the default security database managed by the Admin Interface on port 8001. There are two HTTP servers, `ApplicationA` and `ApplicationB`, connected to `DocumentsA` and `DocumentsB` respectively.

`ExecutePrivilegeA` controls login access to `ApplicationA`, and `ExecutePrivilegeB` to `ApplicationB`. `RoleA` is granted `ExecutePrivilegeA` and `RoleB` is granted `ExecutePrivilegeB`.

With this configuration, users who are assigned `RoleA` can access documents in `DocumentsA` and users of `RoleB` can access documents in `DocumentsB`. Assuming that `ExecutePrivilegeA` or `ExecutePrivilegeB` are appropriately configured as login privileges on every HTTP and XDBC server that accesses either `DocumentsA` or `DocumentsB`, user access to these databases can conveniently be managed by assigning users the role(s) `RoleA` and/or `RoleB` as required.



Note: The Admin Interface at port 8001 is also used to configure all databases, HTTP servers, hosts, and so on. The connection between the Admin Interface and the `Security` database in the diagram simply indicates that the Admin Interface is storing all security objects—users, roles, and privileges—in `Security` database.

The steps below outline the process to create the configuration in the above example.

1. Create two document databases: `DocumentsA` and `DocumentsB`. Leave the security database for the document databases as `Security` (the default setting).
2. Create two execute privileges: `ExecutePrivilegeA` and `ExecutePrivilegeB`. They represent the privilege to access `ApplicationA` and `ApplicationB` respectively. `ApplicationA` and `ApplicationB` are two HTTP servers that are created later in this procedure.

Note: The new execute privileges created using the Admin Interface are stored in the `Security` database. The new roles and users created below are also stored in the `Security` database.

3. Create two new roles. These roles are used to organize users into groups and to facilitate granting access to users as a group.
 - a. Create a new role. Name it `RoleA`.
 - b. Scroll down to the Execute Privileges section and select `ExecutePrivilegeA`. This associates `ExecutePrivilegeA` with `RoleA`. Any user assigned `RoleA` is granted `ExecutePrivilegeA`.
 - c. Repeat the steps for `RoleB`, selecting `ExecutePrivilegeB` instead.
4. Create two new HTTP servers:
 - a. Create a new HTTP server. Name it `ApplicationA`.
 - b. Select `DocumentsA` as the database. `ApplicationA` is now attached to `DocumentsA` which in turn uses `Security` as its security database.
 - c. Select basic, digest or digest-basic authentication scheme.
 - d. Select `ExecutePrivilegeA` in the privilege drop down menu. This indicates that `ExecutePrivilegeA` is required to access `ApplicationA`.
 - e. Repeat the steps for `ApplicationB`, selecting `ExecutePrivilegeB` instead.
5. Create new users.
 - a. Create a new user named `UserA1`.

- b. Scroll down to the Roles section and select `RoleA`.
- c. Repeat the steps for `UserB1`, selecting `RoleB` in the roles section.

`UserA1` is granted `ExecutePrivilegeA` by virtue of its role (`RoleA`) and has login access to `ApplicationA`. Because `ApplicationA` is connected to `DocumentsA`, `UserA1` is able to access documents in `DocumentsA` assuming no additional security requirements are implemented in `ApplicationA`, or added to documents in `DocumentsA`. The corresponding is true for `UserB1`.

The configuration process is now complete. Additional users can be created by simply repeating step 5 and selecting the appropriate role. All users assigned `RoleA` have login access to `ApplicationA` and all users assigned `RoleB` have login access to `ApplicationB`.

This approach can also be easily extended to handle additional discrete databases and user groups by creating additional document databases, roles and execute privileges as necessary.

8.0 Auditing

Auditing is the monitoring and recording of selected operational actions from both application users and administrative users. You can audit various kinds of actions related to document access and updates, configuration changes, administrative actions, code execution, and changes to access control. You can audit both successful and failed activities. This chapter contains the following parts:

- [Why Is Auditing Used?](#)
- [MarkLogic Auditing](#)
- [Configuring Auditing](#)
- [Best Practices](#)

For procedures on setting up auditing as well as a list of audit events, see [Auditing Events](#) in the *Administrator's Guide*.

8.1 Why Is Auditing Used?

You typically use auditing to perform the following activities:

- Enable accountability for actions. These might include actions taken on documents, changes to configuration settings, administrative actions, changes to the security database, or system-wide events.
- Deter users or potential intruders from inappropriate actions.
- Investigate suspicious activity.
- Notify an auditor of the actions of an unauthorized user.
- Detect problems with an authorization or access control implementation. For example, you can design audit policies that you expect to never generate an audit record because the data is protected in other ways. However, if these policies generate audit records, then you know the other security controls are not properly implemented.
- Address auditing requirements for regulatory compliance.

