MarkLogic Server

Scalability, Availability, and Failover Guide

MarkLogic 6
September, 2012

Last Revised: 6.0-1, September, 2012

Copyright © 2012 MarkLogic Corporation. All rights reserved.

Table of Contents

Scalability, Availability, and Failover Guide

1.0 Introduction—About This Guide 5

2.0 Getting Started with Enterprise Edition Distributed Deployments 6
- 2.1 Terminology 6
- 2.2 Fundamentals 8
- 2.3 Advantages to Distributing 10
- 2.4 Considerations for a Distributed System 11
  - 2.4.1 Hosts with Distinct Roles 11
  - 2.4.2 Scale of System 12
  - 2.4.3 Hardware Configuration 12
  - 2.4.4 Storage Architecture 12
- 2.5 Configuring a Cluster 13
  - 2.5.1 Building a Cluster from Scratch 13
    - 2.5.1.1 Installing the Initial Host 13
    - 2.5.1.2 Configuring the Initial Host 14
    - 2.5.1.3 Filling Out the Cluster 14
    - 2.5.1.4 Configuring Forest Layout 15
  - 2.5.2 Increasing the Size of an Existing Cluster 15

3.0 Scalability Considerations in MarkLogic Server 16
- 3.1 Factors To Consider in Large-Scale Deployments 16
  - 3.1.1 Extremely Large Amounts of Content 16
  - 3.1.2 Fast Query Performance, Regardless of the Content Size 17
  - 3.1.3 Large Numbers of Users 17
- 3.2 Forest Sizes Per Data Node Host 17
- 3.3 High Availability 18
  - 3.3.1 High-Availability Features of MarkLogic Server 18
  - 3.3.2 Planning for High Availability 20
  - 3.3.3 High Availability With the Local-Disk and Shared-Disk Failover 20
- 3.4 Hardware and Memory Considerations 21

4.0 Clustering in MarkLogic Server 22
- 4.1 Overview of Clustering 22
- 4.2 Evaluator/Data Node Architecture 22
- 4.3 Communication Between Nodes 23
- 4.4 Communication Between Clusters 24
- 4.5 Installation 24

5.0 High Availability of Data Nodes With Failover 25
5.1 Problems Failover Addresses 25
5.2 Two Types of Failover 26
5.2.1 Local-Disk Failover 26
5.2.2 Shared-Disk Failover 28
5.2.3 Choosing Between Local-Disk and Shared-Disk Failover 29
5.3 How Failover Works 31
5.3.1 Failover at the Forest Level 31
5.3.2 Enabling Failover 31
5.3.3 Forest Mount States 31
5.3.4 Cluster Determines If a Host is Down 32
5.3.5 Different Host Takes Over the Forest 33
5.4 Requirements for Local-Disk Failover 33
5.4.1 Enterprise Cluster Required 33
5.4.2 Each Host Requires Sufficient Disk Space for Its Forests 33
5.5 Requirements for Shared-Disk Failover 33
5.5.1 Enterprise Cluster Required 34
5.5.2 Public Forest Required 34
5.5.3 Clustered Filesystem Required, Available to All Failover Hosts 34
5.6 Failover Host Must Be “Ready” to Take Over 34
5.7 Scenarios that Cause a Forest to Fail Over 35
5.8 Architecting a Failover Solution 35
5.8.1 Local-Disk Failover Architectures 36
5.8.2 Shared-Disk Failover Architectures 36
5.8.3 Deciding Which Type of Failover to Use 37

6.0 Configuring a Cluster 38
6.1 Determining Which Nodes Are in Which Group in a Cluster 38
6.2 Adding Another Node to a Cluster 38
6.3 Removing a Node from a Cluster 39
6.4 Upgrading a Cluster to a New Maintenance Release of MarkLogic Server 40

7.0 Configuring Local-Disk Failover for a Forest 42
7.1 Setting Up Local-Disk Failover for a Forest 42
7.1.1 Enabling Failover in a Group 42
7.1.2 Configuring Local-Disk Failover For a Forest 43
7.2 Reverting a Failed Over Forest Back to the Primary Host 44

8.0 Configuring Shared-Disk Failover for a Forest 47
8.1 Setting Up Shared-Disk Failover for a Forest 47
8.1.1 Enabling Failover in a Group 47
8.1.2 Setting Journaling Database Option to Strict 48
8.1.3 Configuring Shared-Disk Failover For a Forest 49
8.2 Reverting a Failed Over Forest Back to the Primary Host 51

9.0 Other Failover Configuration Tasks 52
9.1 XDQP Timeout, Host Timeout, and Host Initial Timeout Parameters 52
9.2 Disabling Failover for a Group or a Forest 53
9.3 Moving An Existing Private Forest to a Public Directory 54
9.4 Migrating the Security Database to Use Failover Forests 55
9.5 Migrating Other Auxiliary Databases to Use Failover Forests 56

10.0 Technical Support 57

Combined Product Notices 58
COPYRIGHT 58
TRADEMARK NOTICE 70

1.0 Introduction—About This Guide

This *Scalability, Availability, and Failover Guide* guide describes some of the features and characteristics that make MarkLogic Server scale to extremely large amounts of content. This documents assumes you are familiar with the installation and administration concepts and procedures for MarkLogic Server. For details on the installation procedure (including details on installing a cluster), see the *Installation Guide*. For details on MarkLogic Server concepts and administration procedures, see the *Administrator's Guide*. For details on the overall MarkLogic Server cluster architecture and terminology, see "Getting Started with Enterprise Edition Distributed Deployments" on page 6.

The topics of scalability and availability are combined in this guide because they tend to overlap each other. Additionally, the topic of clustering is covered because it is the mechanism used to scale a MarkLogic Server deployment. This guide is useful to anyone interested in the scalability and high-availability features of MarkLogic Server, and it also goes into details about planning for and configuring failover.

The following chapters in this guide supplement the material in the *Administrator's Guide*, *Application Developer's Guide*, and the *Installation Guide*:

- "Scalability Considerations in MarkLogic Server" on page 16 describes some of the motivations for why you might need scalability, describes the general concept of high availability as well as outlines some of the high-availability features in MarkLogic Server, and provides some rule-of-thumb guidelines for sizing systems.
- "Getting Started with Enterprise Edition Distributed Deployments" on page 6 explains the high-level concepts of MarkLogic Server clusters.
- "Clustering in MarkLogic Server" on page 22 reviews the basics of how MarkLogic Server clusters operate.
- "High Availability of Data Nodes With Failover" on page 25 describes the two types of failover, including describing the problems it addresses and listing the requirements for using it.
- "Configuring a Cluster" on page 38 describes the basic steps needed to configure a MarkLogic Server cluster.
- "Configuring Local-Disk Failover for a Forest" on page 42 provides the detailed steps to configure local-disk failover. This includes procedures to set up failover-enabled forests.
- "Configuring Shared-Disk Failover for a Forest" on page 47 provides the detailed steps to configure shared-disk failover. This includes procedures to set up failover-enabled forests.
- "Other Failover Configuration Tasks" on page 52 includes tasks applicable to both types of failover, including procedures to migrate forests that are not failover-enabled to become failover-enabled.

2.0 Getting Started with Enterprise Edition Distributed Deployments

MarkLogic Server Enterprise Edition supports large-scale high-performance architectures through multi-host distributed architectures. These architectures introduce additional complexity to both the planning and the deployment processes. This chapter introduces key concepts and terminology, outlines some alternative distribution strategies, and provides a high-level guide to configuring a cluster from scratch. The following topics are included:

- Terminology
- Fundamentals
- Advantages to Distributing
- Considerations for a Distributed System
- Configuring a Cluster

2.1 Terminology

It is important to understand the following terminology when considering a distributed implementation of MarkLogic Server:

*cluster*A cluster is a set of one or more instances (see hosts, below) of MarkLogic Server that will work together as a unified whole to provide content services.

*host*A host is a single instance of MarkLogic Server running on a single machine. MarkLogic Server Standard Edition can only be configured to run in single-host configurations. MarkLogic Server Enterprise Edition enables multi-host configurations.

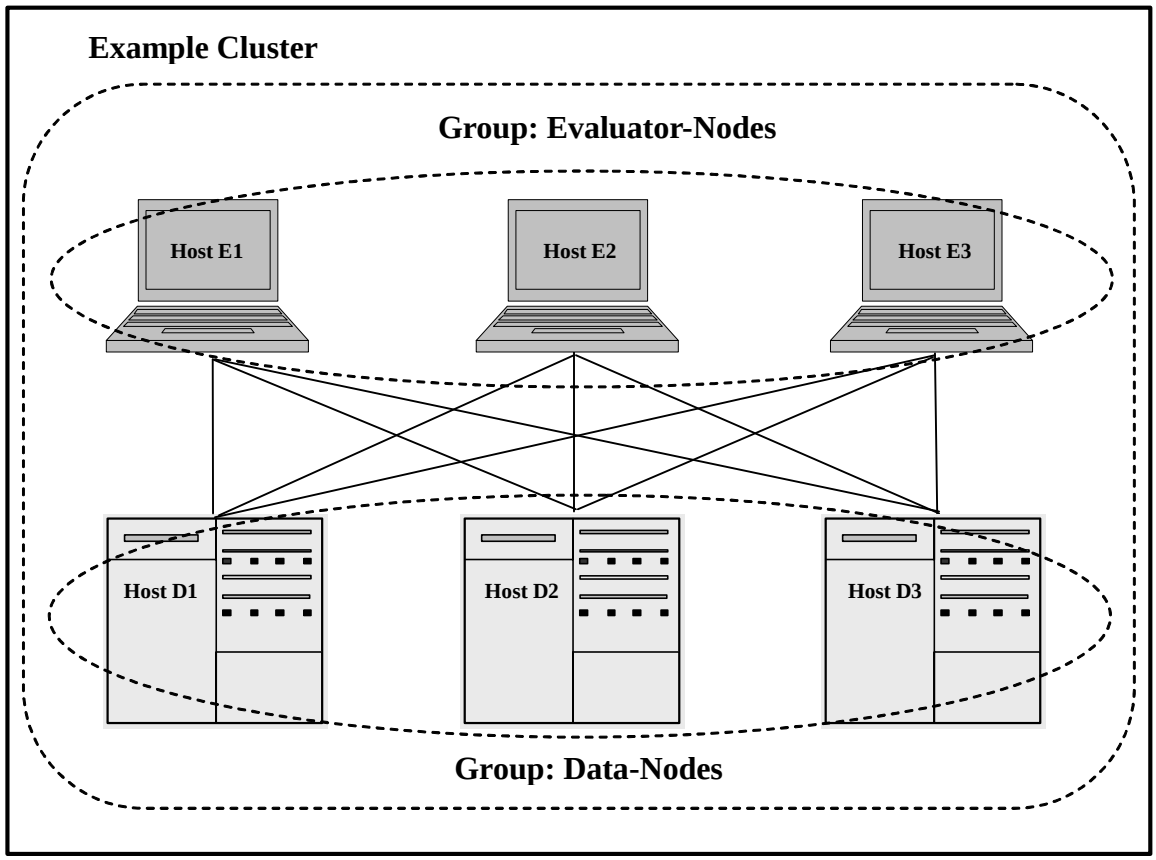
*group*A group is a set of hosts with uniform HTTP, WebDAV and XDBC server configurations (but not necessarily uniform forest configurations). Groups are used to simplify cluster management.

*forest*A forest is a repository for documents. Each forest is managed by a single host. The mapping of which forest is managed by which host is transparent to queries, as queries are processed against databases, not forests. Hosts can manage more than one forest simultaneously.

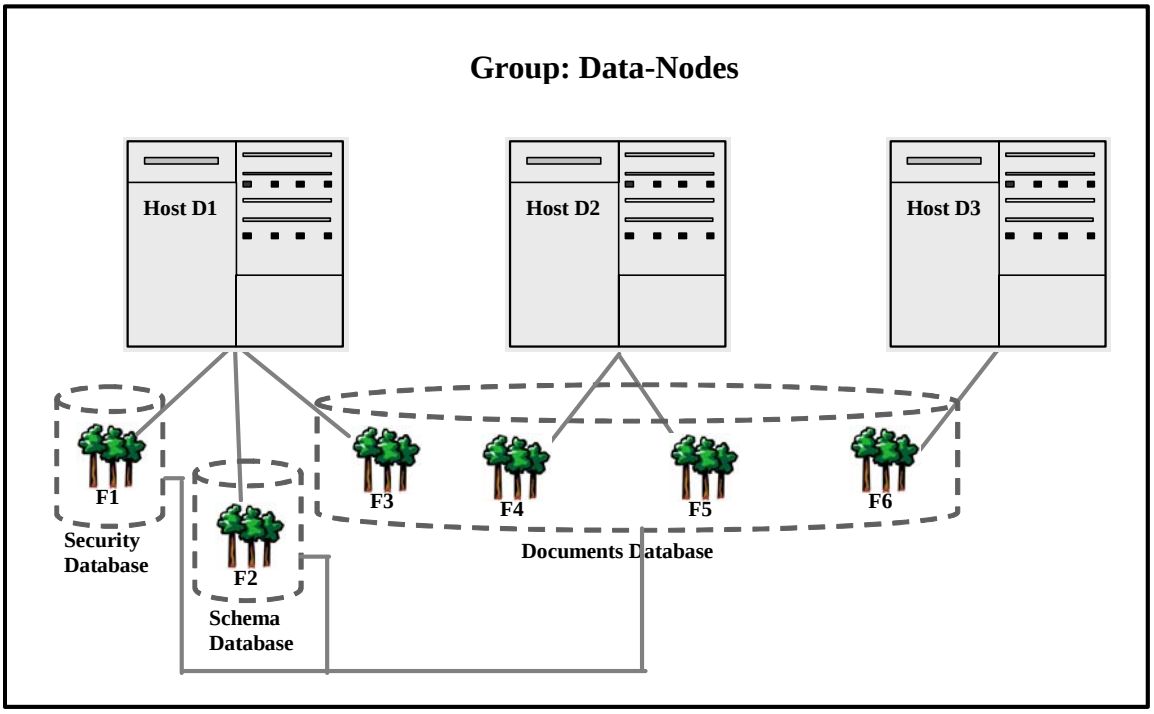
*database*A database is a set of one or more forests that appears as a single contiguous set of content for query purposes. Each forest in a database must be configured consistently. HTTP, WebDAV, and XDBC servers evaluate queries against a single database. In addition to databases created by the administrator for user content, MarkLogic Server maintains databases for administrative purposes: *security* databases, which contain user authentication and permissions information; *schema* databases, which are used to store schemas used by the system; *modules* databases, which are used to store executable XQuery code; and *triggers* databases, used to store trigger definitions. There are backup and restore utilities at both the forest and the database level.

