
MarkLogic Server

Scalability, Availability, and Forest-Level Failover

Release 4.0
September, 2008

Last Revised: 4.0-7, October, 2009

Copyright

© Copyright 2002-2009 by Mark Logic Corporation. All rights reserved worldwide.

This Material is confidential and is protected under your license agreement.

Excel and PowerPoint are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. This document is an independent publication of Mark Logic Corporation and is not affiliated with, nor has it been authorized, sponsored or otherwise approved by Microsoft Corporation.

Contains LinguistX, from Inxight Software, Inc. Copyright © 1996-2006. All rights reserved. www.inxight.com.

Antenna House OfficeHTML Copyright © 2000-2008 Antenna House, Inc. All rights reserved.

Argus Copyright ©1999-2008 Icenit Technology Ltd. All rights reserved.

Contains Rosette Linguistics Platform 6.0 from Basis Technology Corporation, Copyright © 2004-2008 Basis Technology Corporation. All rights reserved.

Table of Contents

Scalability, Availability, and Forest-Level Failover

1.0	Introduction—About This Guide	5
2.0	Scalability Considerations in MarkLogic Server	6
2.1	Factors To Consider in Large-Scale Deployments	6
2.1.1	Extremely Large Amounts of Content	6
2.1.2	Fast Query Performance, Regardless of the Content Size	7
2.1.3	Large Numbers of Users	7
2.2	Forest Sizes Per Data Node Host	7
2.3	High Availability	8
2.3.1	High-Availability Features of MarkLogic Server	8
2.3.2	Planning for High Availability	10
2.3.3	High Availability With the Forest-Level Failover	10
2.4	Hardware and Memory Considerations	10
3.0	Getting Started with Enterprise Edition Distributed Deployments	12
3.1	Terminology	12
3.2	Fundamentals	14
3.3	Advantages to Distributing	16
3.4	Considerations for a Distributed System	17
3.4.1	Hosts with Distinct Roles	17
3.4.2	Scale of System	18
3.4.3	Hardware Configuration	18
3.4.4	Storage Architecture	18
3.5	Configuring a Cluster	19
3.5.1	Building a Cluster from Scratch	19
3.5.1.1	Installing the Initial Host	19
3.5.1.2	Configuring the Initial Host	20
3.5.1.3	Filling Out the Cluster	20
3.5.1.4	Configuring Forest Layout	21
3.5.2	Increasing the Size of an Existing Cluster	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	Installation	24

5.0	High Availability of Data Nodes With Forest-Level Failover	25
5.1	Problems Forest-Level Failover Addresses	25
5.2	How Forest-Level Failover Works	26
5.2.1	Forest-Level Failover	26
5.2.2	Enabling Failover	26
5.2.3	Forest Mount States	26
5.2.4	Cluster Determines If a Host is Down	27
5.2.5	Different Host Takes Over the Forest	27
5.3	Requirements for Failover	29
5.3.1	Enterprise Cluster Required	29
5.3.2	Public Forest Required	29
5.3.3	Clustered Filesystem Required, Available to All Failover Hosts	29
5.3.4	License Key Required	30
5.4	Failover Host Must Be “Ready” to Take Over	30
5.5	Scenarios that Cause a Forest to Fail Over	30
5.6	Architecting a Forest-Level Failover Solution	31
6.0	Configuring a Cluster	33
6.1	Determining Which Nodes Are in Which Group in a Cluster	33
6.2	Adding Another Node to a Cluster	33
6.3	Removing a Node from a Cluster	34
6.4	Upgrading a Cluster to a New Maintenance Release of MarkLogic Server	35
7.0	Configuring Failover for a Forest	37
7.1	Setting Up Failover for a Forest	37
7.1.1	Enabling Failover in a Group	37
7.1.2	Configuring Failover For a Forest	38
7.2	Reverting a Failed Over Forest Back to the Primary Host	40
7.3	XDQP Timeout, Host Timeout, and Host Initial Timeout Parameters	41
7.4	Disabling Failover for a Group or a Forest	42
7.5	Moving An Existing Private Forest to a Public Directory	42
7.6	Migrating the Security Database to Use Failover Forests	43
7.7	Migrating Other Auxiliary Databases to Use Failover Forests	44
8.0	Technical Support	45

1.0 Introduction—About This Guide

This *Scalability, Availability, and Forest-Level Failover* guide describes some of the features and characteristics that make MarkLogic Server scale to extremely large amounts of content. This document 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 12.