8.2 MarkLogic Auditing

MarkLogic Server includes an auditing capability. You can enable auditing to capture security-relevant events to monitor suspicious database activity or to satisfy applicable auditing requirements. You can configure the generation of audit events by including or excluding MarkLogic Server roles, users, or documents based on URI. Some actions that can be audited are the following:

- startup and shutdown of MarkLogic Server
- adding or removing roles from a user
- usage of amps
- starting and stopping the auditing system

For the complete list of auditable events and their descriptions, see [Auditing Events](#) in the *Administrator's Guide*.

8.3 Configuring Auditing

Auditing is configured at the MarkLogic Server cluster management group level. A MarkLogic Server group is a set of similarly configured hosts in a cluster, and includes configurations for the HTTP, WebDAV, ODBC, and XDBC App Servers in the group. The group auditing configuration includes enabling and disabling auditing for each cluster management group.

Audit records are stored on the local file system of the host on which the event is detected and on which the Server subsystem is running.

Rotation of the audit logs to different files is configurable by various intervals, and the number of audit files to keep is also configurable.

For more details and examples of audit event logs, see [Auditing Events](#) in the *Administrator's Guide*.

8.4 Best Practices

Auditing can be an effective method of enforcing strong internal controls enabling your application to meet any applicable regulatory compliance requirements. Appropriate auditing can help you to monitor business operations and detect activities that may deviate from company policy. If it is important to your security policy to monitor this type of activity, then you should consider enabling and configuring auditing on your system.

Be selective with auditing and ensure that it meets your business needs. As a general rule, design your auditing strategy to collect the amount and type of information that you need to meet your requirements, while ensuring a focus on events that cause the greatest security concerns.

If you enable auditing, develop a monitoring mechanism to use the audit event logs. Such a system might periodically archive and purge the audit event logs.

9.0 Designing Security Policies

This chapter describes the general steps to follow when using security in an application. Because of the flexibility of the MarkLogic Server security model, there are different ways to implement similar security policies. These steps are simple guidelines; the actual steps you take depends on the security policies you need to implement. The following sections are included:

- [Research Your Security Requirements](#)
- [Plan Roles and Privileges](#)

9.1 Research Your Security Requirements

As a first step in planning your security policies, try to have answers for the following types of questions:

- What documents do you want to protect?
- What code do you want to control the execution of?
- Are there any natural categories you can define based on business function (for example, marketing, sales, engineering)?
- What is the level of risk posed by your users? Are your applications used only by trusted, internal people or are they open to a wider audience?
- How sensitive is the data you are protecting?

This list is not necessarily comprehensive, but is a good way to start thinking about your security policy.

9.2 Plan Roles and Privileges

Depending on your security requirements and the structure of your enterprise or organization, plan the roles and privileges that make the most sense.

1. Determine the level of granularity with which you need to protect objects in the database.
2. Determine how you want to group privileges together in roles.
3. Create needed URI and execute privileges.
4. Create roles.
5. Create users.
6. Assign users to roles.
7. Set default permissions for users, either indirectly through roles or directly through the users.

8. Protect code with `xdmp:security-assert` functions, where needed.
9. Load your documents with the appropriate permissions. If needed, change the permissions of existing documents using the `xdmp:document-add-permissions`, `xdmp:document-set-permissions`, and `xdmp:document-remove-permissions` functions.
10. Assign access privileges to HTTP, WebDAV, ODBC, and XDBC servers as needed.

10.0 Sample Security Scenarios

This chapter describes some common scenarios for defining security policies in your applications. The scenarios shown here are by no means exhaustive. There are many possibilities for how to set up security in your applications. The following sections are included:

- [Protecting the Execution of XQuery Modules](#)
- [Choosing the Access Control for an Application](#)
- [Implementing Security for a Read-Only User](#)

10.1 Protecting the Execution of XQuery Modules

One simple way to restrict access to your MarkLogic Server application is to limit the users that have permission to run the application. If you load your Xquery code into a modules database, you can use an execute permission on the XQuery document itself to control who can run it. Then, a user must possess `execute` permissions to run the module. To set up a module to do this, perform the following steps:

1. Using the Admin Interface, specify a modules database in the configuration for the App Server (HTTP or WebDAV) that controls the execution of your XQuery module.
2. Load the XQuery module into the modules database, using a URI with an `.xqy` extension, for example `my_module.xqy`.
3. Set `execute` permissions on the XQuery document for a given role. For example, if you want users with the `run_application` role to be able to execute an XQuery module with the URI `http://modules/my_module.xqy`, run a query similar to the following:

```
xdmp:document-set-permissions("http://modules/my_module.xqy",  
                               xdmp:permission("run_application", "execute") )
```

4. Create the `run_application` role.
5. Assign the `run_application` role to the users who can run this application.

Now only users with the `run_application` role can execute this document.