*collection*A collection is a subset of one or more documents within a database. Documents can belong to more than one collection simultaneously. Collections can span multiple forests, and are not relevant to the cluster configuration process.

The following diagram illustrates how the concepts of clusters, hosts and groups are implemented in an example multi-host distributed architecture. The diagram shows a single cluster involving six hosts that are segmented into two groups:



This second diagram shows how a single database called “Documents” could comprise four forests spread across the three hosts in the group “Data-Nodes” from the above diagram. The diagram also illustrates two internal databases maintained by the system for administrative usage:



2.2 Fundamentals

With the key terminology introduced, here is a short list of fundamentals to keep in mind while considering any multi-host deployment:

1. *All hosts in a cluster must run on the same platform.*

Because MarkLogic Server takes advantage of platform-specific optimizations, it is not possible to mix and match platforms in a single cluster. This means that you will need to consider the pros and cons of different CPU architectures and operating systems across all hosts in your cluster when planning a multi-host deployment.

2. *Backups are not portable across platforms.*

Not only must clusters be platform homogeneous, the internal data formats used by forests are platform-specific. Consequently, forests and forest backups are not portable across platforms. To move database or forest content from a MarkLogic Server implementation running on one platform to an implementation running on another platform, the database must be unloaded from the first system and loaded from scratch into the second database.

3. *All hosts in a cluster run identical software.*

Even though each host in a cluster can be configured to perform a different task, the full MarkLogic Server software runs on each host. Consequently, software installation is identical across the nodes. However, hosts can be reconfigured rapidly—on an individual or cluster-wide basis—to perform whatever role is required of them. Of course, it is quite possible that different hardware configurations will be used for different hosts in the cluster, depending on the roles they are expected to play.

4. *Front-end query distribution is the responsibility of external infrastructure.*

Multi-host deployments may incorporate multiple hosts running the same HTTP, WebDAV, or XDBC server applications, allowing queries to be distributed between those hosts for improved performance. MarkLogic Server transparently manages the distribution of content storage across forests within databases and the distribution of query execution across the hosts managing those forests. However, MarkLogic Server does not manage the distribution of queries across front-end hosts configured with equivalent HTTP, WebDAV, or XDBC server applications. The multi-host deployment relies on surrounding infrastructure for this load-balancing task.

5. *There is no master/slave relationship in the cluster.*

Many distributed architectures rely on a single designated “master” to direct and manage the operations of the cluster. In MarkLogic’s distributed architecture, all hosts are considered equal, regardless of the roles for which they are configured. Consequently, there is no designated “master” in the cluster.

6. *Administration can be carried out through any host offering appropriate services.*

With no designated “master” in the cluster, administrative operations can be conducted on any cluster host configured to offer browser-based administrative operations or to accept XQuery-based administrative directives.

7. *Each forest is assigned to a single host.*

All queries that need to check for content stored in that forest or retrieve content from that forest will be routed to that host for processing. If you are using local-disk failover, you can replicate a forest across multiple hosts (for details, see “High Availability of Data Nodes With Failover” on page 25).

2.3 Advantages to Distributing

MarkLogic Server Standard Edition provides a powerful workgroup or departmental content platform. However, its single server architecture can constrain its scalability and performance in several ways:

- Query capacity is constrained by the number of CPUs that can be put to work processing queries simultaneously.
- Content volume can be constrained both by available address space and by the amount of memory installed in the server.
- Data access rates across multiple queries can be gated by the limited I/O path(s) inherent in single server architectures.

MarkLogic Server Enterprise Edition's distributed architecture has been designed specifically to address these issues. Distributed deployments can be used to support scalability and performance across multiple dimensions:

- Large volumes of content can be supported by increasing the number of hosts hosting forests.
- High data access rates can be supported by distributing the storage I/O requirements across more hosts.
- Large query volumes can be supported by increasing the number of hosts performing query evaluation.

In addition, MarkLogic Server Enterprise Edition's distributed architecture enables significantly more cost-effective implementations. Enterprise Edition allows data centers to distribute workload across a number of less-expensive entry-level servers or bladed infrastructures rather than requiring an investment in large expensive SMP systems.

For data centers with existing investments in SMP architectures, MarkLogic Server Enterprise Edition can still make effective use of those platforms.

2.4 Considerations for a Distributed System

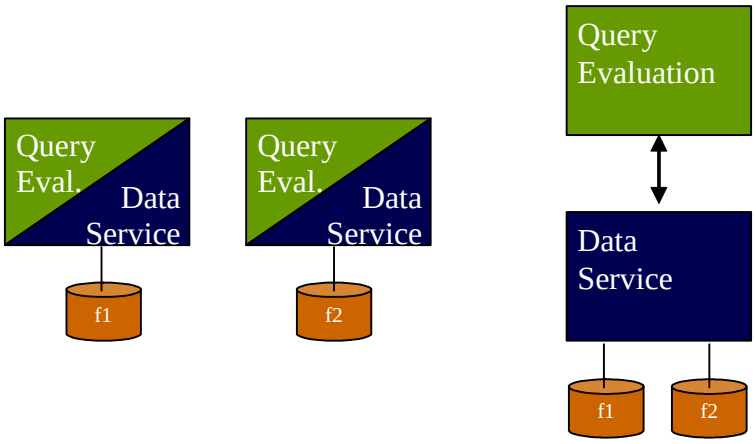
Just as there are different reasons to deploy a distributed architecture, there are different issues to consider in determining a distribution strategy. Each strategy provides certain benefits and involves certain other compromises. In order to determine how best to deploy MarkLogic Server to meet your requirements, you should obtain assistance from MarkLogic's professional services group.

This section includes the following topics:

- Hosts with Distinct Roles
- Scale of System
- Hardware Configuration
- Storage Architecture

2.4.1 Hosts with Distinct Roles

Distributed deployments of MarkLogic Server can be configured using hosts which all play similar roles or hosts which play distinct roles, as illustrated below.



Hosts with similar roles vs. hosts with distinct roles

A first-cut approach to small-scale distribution will have a number of hosts, each providing query evaluation and data service capability. This model can work well it static environments, and when the anticipated loads for each role map well.

In more dynamic environments, when the scale of the environment relative to individual host capacity is large, or when the anticipated scales of the query evaluation and data service loads differ markedly, the recommended approach is to partition the hosts into different roles. This architecture makes it easier to scale role-based capacity independently, and can also have some configuration advantages when different services operating at different scales are being offered out of the same cluster.

2.4.2 Scale of System

MarkLogic Server can be deployed on a small number of relatively larger-scale systems, or a larger number of relatively small-scale systems. Data center platform and system standards will be the primary driver for this architecture decision.

There is no requirement that all systems in a cluster be of the same scale. In fact, as outlined below, there are potential tuning advantages to be obtained from customizing systems within the cluster based on the role played by the hosts operating on that system.

2.4.3 Hardware Configuration

While MarkLogic Server requires that all hosts in a cluster run on the same platform, the systems for each host can be configured to optimize performance for that host's particular function.

In deployments architected following a distinct role strategy, it is to be expected that the data service nodes would be configured differently from the query evaluation nodes. The configuration differences will begin with storage connectivity, but will likely extend to memory configuration and possibly to processor count.

2.4.4 Storage Architecture

In addition to the alternatives outlined above, the underlying storage architecture can have a significant impact on scalability and performance.

Storage that is locally attached to each host offering data service can be cost- and performance-effective, but difficult to manage from a data center viewpoint. Centralized storage options include both NAS (network attached storage) and SAN (storage area network) approaches, with their own distinct trade-offs. Consult your MarkLogic professional services consultant for detailed deployment architecture advice.

2.5 Configuring a Cluster

When first configuring and deploying a cluster, the best place to start is with the overall architecture and distribution strategy. With the right strategy in hand, you can rapidly develop the actual configuration parameter sets that you will need. This section includes the following topics:

- Building a Cluster from Scratch
- Increasing the Size of an Existing Cluster

2.5.1 Building a Cluster from Scratch

There are four key stages to build a cluster from scratch:

- Installing the initial host in the cluster
- Configuring the initial host with the groups and databases that you will be using in the cluster
- Filling out the cluster by installing the additional hosts and adding them to the cluster
- Configuring the forest layout within the cluster.

The purpose of this process guide is to outline the overall workflow involved in setting up a cluster. For detailed explanations of each step below, see the *Installation Guide*. The following are the steps to building a cluster:

- Installing the Initial Host
- Configuring the Initial Host
- Filling Out the Cluster
- Configuring Forest Layout

2.5.1.1 Installing the Initial Host

To install the initial host in the cluster:

1. Install MarkLogic Server Enterprise Edition on one of the systems that will act as a host in the cluster. Follow the directions outlined in the section “Installing MarkLogic Server” in the *Installation Guide*.

   Note: The system on which you install the first host of the cluster will be the system on which the cluster’s default Security and Schema databases will reside. Consequently, do not choose a system on whose role is solely a query evaluator.

2. Start the server as outlined in the section “Starting MarkLogic Server” in the *Installation Guide*.

3. Accept the software license as outlined in the section “Accepting the License Agreement” in the *Installation Guide*.

   Note: Accepting this license agreement is optional. However, to begin using MarkLogic Server, you must accept the terms and conditions outlined in the license agreement. If you have executed a written software license with MarkLogic Corporation, the written license overrides the license displayed.

4. Configure the host by following the steps outlined in the section “Configuring a Single Host or the First Host in a Cluster” in the *Installation Guide*.

5. Configure an initial administrative user in response to the Security Setup screen.

Once the Admin Interface displays, you have completed the first stage in the process.

2.5.1.2 Configuring the Initial Host

To configure the initial host that you will be using in the system:

1. Access the Admin Interface through port 8001 on the initial host.

2. Use the Admin Interface to configure each of the databases that you will use in the cluster.

   Configuring Databases at this time will simplify the next step of configuring groups.

   Note: Because the cluster does not yet include all of its hosts, you will not be able to configure Forests at this time. Set up the databases without any forests.

3. Use the Admin Interface to configure each of the groups that you will use in the cluster.

   Configuring groups at this time will accelerate the process of deploying the cluster.

2.5.1.3 Filling Out the Cluster

To fill out the cluster by installing the additional hosts and adding them to the cluster, repeat the following steps for each system that will be a host in the cluster:

1. Install MarkLogic Server Enterprise Edition on the system. Follow the directions outlined in the section “Installing MarkLogic Server” in the *Installation Guide*.

2. Start the server as outlined in the section “Starting MarkLogic Server” in the *Installation Guide*.

3. Join the cluster as outlined in the section “Enterprise Edition: Configuring an Additional Host in the Cluster” in the *Installation Guide*.

   Refer to your distribution strategy and configuration plan to determine the Group to which this host should belong.

Once the Admin Interface displays, you have added this new host to the cluster. Repeat these steps for each host you want to add.

2.5.1.4 Configuring Forest Layout

To configure forest layout and complete database configuration across the cluster:

1. Access the Admin Interface through port 8001 on any of the hosts you have configured in a group that offers the Admin Interface.

2. Use the Admin Interface to configure forests throughout the cluster.

3. Use the Admin Interface to complete database configuration by attaching forests to the databases.

4. Use the Admin Interface to complete any other required configuration activities.

At this point, your cluster should be ready to load content and evaluate queries. See the *Administrator’s Guide* for more detailed information on any of the configuration steps outlined above.

2.5.2 Increasing the Size of an Existing Cluster

Adding a new host to an existing cluster is a simple task, outlined in the section “Enterprise Edition: Configuring an Additional Host in the Cluster” in the *Installation Guide*. Once the new host has been added, you may need to reconfigure how hosts in your cluster are used to redistribute workload. See the *Administrator’s Guide* for the appropriate procedures.

3.0 Scalability Considerations in MarkLogic Server

MarkLogic Server is designed for extremely large content sets, and to support these large content sets, scales to clusters of hundreds of machines, each of which runs MarkLogic Server. Each machine in a MarkLogic Server cluster is called a *host*, and a host is sometimes referred to as a *node* in the cluster. This chapter provides an overview of the goals of MarkLogic Server scalability, and includes the following sections:

- Factors To Consider in Large-Scale Deployments
- High Availability
- Forest Sizes Per Data Node Host
- Hardware and Memory Considerations

For information on how clusters work, see “Clustering in MarkLogic Server” on page 22.