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 forest-level failover.

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

- “Scalability Considerations in MarkLogic Server” on page 6 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 12 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 Forest-Level Failover” on page 25 describes forest-level failover, including describing the problems it addresses and listing the requirements for using it.
- “Configuring a Cluster” on page 33 describes the basic steps needed to configure a MarkLogic Server cluster.
- “Configuring Failover for a Forest” on page 37 provides the detailed steps to configure forest-level failover. This includes procedures to set up failover-enabled forests and procedures to migrate forests that are not failover-enabled to become failover-enabled.

2.0 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.

2.1 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 Mark Logic 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](#)

2.1.1 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.

2.1.2 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.

2.1.3 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.

2.2 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. If the content is being updated on a regular basis, each forest should ideally have two CPUs (or cores) per forest. That allows 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.

2.3 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 Forest-Level Failover](#)

2.3.1 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 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.

2.3.2 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 forest-level failover (see “High Availability With the Forest-Level Failover” on page 10). 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.

2.3.3 High Availability With the Forest-Level Failover

In addition to the standard high-availability features listed above, MarkLogic Server provides high availability for content hosted on d-nodes with forest-level failover. The forest-level 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. For details about forest-level failover, see “High Availability of Data Nodes With Forest-Level Failover” on page 25.

2.4 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 tend 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.

3.0 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](#)

3.1 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.

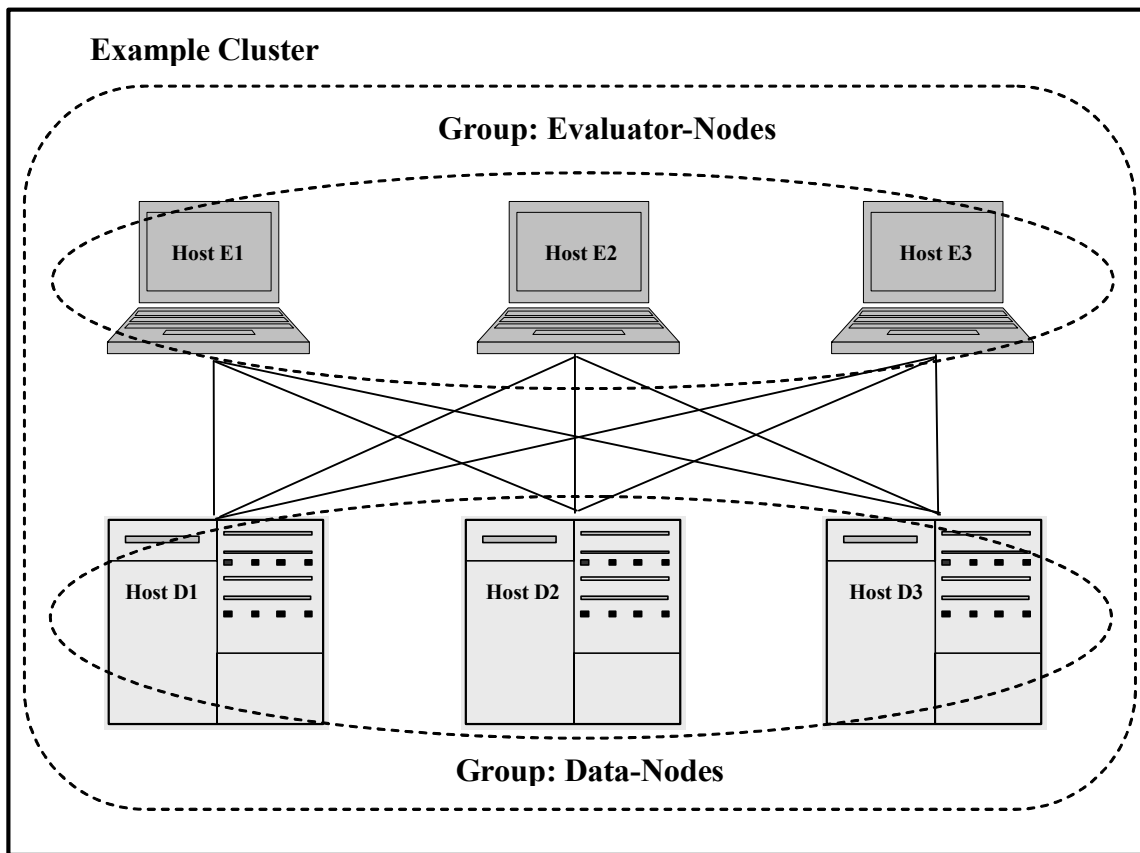
*group*A group is a set of hosts with uniform HTTP, WebDAV and XDBC server configurations (but not necessarily uniform forest configurations). Groups are used to simplify cluster management.