Note: Because your application could also contain amped functions, this technique can help restrict access to applications that use amps.

10.2 Choosing the Access Control for an Application

The role-based security model in MarkLogic Server combined with the supported authentication schemes provides numerous options for implementing application access control. This section describes common application access control alternatives:

- [Open Access, No Log In](#)
- [Providing Uniform Access to All Authenticated Users](#)
- [Limiting Access to a Subset of Users](#)
- [Using Custom Login Pages](#)
- [Access Control Based on Client IP Address](#)

For details on the different authentication schemes, see “Types of Authentication” on page 31.

10.2.1 Open Access, No Log In

This approach may be appropriate if security is not a concern for your MarkLogic Server implementation or if you are just getting started and want to explore the capabilities of MarkLogic Server before contemplating your security architecture. This scenario provides all of your users with the `admin` role.

You can turn off access control for each HTTP or WebDAV server individually by following these steps using the Admin Interface:

1. Go to the Configure tab for the HTTP server for which you want to turn off access control.
2. Scroll down to the authentication field and choose `application-level` for the authentication scheme.
3. Choose a user with the `admin` role for the default user. For example, you may choose the `admin` user you created when you installed MarkLogic.

Note: To assist with identifying users with the `admin` role, the default user selection field places `(admin)` next to `admin` users.

In this scenario, all users accessing the application server are automatically logged in with a user that has the `admin` role. By default, the `admin` role has the privileges and permissions to perform any action and access any document in the server. Therefore, security is essentially turned off for the application. All users have full access to the application and database associated with the application server.

10.2.2 Providing Uniform Access to All Authenticated Users

This approach allows you to restrict application access to users in your security database, and gives those users full access to all application servers defined in MarkLogic Server. There are multiple ways to achieve the same objective but this is the simplest way.

1. In the Admin Interface, go to the Users tab under Security.
2. Give all users in the security database the `admin` role.
3. Go to the Configuration tab for all HTTP and WebDAV servers in the system.
4. Go to the authentication field and choose `digest`, `basic` or `digest-basic` authentication.
5. Leave the privilege field blank since it has no effect in this scenario. This field specifies the privilege that is needed to log into application server. However, the users are assigned the `admin` role and are treated as having all privileges.

In this scenario, all users must authenticate with a username and password. Once they are authenticated, however, they have full access to all functions and data in the server.

10.2.3 Limiting Access to a Subset of Users

This application access control method can be modified or extended to meet the requirements in many application scenarios. It uses more of the available security features and therefore requires a better understanding of the security model.

To limit application access to a subset of the users in the security database, perform the following steps using the Admin Interface:

1. Create an execute privilege named `exe-priv-app1` to represent the privilege to access the App Server.
2. Create a role named `role-app1` that has `exe-priv-app1` execute privilege.
3. Add `role-app1` to the roles of all users in the security database who should have access to this App Server.
4. In the Configuration page for this App Server, scroll down to the authentication field and select `digest`, `basic` or `digest-basic`. If you want to use application-level authentication to achieve the same objective, a custom login page is required. See the next section for details.
5. Select `exe-priv-app1` for the privilege field. Once this is done, only the users who have the `exe-priv-app1` by virtue of their role(s) are able to access this App Server.

Note: If you want any user in the security database to be able to access the application, leave the privilege field blank.

At this point, the application access control is configured.

This method of authentication also needs to be accompanied by the appropriate security configuration for both users and documents associated with this App Server. For example, functions such as `xdmp:document-insert` and `xdmp:document-load` throw exceptions unless the user possesses the appropriate execute privileges. Also, users must have the appropriate default permissions (or specify the appropriate permissions with the API) when creating new documents in a database. Documents created by a user who does not have the `admin` role must be created with at least one update permission or else the transaction throws an `XDMP-MUSTHAVEUPDATE` exception. The update permission is required because otherwise once the documents are created no user (except users with the `admin` role) would be able to access them, including the user who created them.

10.2.4 Using Custom Login Pages

Digest and basic authentication use the browser's username and password prompt to obtain user credentials. The server then authenticates the credentials against the security database. There is no good way to create a custom login page using digest and basic authentication. To create custom login pages, you need to use application-level authentication.

To configure MarkLogic Server to use a custom login page for an App Server, perform the following steps using the Admin Interface:

1. Go to the Configuration tab for the HTTP App Server for which you want to create a custom login page.
2. Scroll down to the authentication field and select application-level.
3. Choose `nobody` as the default user. The `nobody` user is automatically created when MarkLogic Server is installed. It does not have an associated role and therefore has no privileges. The `nobody` user can only access pages and perform functions for which no privileges are required.
4. Create a custom login page that meets your needs. We refer to this page as `login.xqy`.
5. Make `login.xqy` the default page displayed by the application server. Do not require any privilege to access `login.xqy` (that is, do not place `xdmp:security-assert()` in the beginning of the code for `login.xqy`. This makes `login.xqy` accessible by `nobody`, the default user specified above, until the actual user logs in with his credentials.