3.1 Factors To Consider in Large-Scale Deployments

When determining the scale of a MarkLogic Server deployment, there are several factors to analyze. These factors are largely requirements-based. The size of a deployment, including how much hardware a deployment should use, will vary based on these and other factors. This section is not intended to provide an exact formula for determining machine requirements, but rather it highlights the types of questions you should be asking. For a specific requirements analysis of your environment, contact MarkLogic Consulting. The following topics are included:

- Extremely Large Amounts of Content
- Fast Query Performance, Regardless of the Content Size
- High Availability
- Large Numbers of Users

3.1.1 Extremely Large Amounts of Content

One of the primary drivers of a large-scale deployment of MarkLogic Server is a large amount of content. “Large” can mean many things, but typically it means 100s of gigabytes, terabytes (1000s of gigabytes), or more, measured in terms of raw XML. The disk space used by a database will range from less than the size of the XML content to approximately 10 times the size, depending on the indexing options, the number of range indexes, and how well the content compresses.

To handle extremely large content sets, you scale the number of data nodes in the MarkLogic Server cluster. MarkLogic Server is designed to scale to 100s of nodes or beyond.

3.1.2 Fast Query Performance, Regardless of the Content Size

Many applications have fast query performance as an important requirement. MarkLogic Server is built with solid foundations derived from both database and search engine architectures. Consequently, updates become available for querying as soon as they commit and queries against extremely large content sets return very quickly.

MarkLogic Server evaluates queries on an *evaluator node* (e-node), and the e-node gathers any needed content from the *data nodes* (d-nodes). If the content set is large and spread across several d-nodes, each of those d-nodes is involved in the query (even if only to inform the e-node that it has no content matching the query). The calls to the d-nodes are returned to the e-node in parallel, and because the content is indexed when it is loaded, the calls to each of the d-nodes return very quickly.

3.1.3 Large Numbers of Users

The number of users accessing an application is another dimension in which to measure scalability. In MarkLogic Server, users typically access an evaluator node, and a single e-node can service a large number of users. How large a number will depend on what the application is doing, how large the server is (for example, how much memory and how many processors/cores), as well as other factors. When the e-nodes are getting maxed out, you can scale the system by adding more e-nodes.

If your scalability challenge is large numbers of users, you can add e-nodes to a MarkLogic Server cluster and be able to see immediate benefits by routing queries across the various e-nodes in the cluster. The process of adding an e-node is simply a matter of bringing another MarkLogic Server machine online and joining that machine to the cluster.

3.2 Forest Sizes Per Data Node Host

As your content grows in size, you might need to add forests to your database. There is no limit to the number of forests in a database, but there are some guidelines for individual forest sizes where, if the guidelines are greatly exceeded, then you might see performance degradation.

The numbers in these guidelines are not exact, and they can vary considerably based on the content. Rather, they are approximate, rule-of-thumb sizes. These numbers are based on average sized fragments of 10k to 100k. If your fragments are much larger on average, or if you have a lot of large binary documents, then the forests can probably be larger before running into any performance degradation.

The rule-of-thumb maximum size for a forest on a 64-bit system is 200GB per forest, or 32-million fragments, whichever comes first. If the forest size grows past 32-million fragments, the system will start logging messages to the `ErrorLog.txt` file warning as to the number of fragments in the forest. All binary documents should be considered in fragment count estimations, but large and external binary documents may be excluded from the 200GB per forest limit since they contribute little to the size of stands within a forest. For example:

```
15M non-binary fragments
10M small binary fragments
10M large/external binary fragments
==> 35M fragments total, guideline exceeded

100 GB non-binary size
150 GB small binary size
1   TB large/external binary size
==> 150 GB non-binary and small binary size, guideline met
```

For details about binary content, see "Working With Binary Documents" in the *Application Developer's Guide*.

If the content is updated on a regular basis, each forest should ideally have two CPUs (or cores) per forest. That allows for concurrent operations, such as one core servicing merges while another core is servicing queries. For example, a dual processor/dual core machine should be able to host two 200GB forests, and a quad processor/dual core system should be able to host four 200GB forests.

It is a good idea to run performance tests with your own content. Indexing options, the size of the fragments, and many other factors can make larger forests work either better or not as well. If your forests are larger but your performance is still good, then the larger forest is probably OK for your setup.

3.3 High Availability

Another requirement that becomes increasingly important as an application becomes business-critical is *high availability.* A high availability system can continue functioning and processing queries, even if a part of the system goes down. For example, if a computer that hosts a data node has a hardware failure that causes it to shut down, a high availability system will be able to recover from such a failure. This section describes the following aspects of high availability in MarkLogic Server:

- High-Availability Features of MarkLogic Server
- Planning for High Availability
- High Availability With the Local-Disk and Shared-Disk Failover

3.3.1 High-Availability Features of MarkLogic Server

MarkLogic Server is designed for large-scale, production-class applications. These types of applications require fast and reliable performance, and also require robust recovery from power outages, application errors, or software failures.There are many features in MarkLogic Server designed to keep the server running and available, whether your deployment is a single instance of MarkLogic Server or has hundreds of servers:

- Fast automatic restart. If MarkLogic Server goes down unexpectedly for any reason, it automatically starts up again. The restart is very fast, often less than a few seconds, and the database remains transactionally consistent.
- Automatic, concurrent forest recovery. When MarkLogic Server starts up, if it needs to run recovery operations (for example, if it was in the middle of a transaction when the forest went offline), all forests can run recovery at the same time, speeding the time before the database is once again available.
- Tunable database parameters, helping you control factors such as recovery time. You can control database settings such as in memory limit, in memory list size, in memory tree size, and in memory range index size. Collectively, these settings control the size of a in-memory stands in a database, and you can tune these to optimize for updates or for recovery (thereby increasing availability), depending on your requirements.
- E-node query retry. Evaluator node operations (queries) will retry in the event of not getting a response from a d-node. If a transient failure occurred (for example, a d-node restarting), because recovery can be so fast, such retries can end up succeeding, even if they are the result of some problem. In these situations, the query would never even know a problem occurred because the system automatically corrected it.
- Online database backup operations. You can perform full and consistent database backup operations while the system is available.
- Hot configuration changes. Many configuration changes in MarkLogic Server happen *hot*; that is, without the need to restart the server. This makes it easier to make changes to environments while keeping all running applications available. There are a few configuration changes that require a server restart, but the majority are hot; for details, see Appendix A: 'Hot' versus 'Cold' Admin Tasks in the *Administrator's Guide*.
- Hot addition of nodes to the cluster. You can add nodes to a cluster without taking the cluster offline, allowing you to scale the cluster as your workload scales (similarly, you can remove nodes from a cluster without taking the cluster offline).
- Make configuration changes while some hosts are offline. You can make configuration changes to the cluster even if some hosts are not connected to the cluster. When the hosts come online, any configuration changes are automatically propagated to the new nodes in the cluster.
- Shared nothing architecture. All memory caches are local to each node in the cluster, allowing for scalability to clusters of 100s of nodes.
- Fast installation and upgrades. Installing a new release of MarkLogic Server is a simple and fast operation, requiring only a small window of downtime.
- You can perform administration activities on the cluster from any node in the cluster. You can run the Admin Interface on any node in the cluster, and you can perform nearly all administrative functions from any one of the nodes (the exceptions are leaving the cluster and changing your license key).

- Resiliency to data corruption. MarkLogic Server is designed to be resilient to corruption in the data store introduced in the underlying storage media. If a portion of the data does become corrupt, only queries that access that portion of the data are affected; other queries continue to function normally.
- Automatic index optimization and repair during merges. MarkLogic Server periodically merges forests. The merges optimize the indexes and also can repair any problems that might develop.
- Reindexing while the database is still available. If you make index changes to a database (for example, adding additional index options) and have reindexing enabled on that database, the database will remain available during reindex operations. It remains available with the previous index settings, and then when the reindexing operation completes, the new settings become available.

3.3.2 Planning for High Availability

When examining your system requirements, if you find that having extreme levels of system uptime is critical to your application, then high availability is clearly an important requirement for your system. When thinking about high availability for a system, you perform a thorough examination of all points of failure for your system, throughout the entire software and hardware stack in which your application runs.

From the MarkLogic Server point of view, consider how (and if) to deal with failures for both d-nodes and for e-nodes. For d-node failure, you can choose one of the failover types (see “High Availability With the Local-Disk and Shared-Disk Failover” on page 20). For e-node failure, the options are slightly different for applications that use the different types of App Servers (HTTP Servers, XDBC Servers, WebDAV Servers, and CPF-based applications or others that use the Task Server) available in MarkLogic Server. For example, a hardware or software-based load balancer or router might be appropriate for HTTP Server-based application, while you might want to code that logic directly into your XCC-based applications (for example, by taking advantage of the XCC Connection Provider SPI package to build failover capability directly into your XCC stack without modifying your applications).

As with any requirements, you tend to make trade-offs when deciding the best way to plan for high availability. Those trade-offs will consider factors such as the cost of potential system downtime, the cost of hardware and software, and the complexity of administration, and compare those costs with the benefits in terms of system availability. There is no one answer that suits everyone; the right answer for you depends on your own cost/benefit analysis.

3.3.3 High Availability With the Local-Disk and Shared-Disk Failover

In addition to the standard high-availability features listed above, MarkLogic Server provides high availability for content hosted on d-nodes with failover for forests. There are two types of failover: local-disk failover and shared-disk failover. Local-disk failover allows you to specify replica forests on another host to fail over to, and shared-disk failover allows you to specify

failover hosts for hosts that have forest data, and the failover host takes over the role of another host in the event of its becoming unavailable. Both types of failover detect when a host is down and perform the needed changes to have another host take over. For more details about failover, see "High Availability of Data Nodes With Failover" on page 25.

3.4 Hardware and Memory Considerations

MarkLogic Server is designed to take advantage of 64-bit systems. MarkLogic Server running on 64-bit systems will scale to sizes orders of magnitude higher than on 32-bit systems. For most deployments, a 64-bit system is recommended. In general, you should use 64-bit systems for all deployments, but particularly if your database size is greater than 2GB. Additionally, 64-bit systems can address much more memory than 32-bit systems, so you can both install more memory and address more virtual memory on a 64-bit system.

As your database size grows, having more memory on the system will generally make performance faster. MarkLogic Server caches content, indexes, and other information in memory, as well as memory maps selected indexes, including range indexes. Adding more memory allows you to raise memory settings (initially set during installation based on the memory on the machine), which can greatly improve performance. While memory can be relatively expensive, it does tends to have a good price/performance ratio.

MarkLogic Server is also designed to take advantage of multiple processors and/or multiple cores. Processors/cores add to the system scalability by allowing you to manage more forests on a host, allow a host to process more queries simultaneously, and allow greater concurrency for a wide range of system activities.

The faster and more scalable the hardware is, the faster certain long running or recurring administrative tasks can complete. For example, MarkLogic Server periodically merges forest content, removing obsolete versions of the content. Additionally, reindexing a database can be a resource intensive operation. These types of administrative tasks benefit from fast, scalable hardware with plenty of memory.

4.0 Clustering in MarkLogic Server

This chapter describes the basics of how clustering works in MarkLogic Server, and includes the following sections:

- Overview of Clustering
- Evaluator/Data Node Architecture
- Communication Between Nodes
- Communication Between Clusters
- Installation

4.1 Overview of Clustering

You can combine multiple instances of MarkLogic Server to run as a *cluster*. The cluster has multiple machines (*hosts*), each running an instance of MarkLogic Server. Each host in a cluster is sometimes called a *node*, and each node in the cluster has its own copy of all of the configuration information for the entire cluster.

When deployed as a cluster, MarkLogic Server implements a *shared-nothing* architecture. There is no single host in charge; each host communicates with every other host, and each node in the cluster maintains its own copy of the configuration. The security database, as well as all of the other databases in the cluster, are available to each node in the cluster. This shared-nothing architecture has great advantages when it comes to scalability and availability. As your scalability needs grow, you simply add more nodes.

For more details of how clustering works in MarkLogic Server, see “Getting Started with Enterprise Edition Distributed Deployments” on page 6.

4.2 Evaluator/Data Node Architecture

There are two roles a node in a MarkLogic Server cluster can perform:

- Evaluator node (e-node)
- Data node (d-node)

E-nodes evaluate XQuery programs, XCC/XDBC requests, WebDAV requests, and other server requests. If the request does not need any forest data to complete, then an e-node request is evaluated entirely on the e-node. If the request needs forest data (for example, a document in a database), then it communicates with one or more d-nodes to service the forest data. Once it gets the content back from the d-node, the e-node finishes processing the request (performs the filter portion of query processing) and sends the results to the application.

D-nodes are responsible for maintaining transactional integrity during insert, update, and delete operations. This transactional integrity includes forest journaling, forest recovery, backup operations, and on-disk forest management. D-nodes are also responsible for providing forest optimization (merges), index maintenance, and content retrieval. D-nodes service e-nodes when the e-nodes require content returned from a forest. A d-node gets the communication from an e-node, then sends the results of the index resolution back to the e-node. The d-node part of the request includes the index resolution portion of query processing. Also, each d-node performs the work needed for merges for any forests hosted on that node.

It is possible for a single node to act as both an e-node and a d-node. In single host configurations, both e-node and d-node activities are carried out by a single host. In a cluster, it is also possible for some or all of the hosts to have shared e-node and d-node duties. In large configurations, however, it is usually best have e-nodes and d-nodes operate on separate hosts in the cluster. For more details about this distributed architecture, see "Getting Started with Enterprise Edition Distributed Deployments" on page 6.

4.3 Communication Between Nodes

Each node in a cluster communicates with all of the other nodes in the cluster at periodic intervals. This periodic communication, known as a *heartbeat*, circulates key information about host status and availability between the nodes in a cluster. Through this mechanism, the cluster determines which nodes are available and communicates configuration changes with other nodes in the cluster. If a node goes down for some reason, it stops sending heartbeats to the other nodes in the cluster.