*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.

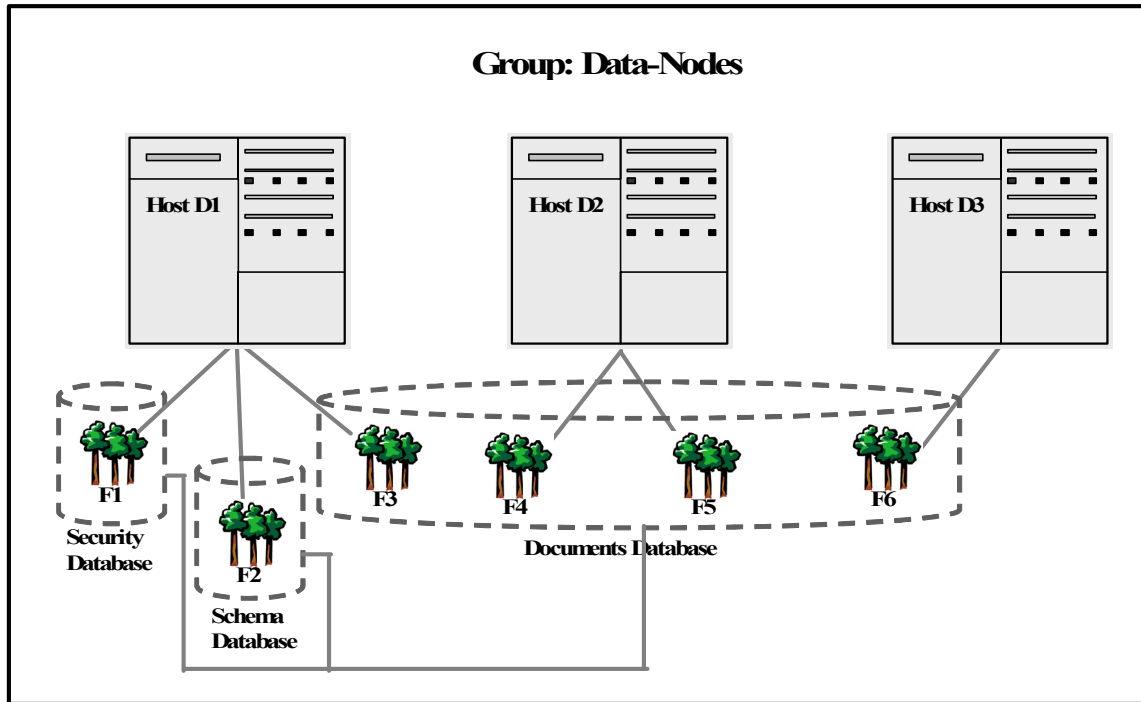
*database*A database is a set of one or more forests that appears as a single contiguous set of content for query purposes. Each forest in a database must be configured consistently. HTTP, WebDAV, and XDBC servers evaluate queries against a single database. In addition to databases created by the administrator 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:



3.2 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 Mark Logic’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. MarkLogic Server does not support replicated forests across multiple hosts, either for improved parallelism or redundancy.

3.3 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.