The `login.xqy` page likely contains a snippet of code as shown below:

```
...return
if xdmp:login($username, $password) then
  ... protected page goes here...
else
  ... redirect to login page or display error page...
```

The rest of this example assumes that all valid users can access all the pages and functions within the application.

Note: If you are using a modules database to store your code, the `login.xqy` file still needs to have an `execute` permission that allows the `nobody` (or whichever is the default) user to access the module. For example, you can put an `execute` permission paired with the `app-user` role on the `login.xqy` module document, and make sure the `nobody` user has the `app-user` role (which it does by default).

6. Create a role called `application-user-role`.
7. Create an execute privilege called `application-privilege`. Add this privilege to the `application-user-role`.
8. Add the `application-user-role` to all users who are allowed to access the application.
9. Add this snippet of code before the code that displays each of the pages in the application, except for `login.xqy`:

```
try
{
  xdmp:security-assert ("application-privilege", "execute")
}
catch ($e)
{
  xdmp:redirect-response ("login.xqy")
}
```

or

```
if (not (xdmp:has-privilege ("application-privilege", "execute")))
then
(
  xdmp:redirect-response ("login.xqy")
)
else ()
```

This ensures that only a user who has the `application-privilege` by virtue of his role can access these protected pages.

Similar to the previous approach, this method of authentication requires the appropriate security configuration for users and documents. See “Introduction to Security” on page 5 for background on the security model.

10.2.5 Access Control Based on Client IP Address

MarkLogic Server supports deployments in which a user is automatically given access to the application based on the client IP address.

Consider a scenario in which a user is automatically logged in if he is accessing the application locally (as `local-user`) or from an approved subnet (as `site-user`). Otherwise, the user is asked to login explicitly. The steps below describe how to configure MarkLogic Server to achieve this access control.

1. Using the Admin Interface, configure the App Server to use a custom login page:
 - a. Go to the Configuration tab for the HTTP or WebDAV App Server for which you want to create a custom login page.
 - b. Scroll down to the authentication field and select application-level.
 - c. For this example, choose `nobody` as the default user. The `nobody` user is automatically created when MarkLogic Server is installed. It does not have an associated role and hence has no privileges. The `nobody` user can only access pages and perform functions for which no privileges are required.
2. Add the following code snippet to the beginning of the default page displayed by the application, for example, `default.xqy`.

```
xquery version "1.0-ml"

declare namespace widget = "http://widget.com"
import module "http://widget.com" at "/login-routine.xqy"

let $login := widget:try-ip-login()
return
if($login) then
  <html>
    <body>
      The protected page goes here.
      You are {xdmp:get-current-user()}
    </body>
  </html>
else
  xdmp:redirect-response("login.xqy")
```

The `try-ip-login` function is defined in `login-routine.xqy`. It is used to determine if the user can be automatically logged in based on the client IP address. If the user cannot be logged in automatically, he is redirected to a login page called `login.xqy` where he has to log in explicitly. See “Using Custom Login Pages” on page 46 for example code for `login.xqy`.

3. Define `try-ip-login`:
 - a. Create a file named `login-routine.xqy` and place the file in the `Modules` directory within the MarkLogic Server program directory. You create an `amp` for `try-ip-login` in `login-routine.xqy` in the next code sample. For security reasons, all `amped`

functions must be located in the specified `Modules` directory or in the `Modules` database for the App Server.

- b. Add the following code to `login-routine.xqy`:

```
xquery version "1.0-ml"

module "http://widget.com"
declare namespace widget = "http://widget.com"

define function try-ip-login() as xs:boolean
{
  let $ip := xdmp:get-request-client-address()
  return
  if(compare($ip,"127.0.0.1") eq 0) then (:local host:)
    xdmp:login("localuser", ())
  else if(starts-with($ip,<approved-subnet>)) then
    xdmp:login("site-user", ())
  else
    false()
}
```

If the user is accessing the application from an approved IP address, `try-ip-login` logs in the user with username `local-user` or `site-user` as appropriate and returns `true`. Otherwise, `try-ip-login` returns `false`.

Note: In the code snippet above, the empty sequence `()` is supplied in place of the actual passwords for `local-user` and `site-user`. The pre-defined `xdmp-login` execute privilege grants the right to call `xdmp:login` without the actual password. This makes it possible to create deployments in which users can be automatically logged in without storing user passwords outside the system.