The cluster uses the heartbeat to determine if a node in the cluster is down. A heartbeat from a given node communicates its view of the cluster at the moment of the heartbeat. This determination is based on a vote from each node in the cluster, based on each node's view of the current state of the cluster. To vote a node out of the cluster, there must be a *quorum* of nodes voting to remove a node. A quorum occurs more than 50% of the *total* number of nodes in the cluster (including any nodes that are down) vote the same way. Therefore, you need at least 3 nodes in the cluster to reach a quorum. The voting that each host performs is done based on how long it has been since it last had a heartbeat from the other node. If at half or more of the nodes in the cluster determine that a node is down, then that node is disconnected from the cluster. The wait time for a host to be disconnected from the cluster is typically considerably longer than the time for restarting a host, so restarts should not cause hosts to be disconnected from the cluster (and therefore they should not cause forests to fail over). There are configuration parameters to determine how long to wait before removing a node (for details, see "XDQP Timeout, Host Timeout, and Host Initial Timeout Parameters" on page 52).

Each node in the cluster continues listening for the heartbeat from the disconnected node to see if it has come back up, and if a quorum of nodes in the cluster are getting heartbeats from the node, then it automatically rejoins the cluster.

The heartbeat mechanism enables the cluster to recover gracefully from things like hardware failures or other events that might make a host unresponsive. This occurs automatically, without human intervention; machines can go down and automatically come back up without requiring intervention from an administrator. If the node that goes down hosts content in a forest, then the database to which that forest belongs goes offline until the forest either comes back up or is detached from the database. If you have failover enabled and configured for that forest, it attempts to fail over the forest to a secondary host (that is, one of the secondary hosts will attempt to mount the forest). Once that occurs, the database will come back online. For details on failover, see "High Availability of Data Nodes With Failover" on page 25.

4.4 Communication Between Clusters

Database replication uses inter-cluster communication. Communication between clusters uses the XDQP protocol. Before you can configure database replication, each cluster in the replication scheme must be aware of the configuration of the other clusters. This is accomplished by coupling the local cluster to the foreign cluster. For more information, see Inter-cluster Communication and Configuring Database Replication in the *Database Replication Guide* and Clusters in the *Administrator's Guide*.

4.5 Installation

Installation of new nodes in a cluster is simply a matter of installing MarkLogic Server on a machine and entering the connection information for any existing node in the cluster you want to join. Once a node joins a cluster, depending on how that node is configured, you can use it to process queries and/or to manage content in forests. Under normal conditions, you can also perform cluster-wide administrative tasks from any host in the cluster. For details on the installation process, see the *Installation Guide*.

5.0 High Availability of Data Nodes With Failover

MarkLogic Server Enterprise Edition provides support for two varieties of failover at the forest level: local-disk failover and shared-disk failover. Local-disk failover allows you to specify a forest on another host to serve as a replica forest which will take over in the event of the forest's host going offline. Shared-disk failover allows you to specify alternate instances of MarkLogic Server to host forests in the event of a forest's primary host going offline. Both varieties of failover provide a high-availability solution for data nodes. This chapter describes failover, and includes the following sections:

- Problems Failover Addresses
- Two Types of Failover
- How Failover Works
- Requirements for Local-Disk Failover
- Requirements for Shared-Disk Failover
- Failover Host Must Be "Ready" to Take Over
- Scenarios that Cause a Forest to Fail Over
- Architecting a Failover Solution

For details about creating and managing forests and databases, see the *Administrator's Guide*. For details about configuring failover, see "Configuring Local-Disk Failover for a Forest" on page 42 and "Configuring Shared-Disk Failover for a Forest" on page 47.

5.1 Problems Failover Addresses

Failover for MarkLogic Server provides high availability for data nodes in the event of a data node failure. Data node failures can include operating system crashes, MarkLogic Server restarts, power failures, or persistent system failures (hardware failures, for example). With failover enabled and configured, a machine that hosts a forest can go down and the MarkLogic Server cluster automatically and gracefully recovers from the outage, continuing to process queries without any immediate action needed by an administrator.

In MarkLogic Server, if a forest becomes unavailable (for example, because of a hardware or power failure on the host managing the forest), then the database to which the forest is attached becomes unavailable for query operations. Without failover, such a failure requires an administrator to either reconfigure the forest to another host or to remove the forest from the configuration. With failover, you can configure the forest to automatically switch to a different host.

This "unattended" kind of recovery can make the difference between eliminating or greatly reducing down times and suffering prolonged and stressful down times. Sometimes, you can create administrative procedures to achieve similar goals, but these types of procedures are error prone, and can be risky if there are transactions updating content on the system.

Failover is designed to handle operating system and hardware failures on data nodes, which can take some time to repair. For example, if the operating system crashes, it can take many minutes or longer for the system to come back up; it can often take much longer if disk recovery is needed upon startup or if some part of the system does not come up correctly on restart. If MarkLogic Server itself suffers a failure or restarts for some reason, it typically restarts very quickly (often, it takes less than a few seconds). Because this restarting is so quick, failover should *not* occur during such events, and the data node will become available again on its own.

Failover provides both high levels of availability and data integrity in the event of a data node failure. Failover maintains data and transactional integrity during failure events. For example, it allows a single host to attempt any writing or recovery operations to a given forest at any particular time.

5.2 Two Types of Failover

Databases in a MarkLogic Server Enterprise Edition cluster have forests that hold their content, and each forest is served by a single host in the cluster. To guard against a host going down and being disconnected from the cluster, each forest allows you to set up one of two types of failover:

- Local-Disk Failover
- Shared-Disk Failover
- Choosing Between Local-Disk and Shared-Disk Failover

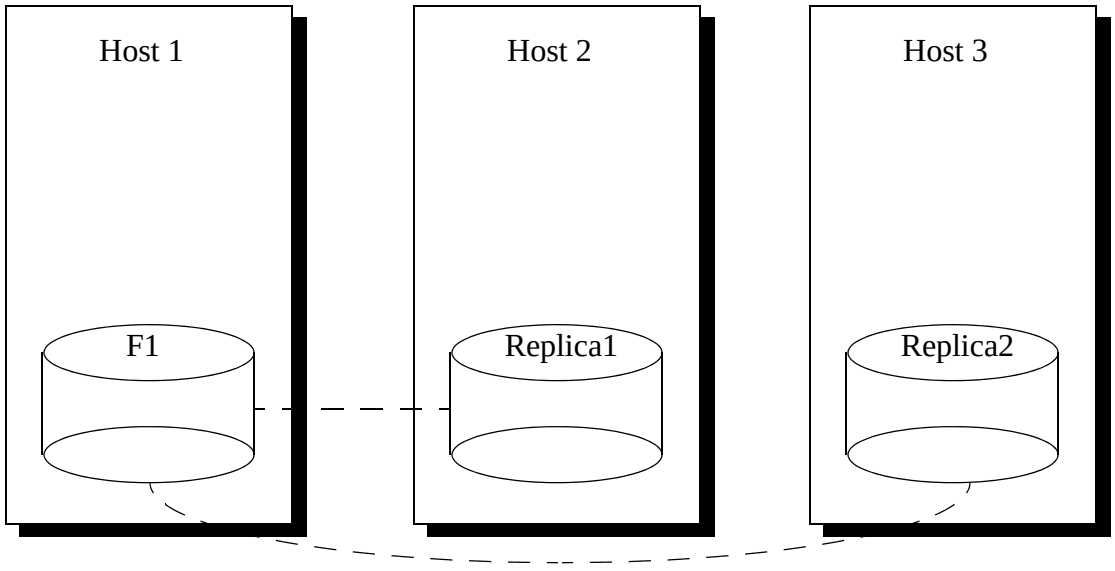
Both types of failover are controlled and configured at the forest level. This section describes the two types of failover and highlights the advantages of each.

5.2.1 Local-Disk Failover

Local-disk failover creates one or more replica forests for each failover forest. The replicas contain the exact same data as the primary forest, and are kept up to date transactionally as updates to the forest occur. Each replica forest should be on a different host from the primary forest so that, in the event of the host for the primary forest going down, another host with a copy of the primary forest's data can take over.

Each forest has its own host, and each host has disk space allocated for the forest. The primary forest is the forest that is attached to the database. Any replica forests configured for the primary forest have their own local disk. As updates happen to the database, the primary forest is updated as well as each configured replica forest.

The following figure shows three hosts that each have a forest configured. Host 1 has the primary forest (F1), and Host 2 and Host 3 have replica forests Replica1 and Replica2. The Replica1 and Replica2 forests each are stored disks local to their respective hosts. As updates are committed to the forest F1 (via transactions against the database to which F1 is attached), they are simultaneously committed to Replica1 and Replica2.



In the event that Host 1 goes down, the MarkLogic Server cluster will wait until the specified timeout and then it will automatically remove Host 1 from the cluster. Because all of the forests in a database must be available for queries to run, the database is unavailable at this time. At this point, the system fails over the forest to the first available replica forest, and the Replica1 forest is automatically attached to the database. The database once again becomes available. If the host that failed over its forest comes back online, that forest will continue on as a replica. If you want that forest to become the primary forest again, you must restart the forest that is currently the primary forest, as described in “Reverting a Failed Over Forest Back to the Primary Host” on page 51.

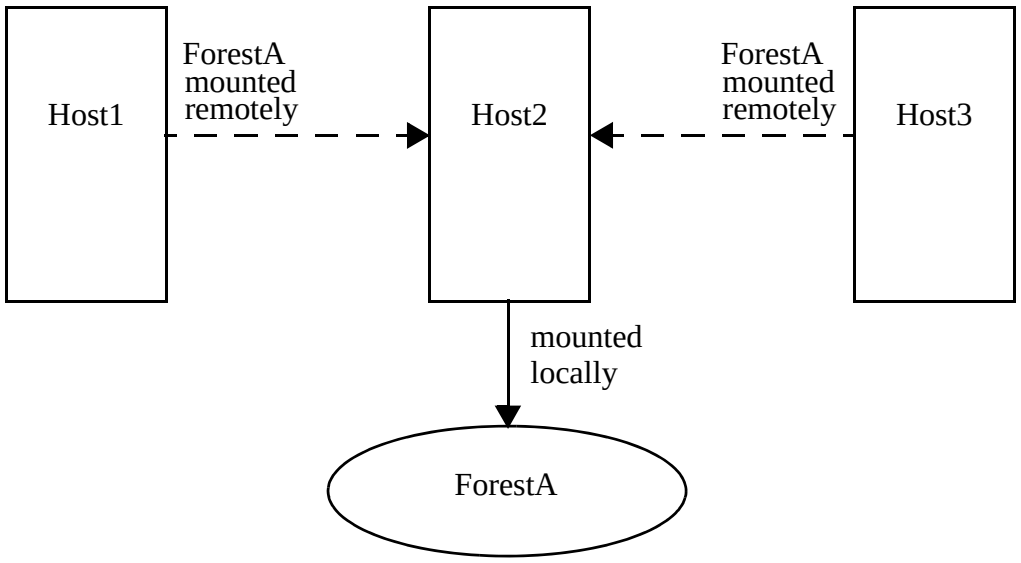
While each replica forest maintains all of the committed documents that are in the primary forest, it is not an exact byte-for-byte copy of the forest. It is a forest in its own right, and will merge when it needs to merge, which might not be at the same times that the primary forest merges. So assuming the replica forest is replicating normally (*sync replicating*, as described in “Forest Mount States” on page 31), it will contain the same committed documents as the primary forest. The host that services each replica forest must meet the needed disk space requirements and be sized appropriately so that merges can occur as needed on the forest.

5.2.2 Shared-Disk Failover

Shared-disk failover uses a clustered filesystem to store the forest data. The clustered filesystem must be available with the same path on each host that is configured as a failover host. In the event of a host that is assigned a shared-disk failover forest going down, another host can take over the assignment of that forest. For a list of supported clustered filesystems, see "Clustered Filesystem Required, Available to All Failover Hosts" on page 34.

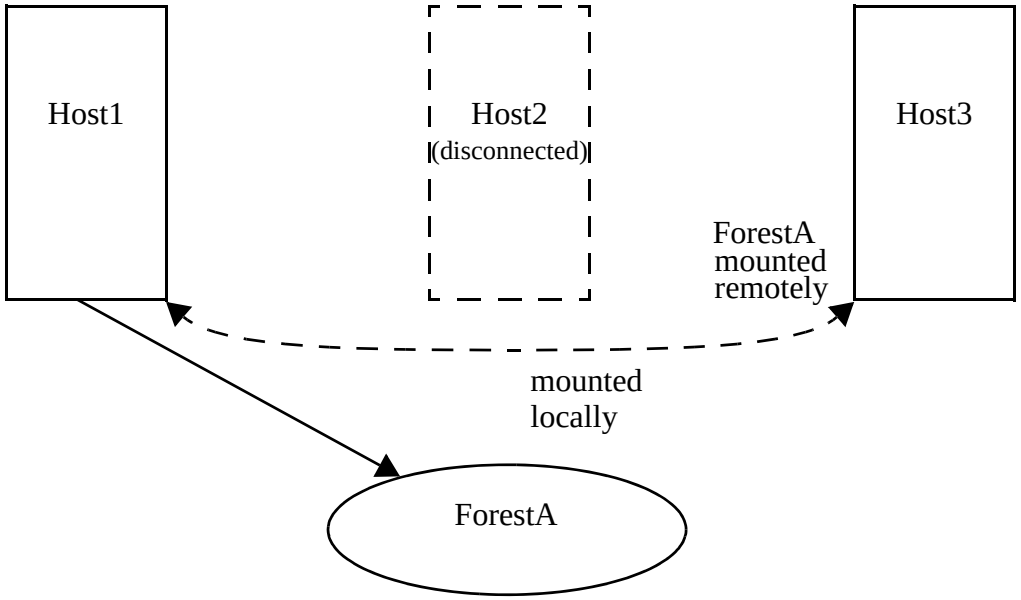
After the cluster has determined that a host is down and disconnected it from the cluster, if failover is enabled and configured for that forest, then one of the failover hosts will attempt to mount the forest locally. The host that attempts to mount the forest is determined based on the list of failover hosts in the forest configuration, and the host that is highest on the list will be the first to attempt to mount the forest. If that host is not available, then the next one in the list will attempt to mount the forest, and so on until the forest is either mounted or there is no failover hosts available. After the forest is mounted locally, the other hosts in the cluster will mount it remotely.