3.4 Considerations for a Distributed System

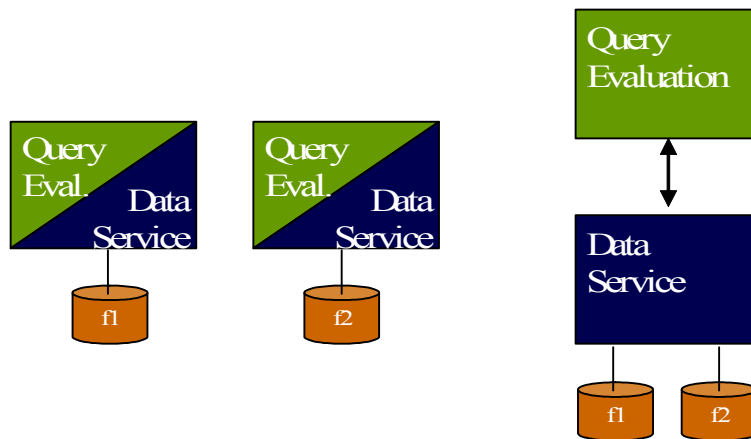
Just as there are different reasons to deploy a distributed architecture, there are different issues to consider in determining a distribution strategy. Each strategy provides certain benefits and involves certain other compromises. In order to determine how best to deploy MarkLogic Server to meet your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

This section includes the following topics:

- [Hosts with Distinct Roles](#)
- [Scale of System](#)
- [Hardware Configuration](#)
- [Storage Architecture](#)

3.4.1 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 in 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.

3.4.2 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.

3.4.3 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.

3.4.4 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 Mark Logic professional services consultant for detailed deployment architecture advice.

3.5 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](#)

3.5.1 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](#)

3.5.1.1 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 Mark Logic 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.

3.5.1.2 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.

3.5.1.3 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.

3.5.1.4 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.

3.5.2 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.

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](#)
- [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 12.

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 12.

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 will stop 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. In order to vote a node out of the cluster, there must be a *quorum* of nodes voting to remove a node. A quorum occurs if 50% or more of the *total* number of nodes in the cluster (including any nodes that are down) vote the same way. 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 least half 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 41).

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 allows the cluster to recover gracefully from things like hardware failures or other events that might make a host unresponsive. This occurs automatically, without any 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 will go 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 will attempt 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 Forest-Level Failover” on page 25.

4.4 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 Forest-Level Failover

MarkLogic Server Enterprise Edition provides support for forest-level failover, which allows you to specify alternate instances of MarkLogic Server to host forests in the event of a forest's primary host going offline. Forest-level failover provides a high-availability solution for data nodes. This chapter describes forest-level failover, and includes the following sections:

- [Problems Forest-Level Failover Addresses](#)
- [How Forest-Level Failover Works](#)
- [Requirements for Failover](#)
- [Failover Host Must Be “Ready” to Take Over](#)
- [Scenarios that Cause a Forest to Fail Over](#)
- [Architecting a Forest-Level Failover Solution](#)

For details about creating and managing forests and databases, see the *Administrator's Guide*.

5.1 Problems Forest-Level Failover Addresses

Forest-level 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 forest-level 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 mount 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.

Forest-level 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.

Forest-level failover provides both high levels of availability and data integrity in the event of a data node failure. Forest-level 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 How Forest-Level Failover Works

This section describes how forest-level failover works in MarkLogic Server and includes the following parts:

- [Forest-Level Failover](#)
- [Enabling Failover](#)
- [Forest Mount States](#)
- [Cluster Determines If a Host is Down](#)
- [Different Host Takes Over the Forest](#)

5.2.1 Forest-Level Failover

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 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. For example, if you have 10 hosts in a cluster, you can configure between 1 and 9 failover hosts for a single forest.

5.2.2 Enabling Failover

You must enable forest-level failover before you can use it. Enabling failover requires a failover-enabled license key, 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 forest-level failover administration on complex systems. For details on configuring failover, see “Configuring Failover for a Forest” on page 37.

5.2.3 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.

5.2.4 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.