4. Finally, to ensure that the code snippet above is called with the requisite `xdmp-login` privilege, configure an amp for `try-ip-login`:
 - a. Using the Admin Interface, create a role called `login-role`.
 - b. Assign the pre-defined `xdmp-login` execute privilege to `login-role`. The `xdmp-login` privilege gives a user of the `login-role` the right to call `xdmp:login` for any user without supplying the password.
 - c. Create an amp for `try-ip-login` as shown below:

New Amp ok cancel

amp -- A role amplification.

local name
A function local-name.
Required. You must supply a value for local-name.

namespace
A namespace.
Required. You must supply a value for namespace.

document uri
A document's URI.
Required. You must supply a value for document-uri.

database
A database the module is found in.

roles -- The roles assigned.

admin

admin-builtins

domain-management

filesystem-access

login-role

An amp temporarily assigns additional role(s) to a user only for the execution of the specified function. The amp above gives any user who is executing `try-ip-login()` the `login-role` temporarily for the execution of the function.

In this example, `default.xqy` is executed as `nobody`, the default user for the application. When the `try-ip-login` function is called, the `nobody` user is temporarily amped to the `login-role`. The `nobody` user is temporarily assigned the `xdmp:login` execute privilege by virtue of the `login-role`. This enables `nobody` to call `xdmp:login` in `try-ip-login` for any user without the corresponding password. Once the login process is completed, the user can access the application with the permissions and privileges of `local-user` or `site-user` as appropriate.

5. The remainder of the example assumes that `local-user` and `site-user` can access all the pages and functions within the application.
 - a. Create a role called `application-user-role`.
 - b. Create an execute privilege called `application-privilege`. Add this privilege to the `application-user-role`.
 - c. Add the `application-user-role` to `local-user` and `site-user`.

- d. Add this snippet of code before the code that displays each of the subsequent pages in the application:

```
try
{
  xdm:security-assert("application-privilege", "execute")
  ...
}
catch($e)
{
  xdm:redirect-response("login.xqy")
}
```

or

```
if(not(xdm:has-privilege("application-privilege", "execute")))
then
(
  xdm:redirect-response("login.xqy")
)
else ()
```

This ensures that only the user who has the `application-privilege` by virtue of his role can access these protected pages.

10.3 Implementing Security for a Read-Only User

In this scenario, assume that you want to implement a security model that enables your users to run any XQuery code stored in the modules database for a specific App Server with read-only permissions on all documents in the database.

Reviewing the MarkLogic security model, recall that users do not have permissions, documents have permissions. And permissions are made up of a role paired with a capability. Additionally, execute privileges protect code execution and URI privileges protect the creation of documents in a specific URI namespace. This example shows one way to implement the read-only user and is divided into the following parts:

- [Steps For Example Setup](#)
- [Troubleshooting Tips](#)

10.3.1 Steps For Example Setup

To set up this example scenario, perform the following steps, using the Admin Interface:

1. Create a role named `ReadsStuff`.
2. Create a user named `ReadOnly` and grant this user the `ReadsStuff` role.
3. Create a role named `WritesStuff` and grant this role the `ReadsStuff` role.

4. Grant the `WritesStuff` role the `any-uri` privilege, as well as any execute privileges needed for your application code.
5. Create a user named `LoadsStuff` and grant this user the `WritesStuff` role. When you load documents, load them as the `LoadsStuff` user and give each document an update and insert permission for the `WritesStuff` role and a read permission for the `ReadsStuff` role.

Here is sample code to create a set of permissions to do this as an option to either the `xdmp:document-insert` function or the `xdmp:document-load` function:

```
(xdmp:permission("ReadsStuff", "read"),  
xdmp:permission("WritesStuff", "insert"),  
xdmp:permission("WritesStuff", "update"))
```

An alternative to specifying the permissions when you load documents is to assign default permissions to the `LoadsStuff` user or the `WritesStuff` role.

10.3.2 Troubleshooting Tips

If you are running a URL rewriter (or an error handler), you need to give the `ReadsStuff` role to the `nobody` user or whichever user is the default user for your App Server. When the URL rewriter executes, the request has not yet been authenticated, so it runs as the default user. The default user is `nobody` unless you have specified a different default for your App Server. The best practice is to create another role, for example `my-app-user` and add an execute permission for the URL rewriter and your error handler (if any) for the `my-app-user` role. This is better because you do not want the `nobody` user to have access to your database.

11.0 Securing Your Production Deployment

A security system is only as good as its weakest link. This chapter describes some general principles to think about with an eye toward hardening your entire environment for security, and contains the following sections:

- [Add Password Protections](#)
- [Adhere to the Principle of Least Privilege](#)
- [Infrastructure Hardening](#)
- [Implement Auditing](#)
- [Develop and Enforce Application Security](#)
- [Use MarkLogic Security Features](#)
- [Read About Security Issues](#)

11.1 Add Password Protections

When your data and business requirements warrant it, design and implement password protections. These protections can range from providing guidelines to your users to implementing programmatic checking to enforce password complexity and management.