The following figure shows a healthy system with the forest mounted locally on Host2.



Healthy system with ForestA hosted by Host2

The following figure shows the same system after Host2 has been disconnected from the cluster, and Host1 has taken over by mounting ForestA locally.



Failed over system, with ForestA now hosted by Host1

If Host2 comes back online, it will automatically connect to the cluster. However, Host1 will continue to host the forest until the forest is restarted. This avoids having the forest "ping pong" between hosts in the event that the primary host has a recurring problem that takes some time to solve. For details on mounting the forest on the original host after the host has reconnected to the cluster, see "Reverting a Failed Over Forest Back to the Primary Host" on page 51.

5.2.3 Choosing Between Local-Disk and Shared-Disk Failover

As with most administrative choices, there are advantages and disadvantages to each type of failover. When choosing between local-disk and shared-disk failover, consider the following advantages, disadvantages, and costs of each.

The following are some of the advantages of local-disk failover:

- There is no need for an expensive and complex clustered filesystem.
- It keeps a complete and usable copy of the forest in each replica forest.
- You can use commodity storage for the filesystem.
- You can alleviate some of the need for backups of the forest (because there is a complete copy of the forest). Backups of the database are still recommended, but you always have at least another copy of the forest around when you have a replica forest on another host.

The following are some of the disadvantages of local-disk failover:

- Local-disk failover requires disk space for the primary forest as well as for each replica forest (including 3 times the forest size to accommodate for merges, as described in the MarkLogic Server requirements in the *Installation Guide*).

- Local-disk failover requires at least one replica forest per primary forest, so there will be more forests to create and manage.
- Because there are more forests to manage, there are more forests that you must monitor for system health.
- Each replica host requires the CPU capacity to update and merge the forest, even when the forest is not failed over.

The following are some of the advantages of shared-disk failover:

- There is only a single filesystem to manage (although it must be one of the filesystems described in “Clustered Filesystem Required, Available to All Failover Hosts” on page 34).
- It takes advantage of the high-availability features of a clustered filesystem.

The following are some of the disadvantages of shared-disk failover:

- Clustered filesystems can be complex and expensive.
- Shared disk failover does not keep multiple copies of the forest around on different filesystems (although clustered filesystems tend to have redundancy built in). You therefore do not have an up-to-date hot backup available.
- Forest that are configured for shared-disk failover always perform their updates using strict journaling mode, which explicitly performs a file synchronization after each commit. While strict journaling is safer because it protects against the computer unexpectedly going down and not just against MarkLogic Server unexpectedly going down, it makes updates slower. For details on strict versus fast journaling, see Understanding the Locking and Journaling Database Settings for Bulk Loads in the *Loading Content Into MarkLogic Server Guide*.

5.3 How Failover Works

This section describes the mechanism for how MarkLogic Server automatically fails an unresponsive host over to another computer, keeping replica forests on other hosts up to date (for local-disk failover) and keeping any forests hosted by the downed host available (for shared-disk failover). This section includes the following parts:

- Failover at the Forest Level
- Enabling Failover
- Forest Mount States
- Cluster Determines If a Host is Down
- Different Host Takes Over the Forest

The basic mechanism used for automatic failover is the same for local-disk failover (keep distinct replicated forests on different hosts) and for shared-disk failover (keep a single physical forest on a clustered filesystem available to multiple hosts). For details on the two types of failover, see "Local-Disk Failover" on page 26 and "Shared-Disk Failover" on page 28.

5.3.1 Failover at the Forest Level

D-node failover in MarkLogic Server is at the forest level, not at the database level. Each forest must have one or more failover hosts configured (for shared-disk failover) or one or more replica forests configured (for local-disk failover) before it can fail over to another host. You can configure up to as many failover hosts as the number of other hosts in the cluster, or up to as many replica forests as available forests in the cluster. For example, if you have 10 hosts in a cluster, you can configure between 1 and 9 failover hosts for a single forest.

5.3.2 Enabling Failover

You must enable failover before you can use it. Enabling failover requires failover enabled at the group level in the Admin Interface and failover to be enabled for the individual forest. The group-level parameter allows you to easily disable failover for an entire group (or for an entire cluster), allowing convenient failover administration on complex systems. For details on configuring failover, see "Configuring Shared-Disk Failover for a Forest" on page 47.

5.3.3 Forest Mount States

When any instance of MarkLogic Server starts up (or when a forest is restarted), each host tries to *mount* all of the forests configured for the entire cluster. A forest is mounted if MarkLogic Server has read the forest information from the forest. If the forest is hosted on the local instance of MarkLogic Server, then the forest is *mounted locally* on the host. If the forest is hosted on a different instance of MarkLogic Server (that is, on a different host), then the forest is *mounted remotely* on that host.

This local and remote mounting allows you to query both local and remote forests from any host in the cluster. Forest mounting occurs automatically on system startup or forest startup, and each time a forest is mounted, a message is logged in the `ErrorLog.txt` log file indicating if the forest is mounted remotely or mounted locally on that host. For shared-disk failover, the host that currently is servicing a forest is the one that has it mounted locally. The state of a forest that is mounted locally is *open*.

For local-disk failover, each forest can be in the following states:

- *sync replicating*: The forest is a stand-by replica and it is synchronously updated with the primary forest (that is, the one that is in the *open* state).
- *async replicating*: The forest is catching up to the primary forest, and it does this asynchronously. This typically occurs if you add a new replica forest to a forest that already has content in it. After it catches up, it will change to the *sync replicating* state.
- *wait replicating*: The forest is waiting to get into one of the other replicating states. Typically, this happens directly after forest or host startup or after a failover occurs. After it starts replicating, it will change to either the *sync replicating* or the *async replicating* state.

Both shared-disk and local-disk failover can have forests in error states:

- *open*: The forest is available and, in the case of local-disk failover, is acting as the primary forest.
- *error*: The forest is not available due to some error condition. This can occur for a variety of reasons, including insufficient disk space.

5.3.4 Cluster Determines If a Host is Down

Hosts in a MarkLogic Server cluster communicate their status periodically with each other via a heartbeat mechanism (for details, see “Communication Between Nodes” on page 23). This communication occurs whether failover is enabled or not. One of the purposes for this communication is to determine if any host has gone down.

The cluster uses a voting algorithm to determine if a host is down. The voting algorithm gets its data from each host’s view of the cluster. If there is a quorum of hosts, each of whose view of the cluster is such that it believes a particular host is down, then the other hosts in the cluster treat that host as if it is down and try to go on without it, *disconnecting* it from the cluster. If the disconnected host has no forests mounted locally, then everything else in the cluster can continue as normal; only requests initiated against the disconnected host will fail.

If the disconnected host had any forests mounted locally, however, then those forests will need to be either mounted to another host or detached from the database before any requests against that database can complete. If failover is not enabled and configured, then an administrator must perform those tasks. If failover is enabled and configured for a forest, however, then another host (a *failover* host) will attempt to mount that forest automatically.

5.3.5 Different Host Takes Over the Forest

After the cluster has determined that a host is down and disconnected it from the cluster, if failover is enabled and configured for that forest, then one of the failover hosts will attempt to take over hosting that forest. If you are using local-disk failover, then the replica host will take over with its copy of the forest, making it the primary forest. If you are using shared-disk failover, then the first configured failover host will attempt to mount the forest locally.

For more details on how the two types of failover work, see “Local-Disk Failover” on page 26 and “Shared-Disk Failover” on page 28.

5.4 Requirements for Local-Disk Failover

This section describes the requirements for setting up local-disk failover in MarkLogic Server, and includes the following requirements:

- Enterprise Cluster Required
- Each Host Requires Sufficient Disk Space for Its Forests

5.4.1 Enterprise Cluster Required

Because local-disk failover requires another host available to take over the failed host, you need a cluster configured in order to set up failover. Clusters require Enterprise Edition of MarkLogic Server. Additionally, a minimum of three hosts are required for the cluster. Three or more hosts ensure reliable voting to determine if a host is offline, which in turn determines if forests need to be failed over.

5.4.2 Each Host Requires Sufficient Disk Space for Its Forests

To provide for merges, each host requires the disk space necessary to manage any forests assigned to the host, including replica forests. Each forest needs to be on a filesystem that is 3 times the size of the forest. For details on disk space requirements, see the *Installation Guide*. The disk space is important because, if a forest does not have sufficient disk space, you will not be able to perform any updates to that forest and it will go into an error state. You cannot fail over to a replica forest that is in an error state. To correct an error state caused by insufficient disk space, you must either free space on the device or add space to the device, and then restart the forest.

5.5 Requirements for Shared-Disk Failover

This section describes the requirements for setting up shared-disk failover in MarkLogic Server, and includes the following requirements:

- Enterprise Cluster Required
- Public Forest Required
- Clustered Filesystem Required, Available to All Failover Hosts

5.5.1 Enterprise Cluster Required

Because failover requires a host available to take over the forest from the failed host, you need a cluster configured in order to set up failover. Clusters require Enterprise Edition of MarkLogic Server. Additionally, a minimum of three hosts are required for the cluster. Three or more hosts ensure reliable voting to determine if a host is offline, which in turn determines if forests need to be failed over.

5.5.2 Public Forest Required

To enable shared-disk failover for a forest, the forest data directory must be accessible from the primary host and the failover host(s), and it must have the same directory path on each host. Such a forest is known as a *public* forest, where the data can be accessed by multiple hosts. It cannot be in the default data directory. If the forest is stored in the default data directory for MarkLogic Server (for example, `/var/opt/MarkLogic` or `c:/Program Files/MarkLogic`), it is known as a *private* forest, and it cannot be used with failover; you first move that forest to a public directory (for a procedure, see "Moving An Existing Private Forest to a Public Directory" on page 54) before it can be configured as a failover forest.

5.5.3 Clustered Filesystem Required, Available to All Failover Hosts

In addition to a public forest directory (described above), shared-disk failover requires the forest to have its data directory reside on a supported clustered filesystem (CFS). The CFS must be accessible from the primary host and the failover host(s), and it must have the same directory path on each host.

Shared-disk failover in MarkLogic Server is supported on the following CFSs:

- Veritas VxFS 5.0 on Sun Solaris 10
- Veritas VxFS 4.1 on 64-bit Red Hat 5 Linux
- Veritas VxFS 5.0 on 64-bit Red Hat 5 or Red Hat 6 Linux
- Red Hat GFS on 64-bit Linux (build 2.6.9-58.3 or greater)
- Red Hat GFS2 on 64-bit Red Hat 5.4 Linux

5.6 Failover Host Must Be "Ready" to Take Over

When you configure a host as a failover host, it should be ready to take over the job of hosting the forest. A failover host must be connected to the cluster in order for a forest to fail over to it, and it must be online and available at the time of host failure. Also, the machine should be capable of handling the load of hosting the forest or forests that fail over to it, at least temporarily.

If the failover host already hosts other forests, this means that the machine needs to be able to handle the additional load that may be placed on it in the event of a failover. For example, if a host manages two forests and is configured as a failover host for a third, the machine should have sufficient memory and CPU resources to manage the third forest, at least temporarily. In more

complex deployments, you should consider the potential load induced by the failure of multiple hosts. In some cases, the cause of system failure can impact multiple hosts, and failover system can themselves fail as a result of such a multi-host concurrent or sequential failure not being anticipated during original system architecture.

5.7 Scenarios that Cause a Forest to Fail Over

In general, failover provides the ability for a forest to change hosts in the event that its host becomes unresponsive for some reason. This section lists the scenarios in which a forest will fail over to another host, assuming the forest and the cluster are configured to failover. For more information about how failover works, see "How Failover Works" on page 31. For information about the forest mount states, see "Forest Mount States" on page 31.

A forest will fail over in the following scenarios:

- The host that currently has the forest mounted locally is shut down. After the host is shut down, the other hosts in the MarkLogic Server cluster will stop getting heartbeats from the shut down host, and will disconnect it from the cluster. This causes one of the other configured failover hosts to mount the forest locally.
- The host that currently has the forest mounted locally becomes unresponsive to the rest of the cluster. The other hosts determine a host's responsiveness based on heartbeats it receives (or heartbeats it does not receive) from that host. When a quorum of the cluster determines a host is unresponsive (that is, they have not received a heartbeat for the `host timeout`), it disconnects from the cluster. If the disconnected host is actually still alive and still has the forest mounted locally, then the forest will not fail over; it will only fail over after the host that has it mounted locally stops accessing the forest (to prevent multiple hosts writing to the forest at the same time). For more details on the host timeout parameters, see "XDQP Timeout, Host Timeout, and Host Initial Timeout Parameters" on page 52.
- Restarting the forest. When you click the `restart` button on the Forest Status page in the Admin Interface, the forest is unmounted by all hosts. It then is automatically mounted by the primary host (or if the primary host is unavailable, by a failover host).

5.8 Architecting a Failover Solution

This section describes things to think about when designing an architecture for your failover solution, and includes the following parts:

- Local-Disk Failover Architectures
- Shared-Disk Failover Architectures
- Deciding Which Type of Failover to Use

5.8.1 Local-Disk Failover Architectures

A simple deployment for local-disk failover is to have one replica forest for each primary forest in the database. The hosts that have the replica forests must be sized appropriately in terms of CPU, memory, and disk space to perform all of the duties of hosting the forest (for example, updates and merges). If the replica forest hosts had the same number of forests as the primary forest hosts, then the replica hosts would need the same configuration as the primary hosts.