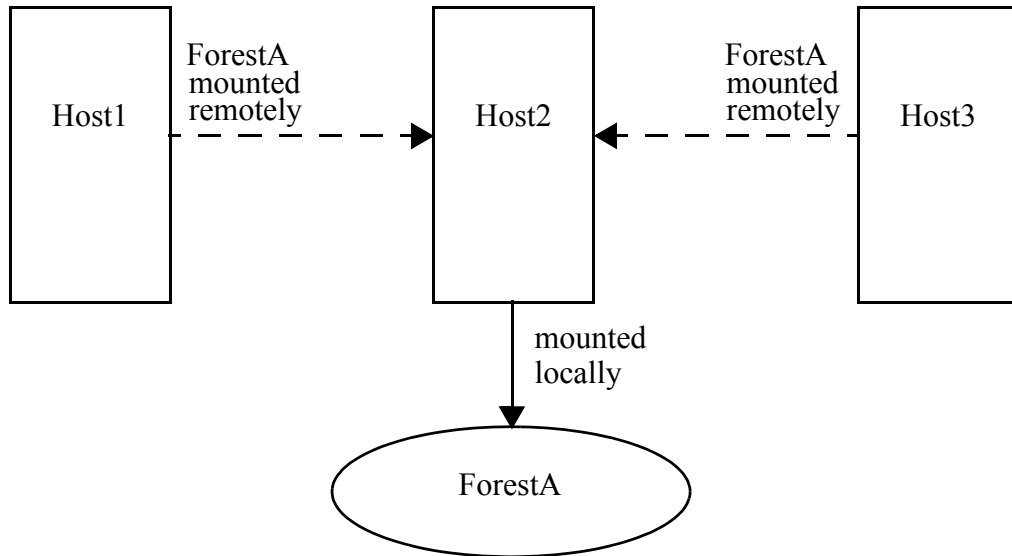
The cluster uses a voting algorithm to determine if a host is down. The voting algorithm gets its data from each host’s view of the cluster. If there is a quorum of hosts, each of whose view of the cluster is such that it believes a particular host is down, then the other hosts in the cluster treat that host as if it is down and tries 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.

5.2.5 Different Host Takes Over the Forest

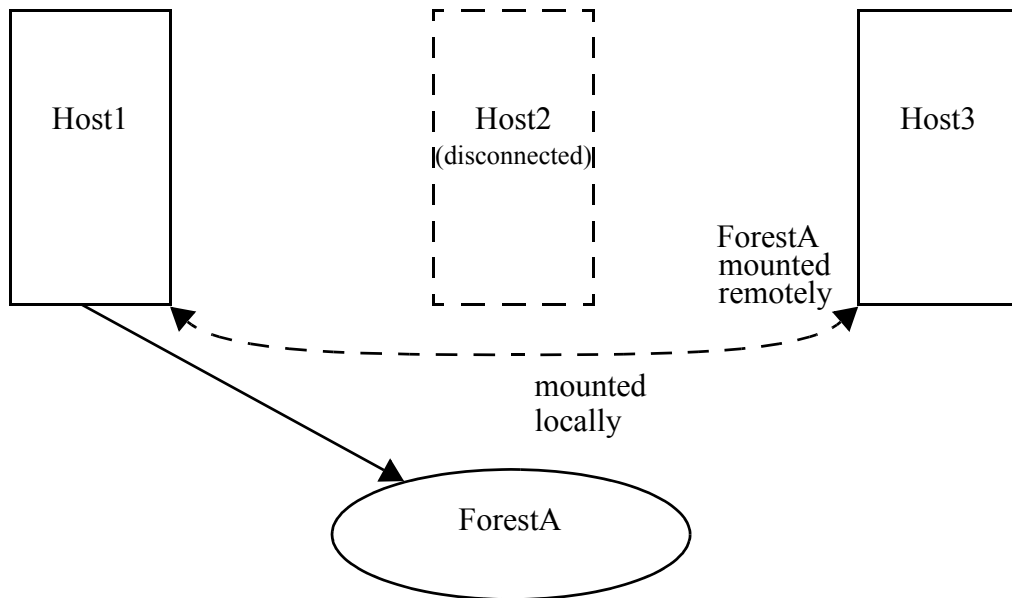
After the cluster has determined that a host is down and disconnected it from the cluster, if failover is enabled and configured for that forest, then one of the failover hosts will attempt to mount the forest locally. The host that attempts to mount the forest is determined based on the list of failover 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 40.