Complexity verification verifies that each password is complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords. This encourages users to create strong passwords.

Password management includes things such as password aging and expiration, automatically locking users out of the application after failed login attempts, and controlling the reuse of old passwords.

To enforce password complexity programmatically, use the password plugins. For more information about the plugin framework and to view a sample password plugin, see [System Plugin Framework](#) and [Password Plugin Sample](#) in the *Application Developer's Guide*.]

11.2 Adhere to the Principle of Least Privilege

Grant necessary privileges only. Do not provide users or roles more privileges than are necessary. If possible, grant privileges to roles, not individual users. The principle of least privilege is that users are given only those privileges that are actually required to efficiently perform their jobs.

Restrict the following as much as possible:

- The number of users granted the `admin` or `security` roles.
- The number of roles or users who are allowed to make changes to security objects, such as roles, users, and document permissions.

- The number of roles that have capabilities to add, change or remove security-related privileges.

11.3 Infrastructure Hardening

Most computer platforms offer network security features to limit outside access to the system. The purpose of infrastructure hardening is to eliminate as many security risks as possible. It can involve both hardware and software, as well as physical restrictions. The following are some infrastructure hardening topics:

- [OS-Level Restrictions](#)
- [Network Security](#)
- [Port Management](#)
- [Physical Access](#)

11.3.1 OS-Level Restrictions

The United States National Security Agency develops and distributes security configuration guidance for a wide variety of software, including the most common operating system platforms. You can view this guidance on their website at:
http://www.nsa.gov/ia/mitigation_guidance/security_configuration_guides/operating_systems.shtml.

11.3.2 Network Security

Encrypt network traffic between the browser and MarkLogic Server by enabling SSL. You can also enable SSL for intra-cluster communication. For high security needs, make sure MarkLogic Server runs in FIPS mode (which is the default mode). This option restricts your SSL ciphers to those that have met the FIPS 140-2 Level 1 validation requirements. For information on how to configure SSL and FIPS mode, see [Clusters](#) in the *Administrator's Guide*.

11.3.3 Port Management

Protect access to MarkLogic's Admin Interface and development tool ports:8000, 8001, 8002 behind a corporate firewall. While your MarkLogic application may run on a publicly available port, such as port 80, it is good practice to secure the MarkLogic Admin Interface and other development application ports behind a firewall.

11.3.4 Physical Access

Ensure that machines running MarkLogic Server are in a physically secure location. Physical access to a server is a high security risk. Physical access to a server by an unauthorized user could result in unauthorized access or modification, as well as installation of hardware or software designed to circumvent security. To maintain a secure environment, you should restrict physical access to your MarkLogic Server host computers.

11.4 Implement Auditing

MarkLogic includes an auditing capability. Designing and implementing an auditing policy can be an important part of your overall security planning. For more details, see [Auditing](#) in this guide. For procedures related to enabling auditing, see [Auditing Events](#) in the *Administrator's Guide*.

11.5 Develop and Enforce Application Security

An important step in creating a MarkLogic application is to ensure that it is properly secure. Network security mostly ignores the contents of HTTP traffic, therefore you can't use network layer protection (firewall, SSL, IDS, hardening) to stop or detect application layer attacks. The Open Web Application Security Project is an open group focused on understanding and improving the security of web applications and web services. You can visit their site at: <http://www.owasp.org/>. The OWASP Top Ten Project is one starting point for understanding how you can build good security into your application.

11.6 Use MarkLogic Security Features

Let collections and document permissions restrict the data access for the user. Do not write your own access restriction code. Write code so that it uses the MarkLogic Server security model and operates on the correct data based on the user's permissions and the current documents in use.

11.7 Read About Security Issues

Many excellent resources exist on the Internet. These sources contain valuable security-related information for everyone in the enterprise software development and deployment chain from software developers and system administrators to managers. For example, the Defense Information Systems Agency (DISA) sponsors the Information Assurance Support Environment website found at <http://iase.disa.mil/index2.html>. This site contains Security Technical Implementation Guides (STIGs). The STIGs contain technical guidance to "lock down" information systems and software that might otherwise be vulnerable to a malicious computer attack.

Another example is the CERT Program, a part of the Software Engineering Institute, a federally funded research and development center operated by Carnegie Mellon University. This organization is devoted to ensuring that appropriate technology and systems management practices are used to resist attacks on networked systems and to limit damage and ensure continuity of critical services in spite of successful attacks, accidents, or failures. For more detailed information about CERT visit their website: <http://www.cert.org/>.

12.0 Technical Support

MarkLogic provides technical support according to the terms detailed in your Software License Agreement or End User License Agreement. For evaluation licenses, MarkLogic may provide support on an “as possible” basis.

For customers with a support contract, we invite you to visit our support website at <http://support.marklogic.com> to access information on known and fixed issues.