You might decide to use slower and/or less expensive machines to host the replica systems. This would likely mean that performance of the application might suffer some during the failed over period, but if that is acceptable to your application, and if your operations team is set up to restore the failed system in a timely fashion, then that might be a tradeoff worth considering.

Whatever tradeoffs you decide are acceptable, you must make sure that the replica hosts can at least handle the load of hosting the forest in the event of a host fails over. It might be acceptable to have a period of somewhat decreased performance after a failover, but it likely is not acceptable for the system to be inadequate for the load.

5.8.2 Shared-Disk Failover Architectures

The simplest conceptual deployment for shared-disk failover is to have a secondary “shadow” host in the cluster with identical hardware configuration to your target d-node. If the d-node in question fails, the “shadow” host can take over for it and provide forest services at exactly the same levels of performance as you had previously.

Of course, this architecture is overly simplistic for many deployments. For instance, if you had multiple d-nodes that you wanted to be able to fail over, this approach would require multiple “shadow” hosts sitting around waiting for failure. Alternatively, with shared-disk failover, the same “shadow” host could be designated as the failover target for all of the d-nodes, in which case should two d-nodes fail in the same time period, the “shadow” host could easily be overwhelmed.

The simplest model also fails to take advantage of the latent performance and capability of the “shadow” host during periods of normal operations. Given that normalcy is what one strives for in operations, this seems like a lost opportunity to deliver additional levels of performance during normal operations.

Consider a three-host cluster deployed as described in the diagrams “Shared-Disk Failover” on page 28. One host might be used for e-node services, one for d-node services, and one acts as the “spare” awaiting d-node failure. These three hosts might be used more profitably if the d-node services were divided between the d-node and the spare—each host hosting half of the forests in your databases. In this configuration, each of the two d-nodes could act as the failover host for forests mounted locally on the other host. Consequently, failover can happen in either “direction”. During periods of normal operations, the cluster has twice the resources available for d-node services.

In a larger cluster, one can consider configuring a series of forests and hosts in an "n-way" format, so that if a single host fails, its forests are all failed over to different failover hosts, thereby maximally distributing the incremental load encountered by any one host, and minimizing any discernible change in performance felt by the end user. So long as the d-node hosts have sufficient resources to be able to support the additional forest load, such a configuration may be able to absorb the failure of multiple hosts in the cluster, and can continue to service queries at levels that meet or exceed your committed service levels. Alternatively, if you chose not to "overspec" your hardware with respect to your service level commitments during periods of normal operations, such a configuration may be able to continue servicing queries at a (potentially significantly) degraded level of performance, depending on your system load.

5.8.3 Deciding Which Type of Failover to Use

When deciding how to implement your failover solution for your forests, you must first decide which type of failover to use: local-disk or shared-disk. Local-disk failover keeps a replica of the forest, so it provides redundancy as well as failover protection. Shared-disk failover allows you to manage fewer forests (because you do not have to create replicas) with the tradeoff of requiring a clustered filesystem to store the forest data. With local-disk failover, the hosts that manage the replica forest remain active as the forest gets updated. With shared-disk failover, the failover hosts do not have to do much work while they are standing by to be failed over to. You must analyze your own requirements and environment to choose the failover type that is right for your environment. For more details, see "Choosing Between Local-Disk and Shared-Disk Failover" on page 29.

Architecting a production environment for failover involves combining the functional support for failover available in MarkLogic Server with your operational procedures, your software license agreements (SLAs), the magnitude of system failure you need to be able to sustain, and your budget for maintaining excess system capacity. The MarkLogic Professional Services organization can help you identify trade-offs and make the appropriate decisions for what architecture makes sense for your needs.

6.0 Configuring a Cluster

This chapter describes some of the basic configuration tasks for a cluster. When you have a cluster, you can perform most administrative tasks from the Admin Interface of any node in the cluster. This chapter describes some of those cluster tasks and includes the following sections:

- Determining Which Nodes Are in Which Group in a Cluster
- Adding Another Node to a Cluster
- Removing a Node from a Cluster
- Upgrading a Cluster to a New Maintenance Release of MarkLogic Server

6.1 Determining Which Nodes Are in Which Group in a Cluster

The Admin Interface has summary and status screens to help you determine the makeup of a cluster. The system summary page (the default Admin Interface page when you log in) has links to each group and to each host in the cluster. If you have a cluster with multiple groups configured and you click on a group, the Admin Interface displays a group summary page which shows all of the hosts in a group. For each group, there is also a status page.

Similarly, each host has a summary page and a status page. For more details about the status pages, see Monitoring MarkLogic Server Performance in the *Query Performance and Tuning Guide* book.

6.2 Adding Another Node to a Cluster

This section describes the general procedure for adding a new node to an existing cluster. For more detailed installation instructions, including the procedure to add a new node to a cluster, see the *Installation Guide*.

The following are the general steps needed to add another node to an existing cluster (the existing cluster can have one or more hosts):

1. Install MarkLogic Server on the new host. All hosts in the cluster must run the same version of MarkLogic Server.

2. Access the Admin Interface on the new host. For example, if the host is installed on a machine named `raymond`, access http://raymond:8001.

3. On the License Key Entry page, enter a valid licensee and corresponding license key. You need an Enterprise license key to be part of a cluster. The license keys are specific to a named machine. MarkLogic Server will restart after you enter the license key.

4. Accept the license agreement. MarkLogic Server will restart after you accept the license agreement.

   Note: Accepting this license agreement is optional. However, to begin using MarkLogic Server, you must accept the terms and conditions outlined in the license agreement. If you have executed a written software license with MarkLogic Corporation, the agreement displayed references that written agreement.

5. On the Server Install page, click OK to install the initial servers. MarkLogic Server will restart after you click OK.

6. On the Join a Cluster page, enter the name of any host in the cluster, with the port number of its Admin Interface. Then Click the OK button.

7. On the next Join a Cluster page, select the group to which you want to join. Then Click the OK button.

8. On the next Join a Cluster page, click OK to confirm that you want to join a cluster.

9. On the Joined a Cluster page, click OK to transfer cluster information to the new node in the cluster. MarkLogic Server will restart on the new node after you click OK.

10. After the MarkLogic Server restarts on the node, it prompts you for a username and password. Enter a username and password for the Admin Interface.

The new node is now part of the cluster.

6.3 Removing a Node from a Cluster

If you want to remove a node from a cluster, you must log into the Admin Interface for that node, go to its host configuration page and click the `leave` button.

Before you remove a node from a cluster, make sure it is not hosting any forests for the cluster (the forest summary page shows this information) or is not acting as a failover host for any forests. Additionally, make sure it is not coupled to any other hosts as a foreign cluster. You must resolve any forest dependencies with other hosts and de-couple any foreign clusters before you can remove a host from the cluster; the Admin Interface will not allow you to remove a host from the cluster if there are other hosts depending on it, and it will attempt to help you to resolve the dependencies. That might involve removing the host as a failover host, detaching the forest from the database, or assigning the forest to another d-node. For details about managing forests, see the *Administrator's Guide*.

Perform the following basic procedure to remove a host from a cluster:

1. Access the Admin Interface for the node you want to remove from the cluster.

2. Navigate to the host configuration page for the host, either through the tree menu (the Configure tab under Hosts > *host_name*) or from one of the summary pages.

3. To leave the cluster, click the `leave` button. The `leave` button only appears when you are using the Admin Interface on which the host machine is running. On the Admin Interface on other nodes in the cluster, there is an `admin` button, which is a link to open the Admin Interface running on that host.

4. On the Leave Cluster page, click OK to confirm. MarkLogic Server on that host will restart, and the host is removed from the cluster.

At this point, the installation process will begin again (although you will not have to reenter the license key information). You can choose to join the cluster again, join another cluster, or skip the cluster installation and install as a stand-alone host.

6.4 Upgrading a Cluster to a New Maintenance Release of MarkLogic Server

Each host in a MarkLogic Server cluster must run the same version of the MarkLogic software. Therefore, if you install a maintenance release, you need to install it on all of the hosts in the cluster.

> **Note:** The security database and the schemas database must be on the same host, and that host should be the first host you upgrade when upgrading a cluster. If two clusters are using database replication to replicate the security database, see the procedure described in Upgrading from Release 6.0-1 or Later in the *Installation Guide*.

The upgrade installation is relatively quick, and should require only a small amount of cluster down time. It does require each host to shut down, uninstall, reinstall, and start up, however.

For a more detailed upgrade procedure, see the *Installation Guide*. The general procedure for upgrading a cluster is as follows:

1. Download the maintenance release of MarkLogic Server you plan on installing. Make sure to download the correct release for your platform.

2. Shut down all of the hosts in the cluster. For example, on the Cluster Status page, click the Shutdown button.

3. After the hosts have finished shutting down, uninstall each instance of MarkLogic Server (from the Add/Remove Programs Control Panel on Windows, with `rpm -e MarkLogic` on Red Hat Linux, and with `pkgrm MARKlogic` on Solaris). For details, see the *Installation Guide*.

4. Install the new release of MarkLogic Server on each host (with the installer on Windows, `rpm -i` on Red Hat Linux, `pkgadd MARKlogic` on Solaris). For details, see the *Installation Guide*.

5. Start up MarkLogic Server on the machine that hosts the security database.

6. Start up MarkLogic Server on all of the other hosts.

7. Access the Admin Interface on the machine in which the security database forest is hosted.

8. When prompted, enter the username and password. The license agreement page appears.

9. Accept the license agreement, either for that host (Accept button) or for all of the hosts in the cluster (Accept for Cluster button). If you choose the Accept for Cluster button, a summary screen appears showing all of the hosts in the cluster. Click the Accept for Cluster button to confirm acceptance (all of the hosts must be started in order to accept for the cluster).

   Note: Accepting this license agreement is optional. However, to begin using MarkLogic Server, you must accept the terms and conditions outlined in the license agreement. If you have executed a written software license with MarkLogic Corporation, the agreement displayed references that written agreement.

10. If you accepted the license just for the one host in the previous step, you must go to all of the Admin Interface for all of the other hosts and accept the license for each host before each host can operate.

This concludes the upgrade, and the upgraded cluster is now available to accept requests.

7.0 Configuring Local-Disk Failover for a Forest

This chapter describes the procedure for configuring local-disk failover for a forest. For details about how failover works and the requirements for failover, see "High Availability of Data Nodes With Failover" on page 25. For details on configuring shared-disk failover, see "Configuring Shared-Disk Failover for a Forest" on page 47. This chapter includes the following sections:

- Setting Up Local-Disk Failover for a Forest
- Reverting a Failed Over Forest Back to the Primary Host

For other failover administrative procedures that apply to both local-disk and shared-disk failover, see "Other Failover Configuration Tasks" on page 52.

7.1 Setting Up Local-Disk Failover for a Forest

Setting up failover for a forest is a relatively simple administrative process. This section describes this procedure. There are two basic parts to the procedure:

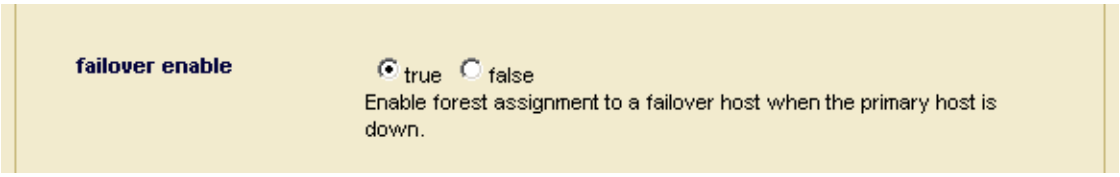
- Enabling Failover in a Group
- Configuring Local-Disk Failover For a Forest

7.1.1 Enabling Failover in a Group

For each group in which you want to host a failover forest, perform the following steps:

1. Before setting up failover, ensure that you have met all the requirements for failover, as described in "Requirements for Local-Disk Failover" on page 33.

2. In the groups configuration page for the group in which the failover host belongs, make sure the `failover enable` button is set to `true`.



   This group-level `failover enable` button provides global control, at the group level, for enabling and disabling failover for all forests in that group.

You can enable or disable failover at the group level at any time. It does not stop you from configuring failover for forests; it only stops failover from actually occurring.

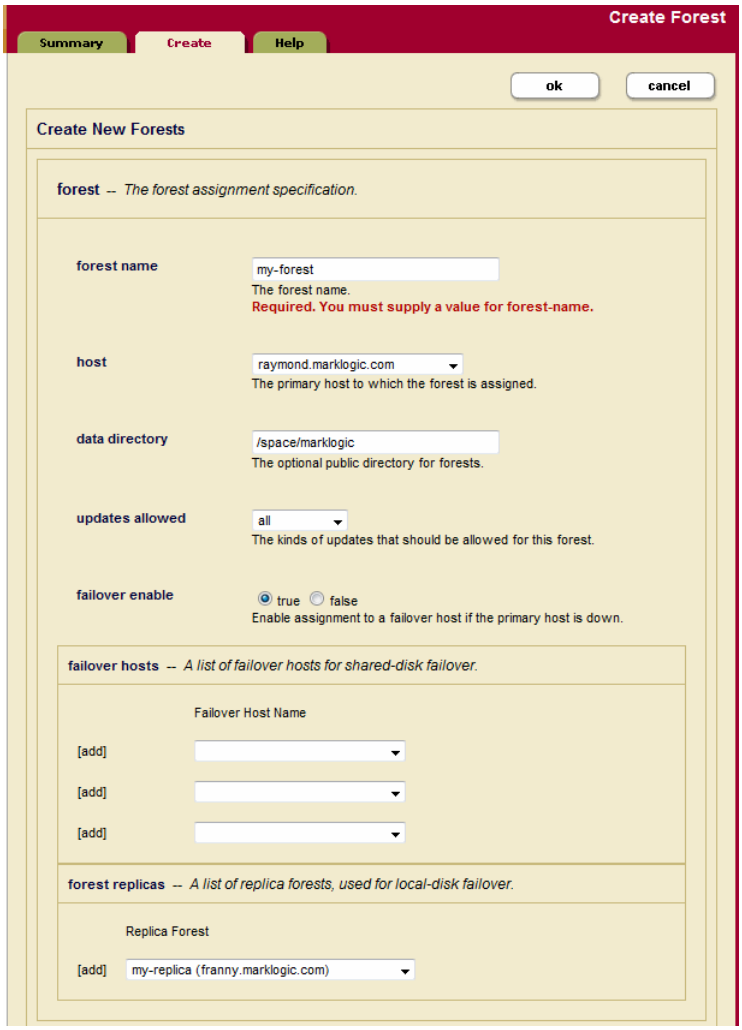
7.1.2 Configuring Local-Disk Failover For a Forest

To set up local-disk failover on a forest and set up one or more replica forests, perform the following steps:

1. Before setting up failover, ensure that you have met all the requirements for failover, as described in “Requirements for Local-Disk Failover” on page 33 and enable failover for the group, as described in “Enabling Failover in a Group” on page 42.