5.3 Requirements for Failover

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

- [Enterprise Cluster Required](#)
- [Public Forest Required](#)
- [Clustered Filesystem Required, Available to All Failover Hosts](#)
- [License Key Required](#)

5.3.1 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.

5.3.2 Public Forest Required

To enable 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 42) before it can be configured as a failover forest.

5.3.3 Clustered Filesystem Required, Available to All Failover Hosts

In addition to a public forest directory (described above), forest-level failover requires the forest to have its data directory reside on a supported clustered file system (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.

Forest-level failover in MarkLogic Server is supported on the following CFSs:

- Veritas VxFS 4.1 on Sun Solaris
- Veritas VxFS 4.1 on 64-bit Red Hat Linux 4
- Veritas VxFS 5.0 on 64-bit Red Hat Linux 4
- Red Hat GFS on 64-bit Linux (build 2.6.9-58.3 or greater)

5.3.4 License Key Required

Failover requires a license key that enables the failover functionality. If your license key does not enable failover, then you will not be able to configure forests to have failover hosts, and you will not be able to use failover. If you need a failover license key, contact Mark Logic Technical Support.

5.4 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.

5.5 Scenarios that Cause a Forest to Fail Over

In general, forest-level failover provides the ability for a forest to change hosts in the event that its host goes 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 forest-level failover works, see “How Forest-Level Failover Works” on page 26. For information about the forest mount states, see “Forest Mount States” on page 26.

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 41.

- 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).

5.6 Architecting a Forest-Level Failover Solution

The simplest conceptual deployment for 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, 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 above. 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.

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 Mark Logic 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* 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 Mark Logic 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. You must resolve any forest dependencies with other hosts 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 Mark Logic software. Therefore, if you install a maintenance release, you need to install it on all of the hosts in the cluster.

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 Mark Logic 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 Failover for a Forest

This chapter describes the procedure for configuring failover for a forest. For details about how failover works and the requirements for failover, see “High Availability of Data Nodes With Forest-Level Failover” on page 25. This chapter includes the following sections:

- [Setting Up Failover for a Forest](#)
- [Reverting a Failed Over Forest Back to the Primary Host](#)
- [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](#)

7.1 Setting Up Failover for a Forest

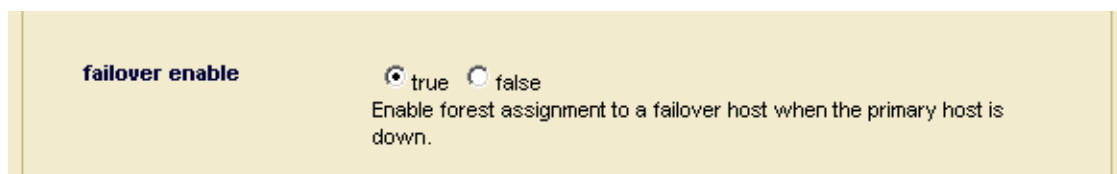
Setting up failover for a forest is a relatively simple administrative process. This section describes this procedure. There are two basic parts to the procedure:

- [Enabling Failover in a Group](#)
- [Configuring 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 Failover” on page 29.
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.

7.1.2 Configuring 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 Failover” on page 29 and enable failover for the group, as described in “Enabling Failover in a Group” on page 37.
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; if you do not have a license key that allows forest-level failover, these parts will not appear on this page.

Create Forest

Summary Create Help

ok cancel

forest -- *The forest assignment specification.*

forest name
The forest name.
Required. You must supply a value for forest-name.

host
The primary host to which the forest is assigned.

data directory
The optional public directory for forests.

failover enable true false
Enable assignment to a failover host if the primary host is down.

failover hosts -- *A list of failover hosts.*

[Keep]	Failover Host Name
[add]	<input type="text"/>
[add]	<input type="text"/>

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.

- 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`.
- 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.
- 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. If you need to add more than two failover hosts, you can do that after initially creating the forest.