For complete product documentation, the latest product release downloads, and other useful information for developers, visit our developer site at <http://developer.marklogic.com>.

If you have questions or comments, you may contact MarkLogic Technical Support at the following email address:

support@marklogic.com

If reporting a query evaluation problem, please be sure to include the sample XQuery code.

Combined Product Notices

MarkLogic Server 6.0 and supporting products.

Last updated: August 19, 2013

COPYRIGHT

Copyright © 2013 MarkLogic Corporation. All rights reserved.

This technology is protected by U.S. Patent No. 7,127,469B2, U.S. Patent No. 7,171,404B2, U.S. Patent No. 7,756,858 B2, and U.S. Patent No 7,962,474 B2.

The MarkLogic software is protected by United States and international copyright laws, and incorporates certain third party libraries and components which are subject to the attributions, terms, conditions and disclaimers set forth below.

1. Contains SAP BusinessObjects Text Analysis XI from SAP AG. Copyright © 1996-2011. All rights reserved.
2. Highslide Software from Highslide Software Torstein Honsi. All Highslide Software is protected by local and international copyright laws. All rights reserved.
3. Icons developed by Yusuke Kamiyamane. Copyright © 2011 Yusuke Kamiyamane. All rights reserved. Icons are licensed subject to <http://creativecommons.org/licenses/by/3.0/legalcode>.
4. Antenna House OfficeHTML Copyright © 2000-2008 Antenna House, Inc. All rights reserved.
5. Argus Copyright ©1999-2008 Icenit Technology Ltd. All rights reserved.
6. Rosette Linguistics Platform 6.5.2/6.5.3 from Basis Technology Corporation, Copyright © 2004-2008 Basis Technology Corporation. All rights reserved.
7. ISYS Search. Copyright © 2011 ISYS™ Search Software, Inc. All rights reserved.
8. Software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>) Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved. Copyright © 1998-2011 The OpenSSL Project. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)" 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org. 5. Products derived from this

software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project. 6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)" THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This product includes cryptographic software written by Eric Young (ey@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). Additional terms may apply to the foregoing software as further set forth at <http://www.openssl.org/source/license.html>

9. ICU v. 4.2.1 available from <http://site.icu-project.org/> Copyright © 1995-2011 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

10. Tidy available at <http://www.w3.org/People/Raggett/tidy/>. Copyright © 1998-2008 World Wide Web Consortium (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved.

11. 'zlib' general purpose compression library v. 1.2.3. Copyright © 1995-2010 Jean-loup Gailly and Mark Adler. http://www.zlib.net/zlib_license.html

12. RSA Data Security, Inc. MD5 Message-Digest Algorithm, copyright © 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved. License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function. License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work. RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind. These notices must be retained in any copies of any part of this documentation and/or software.

13. TRE software, available at <http://laurikari.net/tre/>. Copyright © 2001-2009 Ville Laurikari <vl@iki.fi>. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: (1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

14. Eigen Library, version 2.0.10, available at http://eigen.tuxfamily.org/index.php?title=Main_Page. Eigen Library is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. Eigen Library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with Eigen Library. If not, see: <http://www.gnu.org/copyleft/lesser.html>

15. FunctX XQuery Library v.1.0 available at <http://www.xqueryfunctions.com/xq/>, Copyright © Datypic (Priscilla Walmsley). You can redistribute such library and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the

hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License as available at <http://www.gnu.org/licenses/lgpl-2.1.html#SEC4> for more details.

16. DeployJava.js v. 1.8, Copyright © 2006, 2011, Oracle and/or its affiliates. All rights reserved. ORACLE PROPRIETARY/CONFIDENTIAL. Use of the foregoing library is subject to the following license terms. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of Oracle nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

17. YUI libraries v.2.8 available at <http://developer.yahoo.com/yui/>. Copyright © 2011 Yahoo! Inc. All rights reserved. Use of the foregoing libraries is subject to the following license terms. Redistribution and use of this software in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of Yahoo! Inc. nor the names of YUI's contributors may be used to endorse or promote products derived from this software without specific prior written permission of Yahoo! Inc. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING

NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. Additional notices related to components included in the foregoing library are also available at the following URL: <http://yuilibrary.com/license/>

18. XSLTForms libraries available at <http://sourceforge.net/projects/xsltforms/>. You can redistribute such libraries and/or modify them under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. These libraries are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License as available at <http://www.gnu.org/licenses/lgpl-2.1.html#SEC4> for more details.