2. Create one or more forests on one or more hosts to use as a replica (or use existing forests).

3. Either create a new forest or use an existing forest on a different host from the replica forest(s). This forest will be the primary forest. If you are modifying an existing forest, skip to step 6. To create a new forest, first click the Forests link in the left tree menu, then click the Create tab. The Create Forest page appears. Note the `failover enable` button and the `failover hosts` section at the bottom.



4. Enter a name for the forest.

5. Specify a data directory.

6. Select `true` for `failover enable`. Note that `failover enable` must be set to `true` at both the forest and the group level for failover to be active.

7. Select the replica forest from the drop down menu for `forest replicas`. You can set one or more replica forests.

8. Click OK to create or modify the forest.

The forest is now configured with the specified replica forests. You must attach the primary forest to a database before you can use the forest, but it is ready and set up for failover. You cannot attach the replica forests to a database; they will automatically be kept up-to-date as you update content in the primary forest.

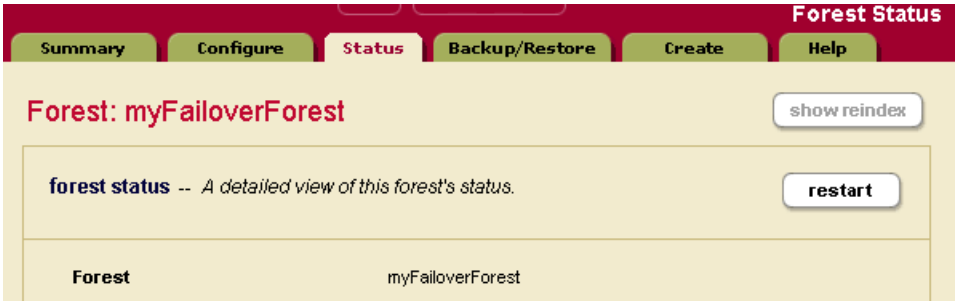
7.2 Reverting a Failed Over Forest Back to the Primary Host

If a forest fails over to a failover host, causing a replica forest to take the role of the primary forest, the replica forest will remain in the open state until the host unmounts the forest. If you have a failed over forest and want to revert it back to the original primary host (*unfailover* the forest), you must either restart the forest that is open or restart the host in which the forest open. You should only do this if the original primary forest has a state of *sync replicating*, which indicates that it is up-to-date and ready to take over. After restarting the forest that is currently *open*, the forest will automatically *open* on the primary host (if the original primary forest is in the state *sync replicating*). Make sure the primary host is back online and corrected before attempting to unfailover the forest. To check the status of the hosts in the cluster, see the Cluster Status Page in the Admin Interface. To check the status of the forest, see the Forest Status Pages in the Admin Interface.

To restart the forest, perform the following steps:

1. Navigate to the Status page for the forest that has failed over. For example, if the forest name is myFailoverForest, click Forests > myFailoverForest in the left tree menu, then click the Status tab.

2. On the Forest Status page, click the restart button.



3. Click OK on the Restart Forest confirmation page.

4. When the Forest Status page returns, if the Mount State is `unmounted`, the forest might not have completed mounting. Refresh the page and the Mount State should indicate that the forest is `open`.

The forest is restarted, and if the primary host is available, the primary host will mount the forest. If the primary host is not available, the first failover host will try to mount the forest, and so on until there are no more failover hosts to try. If you look in the `ErrorLog.txt` log file for the primary host, you will see messages similar to the following:

```
2010-09-13 20:16:47.751 Info: Mounted forest myFailoverForest locally
on /space/marklogic/Forests/myFailoverForest

2010-09-13 20:16:47.751 Info: Forest replica accepts forest
myFailoverForest as the master with timestamp 2564905551526239330
```

If you look at the `ErrorLog.txt` log file for any other host in the cluster, you will see messages similar to the following:

```
2010-09-13 20:16:47.751 Info: Forest failover1 accepts forest
myFailover as the master with timestamp 2564905551526239330

2010-09-14 17:01:29.651 Info: Forest replica1 starting synchronization
to forest failover1

2010-09-14 17:01:29.666 Info: Forest replica1 starting bulk replication
to forest failover1

2010-09-14 17:01:29.776 Info: Forest replica1 needs to replicate 0
fragments to forest failover1

2010-09-14 17:01:29.776 Info: Forest replica1 finished bulk replicated
0 fragments to forest failover1

2010-09-14 17:01:29.807 Info: Forest replica1 finished bulk replication
to forest failover1
```

```
2010-09-14 17:01:29.822 Info: Forest replica1 finished synchronizing to
replica forest failover1

2010-09-14 17:09:26.638 Info: Forest replica1 accepts forest failover1
as the master with precise time 12845094147890000
```

8.0 Configuring Shared-Disk Failover for a Forest

This chapter describes the procedure for configuring shared-disk failover for a forest. For details about how failover works and the requirements for failover, see "High Availability of Data Nodes With Failover" on page 25. For details on configuring local-disk failover, see "Configuring Local-Disk Failover for a Forest" on page 42. This chapter includes the following sections:

- Setting Up Shared-Disk Failover for a Forest
- Reverting a Failed Over Forest Back to the Primary Host

For other failover administrative procedures that apply to both local-disk and shared-disk failover, see "Other Failover Configuration Tasks" on page 52.

8.1 Setting Up Shared-Disk Failover for a Forest

Setting up shared-disk failover for a forest is a relatively simple administrative process. This section describes this procedure. There are two basic parts to the procedure:

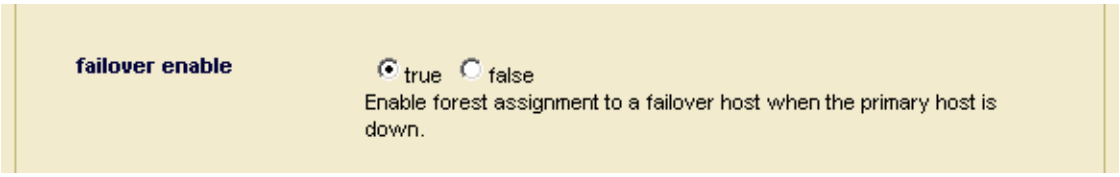
- Enabling Failover in a Group
- Setting Journaling Database Option to Strict
- Configuring Shared-Disk Failover For a Forest

8.1.1 Enabling Failover in a Group

For each group in which you want to host a failover forest, perform the following steps:

1. Before setting up failover, ensure that you have met all the requirements for failover, as described in "Requirements for Shared-Disk Failover" on page 33.

2. In the groups configuration page for the group in which the failover host belongs, make sure the `failover enable` button is set to `true`.



   This group-level `failover enable` button provides global control, at the group level, for enabling and disabling failover for all forests in that group.

You can enable or disable failover at the group level at any time. It does not stop you from configuring failover for forests; it only stops failover from actually occurring.

8.1.2 Setting Journaling Database Option to Strict

In the database in which you plan to attach shared-disk failover forests, MarkLogic recommends setting the `journaling` option to `strict`. The `strict` setting does an explicit file sync to the journal before committing the transaction. The default `fast` setting syncs to the operating system, but does not do an explicit file sync, and instead relies on the operating system to write to the disk. If the operating system fails suddenly (for example, a power failure or a system crash) after the transaction is committed but before the journal has been written to disk, in `fast` mode, it is possible to lose a transaction, but in `strict` mode you will not lose any transactions.

Setting the `journaling` option to `strict` will slow each transaction down (because it has to wait for the filesystem to report back that the journal file was written to disk). In some cases, the slowdown can be significant, and in other cases it can be relatively modest, but in all cases it will be slower. If you are doing many small transactions one after another, the slowdown can be considerable. In these cases, the best practice is to batch the updates into larger transactions (for example, update 100 documents per transaction instead of 1 document per transaction). The optimal size of your batches will vary based on the size of your documents, the hardware in which you are running MarkLogic Server, and other factors. You should perform your own performance benchmarks with your own workload to determine your optimum batch size.

The reason for setting the option to `strict` for shared-disk failover is that, with shared-disk failover, you are specifically trying to guard against hosts going down with only a single copy of the data. With local-disk failover, the replica forests each store a copy of the data, so not synchronizing the filesystem for each transaction is less risky (because the master host as well as all of the replica hosts would need to have the operating system fail suddenly in order to potentially lose a transaction), and it therefore tends to be more of an acceptable performance trade-off to set `journaling` to `fast` (although you could use `strict` if you want to be even safer). You should evaluate your requirements and SLAs when making these trade-offs, as what is acceptable for one environment might not be acceptable for another.

To set the `journaling` database option to `strict`, perform the following steps:

1. In the Admin Interface, navigate to the database configuration page for the database to which you will attach your shared-disk failover forests.

2. On the database configuration page, find the `journaling` option and set it to `strict`.

3. Click OK to save the changes.

All forests attached to this database will use `strict journaling`.

8.1.3 Configuring Shared-Disk Failover For a Forest

To set up failover on a forest, perform the following steps:

1. Before setting up failover, ensure that you have met all the requirements for failover, as described in “Requirements for Shared-Disk Failover” on page 33 and enable failover for the group, as described in “Enabling Failover in a Group” on page 47.

2. Either create a new forest or enable failover for an existing forest with a data directory on a supported CFS. If you are modifying an existing forest, skip to step 6. To create a new forest, first click the Forests link in the left tree menu, then click the Create tab. The Create Forest page appears. Note the `failover enable` button and the `failover hosts` section at the bottom.

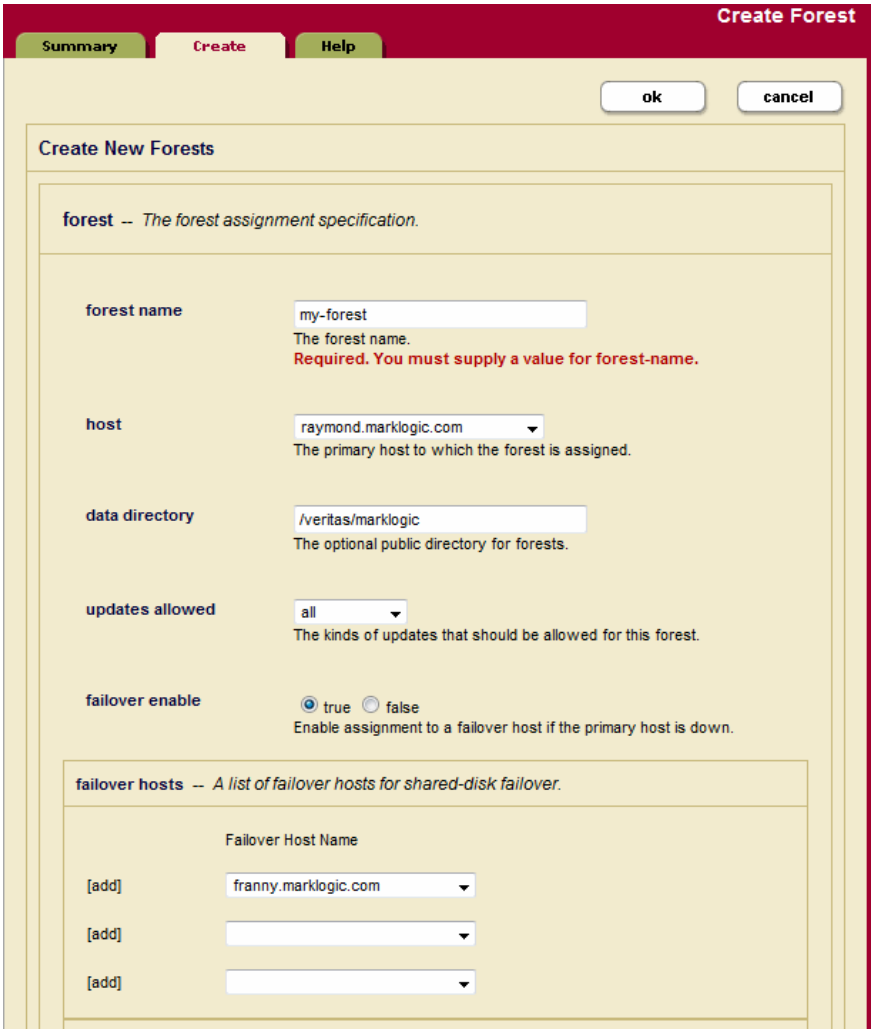
3. Enter a name for the forest.

4. In the Host drop-down list, choose the primary host for this forest. This is the host that will service this forest unless the forest fails over to another host.