Create Forest

Summary Create Help

ok cancel

forest -- *The forest assignment specification.*

forest name
The forest name.
Required. You must supply a value for forest-name.

host
The primary host to which the forest is assigned.

data directory
The optional public directory for forests.

failover enable true false
Enable assignment to a failover host if the primary host is down.

failover hosts -- *A list of failover hosts.*

[Keep]	Failover Host Name
[add]	<input type="text" value="seymour.marklogic.com"/>
[add]	<input type="text"/>

- Click OK to create or modify the forest.

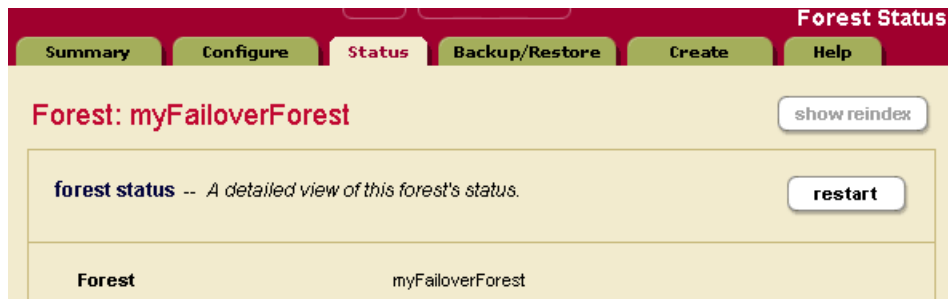
The forest is now configured with a failover host. You must attach the forest to a database before you can use the forest, but it is ready and set up for failover.

7.2 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:

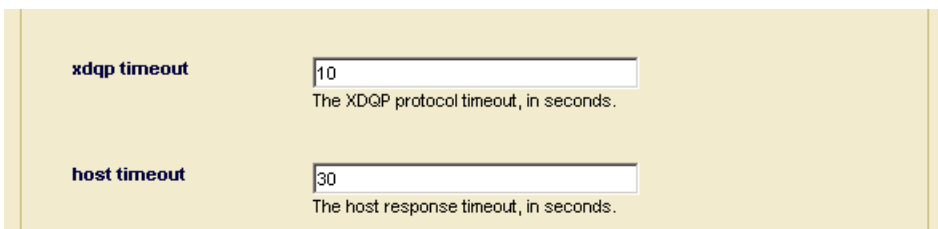
```
2007-03-28 13:20:29.644 Info: Mounted forest myFailoverForest remotely
on seymour.marklogic.com
```

7.3 XDQP Timeout, Host Timeout, and Host Initial Timeout Parameters

Each group configuration has an `xdqp timeout`, a `host timeout`, and a `host initial timeout` setting. These settings 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 30.

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 image shows a configuration interface with two input fields. The first field is labeled "xdqp timeout" and contains the value "10". Below it is the text "The XDQP protocol timeout, in seconds." The second field is labeled "host timeout" and contains the value "30". Below it is the text "The host response timeout, in seconds."

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.

7.4 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.

7.5 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 file system (for details, see “Public Forest Required” on page 29). 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.

The screenshot shows the 'Create Forest' dialog box. It has a title bar with 'Create Forest' and three tabs: 'Summary', 'Create', and 'Help'. There are 'ok' and 'cancel' buttons at the top right. The main area is titled 'forest -- The forest assignment specification.' and contains the following fields:

- forest name:** A text input field containing 'myFailoverForest'. Below it is the text 'The forest name.' and a red error message: 'Required. You must supply a value for forest-name.'
- host:** A dropdown menu showing 'raymond.marklogic.com'. Below it is the text 'The primary host to which the forest is assigned.'
- data directory:** A text input field containing '/veritas/marklogic'. Below it is the text 'The optional public directory for forests.'
- failover enable:** Radio buttons for 'true' (selected) and 'false'. Below it is the text 'Enable assignment to a failover host if the primary host is down.'

Below these fields is a section titled 'failover hosts -- A list of failover hosts.' containing a table with the following structure:

[Keep]	Failover Host Name
[add]	seymour.marklogic.com
[add]	

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 43.
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.

7.6 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 forest-level 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 procedure 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 44.

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 42 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.

7.7 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.

8.0 Technical Support

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

We invite you to visit our support website at <http://support.marklogic.com> to access our full suite of documentation and help materials.

If you have questions or comments, you may contact Mark Logic 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.