19. jQuery v.1.7 libraries available at <http://jquery.com/> and jQuery UI v.1.8.2 libraries available at <http://jqueryui.com/>. Copyright © 2011 John Resig. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

20. JSON v. 2 libraries available at <http://www.json.org/json2.js>. Copyright © 2002 JSON.org. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. The Software shall be used for Good, not Evil. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

21. CodeMirror libraries available at <http://codemirror.net/>. Copyright © 2011 by Marijn Haverbeke marijnh@gmail.com. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. **THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.**

22. History.js available <https://github.com/balupton/History.js/> Copyright © 2011, Benjamin Arthur Lupton. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of Benjamin Arthur Lupton nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. **THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.**

23. Sample content provided for use with the software is copyrighted by the respective authors as contributed by each of them to Wikipedia pursuant to the Creative Commons Attribution-ShareAlike License found at <http://creativecommons.org/licenses/by-sa/3.0/>.

24. Visual Studio Tools for Office 3.0 (Microsoft VSTO 3.0) licensed under Microsoft Public License Microsoft Public License (Ms-PL) found at <http://msdn.microsoft.com/en-us/library/ff647676.aspx>.

25. The MarkLogic Connector for Hadoop contains Apache Jakarta Commons Modeler. Copyright 2001-2007 The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"). You may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

26. jemalloc <http://www.canonware.com/jemalloc/index.html>, released under the terms of the following BSD-derived licenses: Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice(s), this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice(s), this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER(S) ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER(S) BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. Portions of the software are copyright (C) 2002-2013 Jason Evans, and portions copyright (C) 2009-2013 Facebook, Inc. and copyright (C) 2007-2012 Mozilla Foundation. Neither the name of Facebook, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. All rights reserved.

27. Gziphelper; version: 1.0, Feb, 2003, authored by Gao Dasheng, is distributed subject to the following terms: This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions: (1.) the origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required; (2) altered source versions must be plainly marked as such, and must not be misrepresented as being the original software; (3) this notice may not be removed or altered from any source distribution.

28. Some features of MarkLogic allow you to develop and use Google Maps. Review Google Map's privacy policy at <http://www.google.com/policies/privacy/>, legal notices at http://maps.google.com/help/legalnotices_maps.html, and terms of use at http://maps.google.com/help/terms_maps.html.

29. commons-codec, HttpComponents, commons-logging, and Google Maps Utility are used under the Licensed under the Apache License, Version 2.0 (the "License"). You may not use these files except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

30. Mapstraction. Copyright (c) 2012, Mapstraction. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

31. logback-classic and logback-core are provided in object code form only under the Eclipse Public License - v 1.0 available at <http://www.eclipse.org/legal/epl-v10.html>. Copyright (C) 1999-2012, QOS.ch. All rights reserved. Source code for the Program is available <http://logback.qos.ch/>.

32. Simple Logging Facade for Java (SLF4J). Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

33. cURL v. 7.22.0 available at <http://curl.haxx.se/>. Copyright © 1996 - 2011, Daniel Stenberg, <daniel@haxx.se>. All rights reserved. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

34. heatmap.js distributed subject to the "THE BEER-WARE LICENSE" (Revision 42): <phk@FreeBSD.ORG> wrote this file. As long as you retain this notice you can do whatever you want with this stuff. If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Poul-Henning Kamp.

35. Except for components that are otherwise subject to other open source licenses, the MarkLogic ODBC Driver is released under the GNU Lesser General Public License as published by the Free Software Foundation, version 2.1 of the License.

36. Solr-commons-csv-3.5.0 Copyright 2005-2006 The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

37. Avro v. 1.7.1. Copyright © The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

38. Guava v. 11.0.2. Copyright © Google. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

39. Hadoop-auth v. 2.0.0. Copyright © The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

40. hadoop-hdfs v. 2.0. Copyright © The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

41. commons-lang v. 2.4, 2.5. Copyright © The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

42. commons-configuration v. 2.0. Copyright © The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

43. jackson-core-asl v. 1.8.8. Copyright © FasterXML. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

44. jackson-mapper-asl v. 1.8.8. Copyright © FasterXML. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

45. log4j v. 1.2.16. Copyright © The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, this file is distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

46. protobuf-java v. 2.4.0a. Copyright © Google. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of Google nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

47. xpp3 v. 1.1.3.3. Copyright © 2002 Extreme! Lab, Indiana University. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab(<http://www.extreme.indiana.edu/>)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. The names "Indiana Univeristy" and "Indiana Univeristy Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <http://www.extreme.indiana.edu/>. Products derived from this software may not use "Indiana Univeristy" name nor may "Indiana Univeristy" appear in their name, without prior written permission of the Indiana University. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS, COPYRIGHT HOLDERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

48. xstream v. 1.4.2. Copyright © 2003-2006, Joe Walnes. Copyright © 2006-2009, 2011 XStream Committers. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of XStream nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR

BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

TRADEMARK NOTICE

The MarkLogic name and logo are registered trademarks of MarkLogic Corporation in the United States and/or other countries. Excel and PowerPoint are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Any other names or logos included in the Software, this notice or the documentation are property of the respective trademark owners.