5. Specify a data directory for the forest that is on a supported CFS. For example, if the CFS is mounted to `/veritas/marklogic` on your primary host and all of your failover hosts, specify `/veritas/marklogic`.

6. Select `true` for `failover enable`. Note that `failover enable` must be set to `true` at both the forest and the group level for failover to be active.

7. In the first drop-down list in the `failover hosts` section, choose a failover host. If you are specifying more than one failover hosts, choose another one in the second list, and so on. If you need to add more than two failover hosts, you can do that after initially creating the forest.



8. Click OK to create or modify the forest.

The forest is now configured with a failover host. You must attach the forest to a database before you can use the forest, but it is ready and set up for failover.

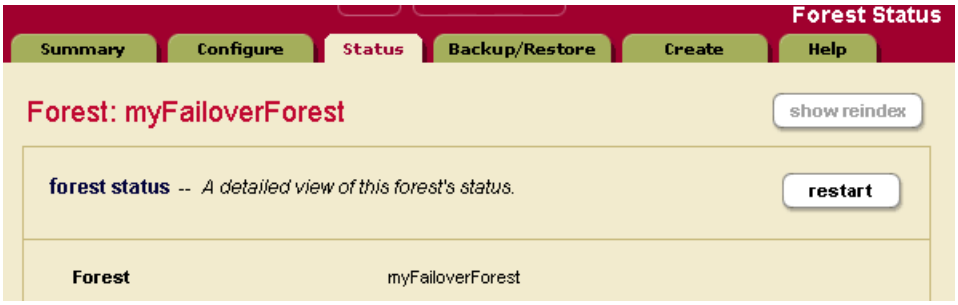
8.2 Reverting a Failed Over Forest Back to the Primary Host

If a forest fails over to a failover host, it will remain mounted locally to the failover host until the host unmounts the forest. If you have a failed over forest and want to revert it back to the primary host (*unfailover* the forest), you must either restart the forest or restart the host in which the forest is locally mounted. After restarting, the forest will automatically mount locally on the primary host if the primary host is back online and corrected. To check the status of the hosts in the cluster, see the Cluster Status Page in the Admin Interface.

To restart the forest, perform the following steps:

1. Navigate to the Status page for the forest that has failed over. For example, if the forest name is myFailoverForest, click Forests > myFailoverForest in the left tree menu, then click the Status tab.

2. On the Forest Status page, click the restart button.



3. Click OK on the Restart Forest confirmation page.

4. When the Forest Status page returns, if the Mount State is `unmounted`, the forest might not have completed mounting. Refresh the page and the Mount State should indicate that the forest is open.

The forest is restarted, and if the primary host is available, the primary host will mount the forest. If the primary host is not available, the first failover host will try to mount the forest, and so on until there are no more failover hosts to try. If you look in the `ErrorLog.txt` log file for the primary host, you will see a message similar to the following:

```
2007-03-28 13:20:29.644 Info: Mounted forest myFailoverForest locally
on /veritas/marklogic/Forests/myFailoverForest
```

If you look at the `ErrorLog.txt` log file for any other host in the cluster, you will see a message similar to the following:

```
2007-03-28 13:20:29.644 Info: Mounted forest myFailoverForest remotely
on seymour.marklogic.com
```

9.0 Other Failover Configuration Tasks

This chapter describes the other failover procedures for configuring failover for a forest. The task here apply to both local-disk and shared-disk failover. For details about how failover works and the requirements for failover, see "High Availability of Data Nodes With Failover" on page 25. For details on configuring local-disk failover, see "Configuring Local-Disk Failover for a Forest" on page 42. For details on configuring shared-disk failover, see "Configuring Shared-Disk Failover for a Forest" on page 47. This chapter includes the following sections:

- XDQP Timeout, Host Timeout, and Host Initial Timeout Parameters
- Disabling Failover for a Group or a Forest
- Moving An Existing Private Forest to a Public Directory
- Migrating the Security Database to Use Failover Forests
- Migrating Other Auxiliary Databases to Use Failover Forests

9.1 XDQP Timeout, Host Timeout, and Host Initial Timeout Parameters

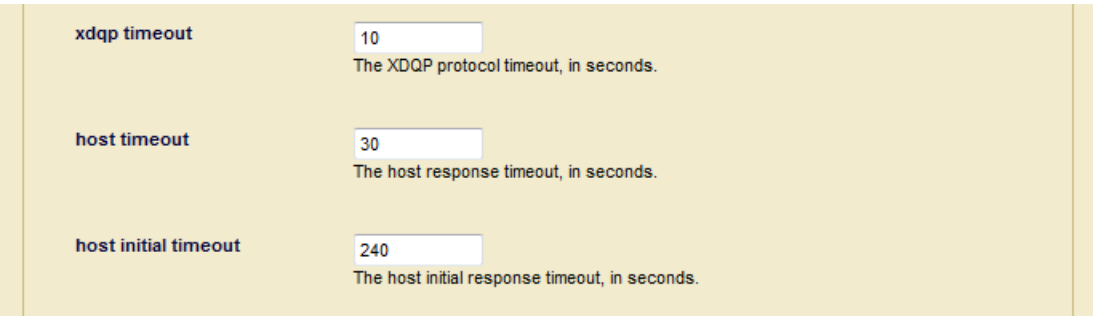
The XDQP port is the port in which each host in the cluster listens for communication with the other hosts in the cluster. This communication happens over the XDQP port, which is set to 7999 by default. The XDQP port must be open on the network so that all other hosts in the cluster can communicate over it. If the XDQP port is not available, then hosts can get `XDMP-FORESTMNT` and other errors when trying to start up. For more details about communication between nodes in a cluster, see "Communication Between Nodes" on page 23.

Each group configuration has an `xdqp timeout`, a `host timeout`, and a `host initial timeout` setting. These settings monitor activity over the XDQP port, and govern the time periods which will induce failover in various scenarios.

The `xdqp timeout` is the time, in seconds, after which communication between e-node and d-node hosts (which happens over an internal MarkLogic Server protocol named XDQP) will time out if the host is unresponsive. If an `xdqp timeout` is reached during a request (for example, during a query), a message is logged to the `ErrorLog.txt` file and the request is retried until the `host timeout` is reached, after which time, if the d-node host is still unresponsive, the request will fail with an exception. The `host timeout` is the time, in seconds, after which a host will time out if the host is responsive, and then it will be disconnected from the cluster. The `xdqp timeout` must be less than the `host timeout`, and should typically be about one-third the value of the `host timeout`.

This allows the system to restart the connection with the unresponsive host after the `xdqp timeout` occurs but before the host is disconnected from the cluster. The `host timeout` is what can trigger a forest to fail over. For details on when a forest will fail over, see "Scenarios that Cause a Forest to Fail Over" on page 35.

The default settings for `xdqp timeout` and `host timeout` should work well for most configurations. If, however, you are seeing hosts disconnect from the cluster because of timeouts, you can raise these limits (keeping the 1 to 3 ratio between `xdqp timeout` and `host timeout`). Keep in mind, though that if the hosts are timing out, there might be some other issue that is causing the timeouts (such as a network problem, a disconnected cable, and so on).



The `host initial timeout` is the time, in seconds, that an instance of MarkLogic Server will wait for another node to come online when the cluster first starts up before deciding that the node is down. This setting is designed to allow for "staggered" cluster startups, where one machine might take a little longer to reboot than another, and avoid unneeded failover of forests during this initial system startup period. The default setting is 4 minutes, and is based on the amount of time it might take for an entire system to reboot (after a power outage, for example). Failover for any forests on a particular host will not be initiated during that cluster startup for this time period. If you know that your machines take more or less time to start up, you can change the `host initial timeout` accordingly.

9.2 Disabling Failover for a Group or a Forest

You can disable failover at two levels of granularity: you disable failover for a group, or you can disable failover for an individual forest. To disable failover, navigate to the group or the forest and set `failover enable` to `false`. Then, if a primary host fails, it will not fail over.

9.3 Moving An Existing Private Forest to a Public Directory

In order to configure failover on a forest, it must be configured to use a supported clustered filesystem (for details, see “Public Forest Required” on page 34). If you have a private forest with which you want to use failover, you must move the contents of that forest into a forest configured to store its data on a supported CFS.

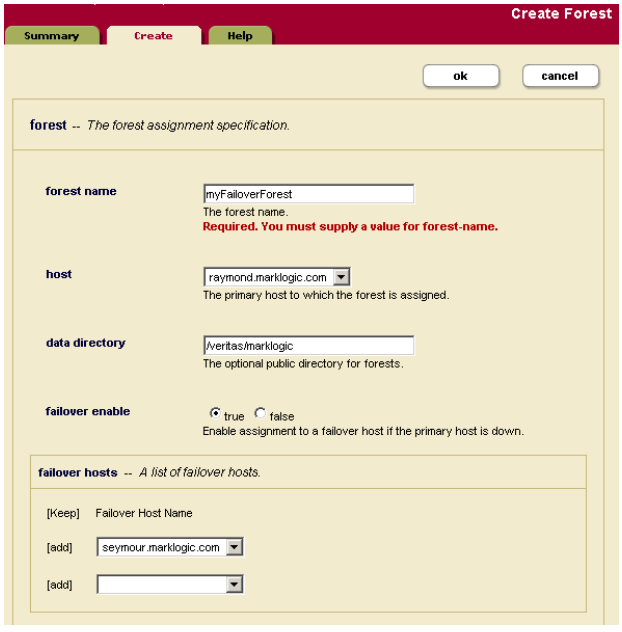
The following procedure assumes there is no activity on the forest(s) being moved. If there are any updates to the forest(s) being moved, they might end up in different states. Note that this should not be done on active systems, as there will be a short outage period between detaching the old forest and attaching the new forest.

To move an existing private forest to a public directory, perform the following steps:

1. Use the forest backup/restore page of the Admin Interface to backup the private forest. For example, for a forest named `test`, navigate to the forest `test` in the Admin Interface, select the Backup/Restore tab, enter a backup directory, select backup, and click OK. For the detailed Backup/Restore procedure, see the *Administrator's Guide*.

2. Create a new forest. For the new forest:

   - Enter the name of the directory where you want the forest data stored (the directory must be on a supported CFS).
   - Set the `failover enable` option to `true`.
   - Select one or more failover hosts.



3. For the new forest, select the Backup/Restore tab.

4. Enter the name of the directory in which you backed up your private forest in step 1.

5. Select the Restore button and click OK. Confirm the restore operation to restore the forest.

   The new forest is now configured for failover and is ready to be attached to a database. Note that if you attach this forest to a database, be sure to first detach the one in which it was restored from (see steps 8 and 9).

6. If you are moving the Security database or one of the other auxiliary databases (Schemas, Modules, and Triggers), skip the next steps and continue with the procedure described in "Migrating the Security Database to Use Failover Forests" on page 55.

7. To detach the old forest and attach the new one, continue with the remaining steps in this procedure.

8. Detach your original private forest from the database, if it is attached. For example, for a forest named `test` in a database named `testDB`, navigate to the database `testDB` in the Admin Interface, select the Forests link in the tree menu, and click the detach button corresponding to the original private forest named `test`. Then click OK to detach it.

9. Attach the newly restored public forest to the database. For example, for a new public forest named `test2` in a database named `testDB`, navigate to the database `testDB` in the Admin Interface, select the Forests link in the tree menu, and click the Attach tab. Then select the new public forest named `test2` and click OK.

10. Optionally, delete the original private forest.

9.4 Migrating the Security Database to Use Failover Forests

The Security database and all of the other auxiliary databases are set up by default to use private forests (forests that store their data in the default data directory). If you want to configure these databases to use shared-disk failover, then you must first move their forests to public forests that store their data on a supported CFS.

The following procedure creates a new database, attaches the failover-enabled forest to the new database, then changes your configuration to use the new database. This procedures uses the Security database, but will work similarly for the other auxiliary database (Schemas, Modules, and Triggers). For information on migrating the other auxiliary databases, see "Migrating Other Auxiliary Databases to Use Failover Forests" on page 56.

1. Create a new database. For example, navigate to the databases page of the Admin Interface, select the Create tab, and create a new database for example, named `SecurityFailover`). For a more detailed procedure, see the *Administrator's Guide*.

2. Perform the procedure described in "Moving An Existing Private Forest to a Public Directory" on page 54 to create a failover-enabled public forest.

3. Attach the newly restored public forest to the database. For example, for a new public forest named `security2` in a database named `SecurityFailover`, navigate to the database `SecurityFailover` in the Admin Interface, select the Forests link in the tree menu, and click the Attach tab. Then select the new public forest named `security2` and click OK.

4. In the Admin Interface, navigate to the App Server configuration page for the Admin Interface (for example, Groups > Default > App Servers > Admin [HTTP]).

   Note: This step is not needed for the auxiliary (Schemas, Modules, and Triggers) databases.

5. On the Database drop-down list, change the setting from `Security` (or what ever is the name of your security database) to the new database (in this example, `SecurityFailover`).

6. Click OK to save the changes.

7. For every database in your cluster, navigate to the Database Configuration page and change the security database to the newly created `SecurityFailover` database.

The new security database named `SecurityFailover` is now configured with a failover-enabled forest.

9.5 Migrating Other Auxiliary Databases to Use Failover Forests

The procedure to migrate the other auxiliary databases (Schemas, Modules, and Triggers) to use failover forests is similar to the previous procedure for the Security database with the following differences:

- You do not need to modify the Admin Interface App Server in step 4 and 5 above.
- For the Schemas and Triggers databases, modify the corresponding entry for each database in the cluster.
- For the Modules database, for each App Server in the cluster that uses a Modules database, select the newly created modules database.

10.0 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 Iceni 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 (eay@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 OFMERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE ANDNONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BELIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTIONOF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTIONWITH 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 IFADVISED 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.